F.E. Sem - 2 CBCGS SPA – MAY 17

Q.1.a.) Convert 0010 0100 0010 1101 from base 2 to decimal. Convert 134 from base 10 to hexadecimal. Write steps for conversion.

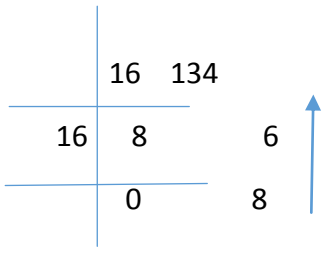
Ans:

Binary to decimal: The steps to be followed are:

1. Multiply the binary digits (bits) with powers of 2 according to their positional weight.
2. The position of the first bit (going from right to left) is 0 and it keeps on incrementing as you go towards left for every bit.
3. Given: $(0010\ 0100\ 0010\ 1101)_2 = (?)_{10}$
4. $0*2^{15}+0*2^{14}+1*2^{13}+0*2^{12}+0*2^{11}+1*2^{10}+0*2^9+0*2^8+0*2^7+0*2^6+1*2^5+0*2^4+1*2^3+1*2^2+0*2^1+1*2^0$
   $=0+0+8192+0+0+1024+0+0+0+0+32+0+8+4+1$
   $=9261$
   $(0010\ 0100\ 0010\ 1101)_2 = (9261)_{10}$

Decimal to hexadecimal: The steps to be followed are:

1. Divide the decimal number by 16. Treat the division as an integer division.
2. Write down the remainder (in hexadecimal).
3. Divide the result again by 16. Treat the division as an integer division.
4. Repeat step 2 and 3 until result is 0.
5. The hexadecimal value is digit sequence of the reminders from last to first.
6. Given: $(134)_{10} = (?)_{16}$



$(134)_{10} = (86)_{16}$

Q.1.b) Enlist all data types in C language along with their memory requirement, format specifiers and range.

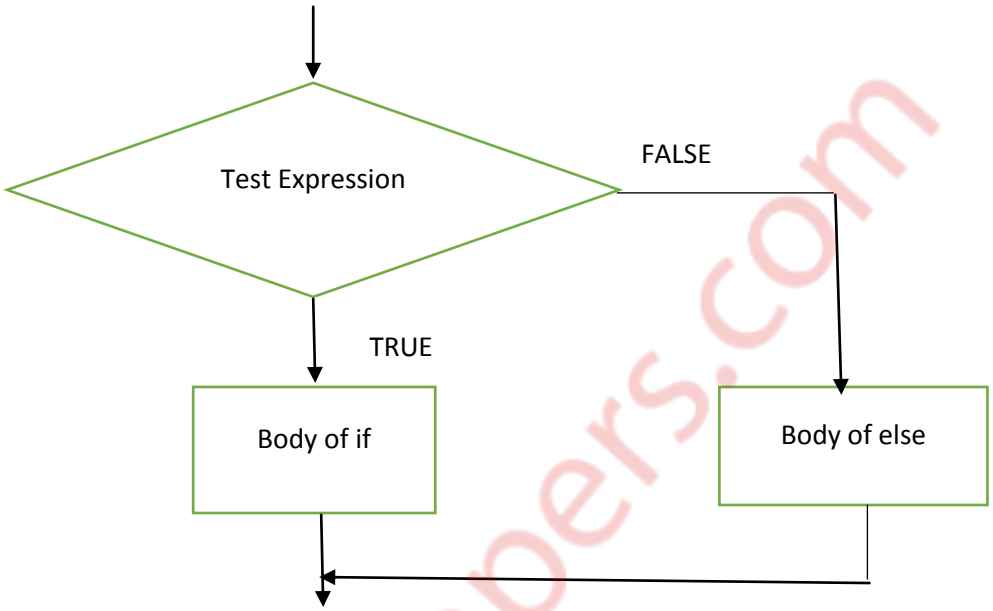
Ans:

Sr. NO.	Data Type	Type of data to be stored	Range	Space required in memory (bytes)
1.	Char	1 character in ASCII form	-127 to 128	1
2.	**Signed char**	**1 character in ASCII**	**-127 to 128**	**1**
3.	**Unsigned char**	**1 character in ASCII form**	**0 to 255**	**1**
4.	**Int**	**Integer nos.**	**-32768 to 32767**	**2**
5.	**Signed Int**	**Integer nos.**	**-32768 to 32767**	**2**
6.	**Unsigned Int**	**Integer nos.**	**0 to 65535**	**2**
7.	**Short Int**	**Integer nos.**	**-32768 to 32767**	**2**
8.	**Long int**	**Integer nos.**	**-2147483648 to 2147483647**	**4**
9.	**Signed short int**	**Integer nos.**	**-32768 to 32767**	**2**
10.	**Signed Long int**	**Integer nos.**	**-2147483648 to 2147483647**	**4**
11.	**Unsigned short int**	**Integer nos.**	**0 to 65535**	**2**
12.	**Unsigned long int**	**Integer nos.**	**0 to 4294967296**	**4**
13.	**Float**	**Fraction nos.**	**$3.4e^{-38}$ to $3.4e^{38}$**	**4**
14.	**Double**	**Fraction nos.**	**$1.7e^{-308}$ to $1.7e^{308}$**	**8**
15.	**Long double**	**Fraction nos.**	**$3.4e^{-4932}$ to $1.1e^{4932}$**	**10**

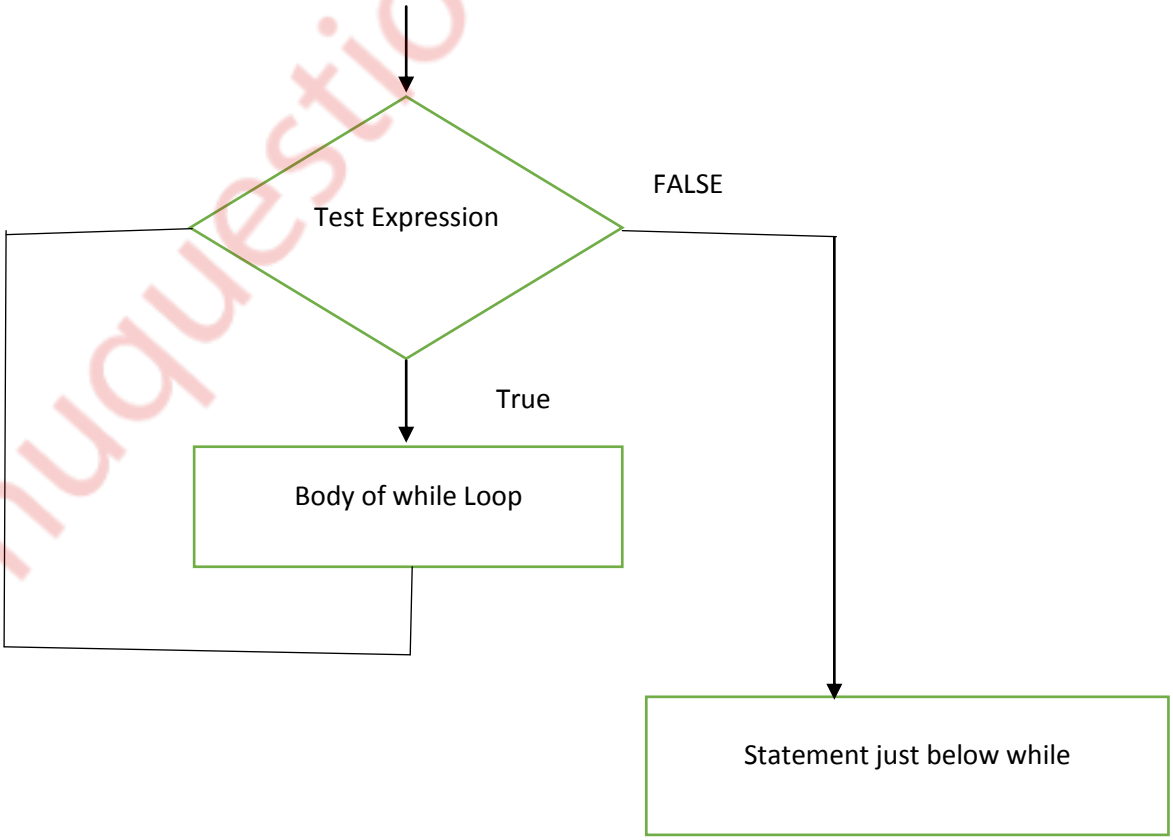
Q.1.c) Draw flowchart for "if…else" and "while" statement of C language.

Ans:

If…..else flowchart



While flowchart



Q.1.d) What is the usage of storage classes? Explain extern storage classes with suitable example.

Ans:

1. The different locations I the computer where we can store data and their accessibility, initial values etc. vary based on the way they are declared. These different ways are termed as different storage classes.
2. In C, we have four storage classes namely
   i. Automatic
   ii. Register
   iii. Static
   iv. Extern or Global.

Extern Classes:

1. In this case data is stored in memory.
2. The initial variable of such variable is zero.
3. The scope of the variable is zero i.e. it is accessible from anywhere in the program.
4. The life of such variable is till the program is alive.
5. Let us see the example

```
#include<stdio.h>
#include<conio.h>
int a=5;
void main
{
 Int a=10;
 printf("%d\n",a);
 printf("%d\n",::a);
a=::a+a;
printf("%d\n",a);
printf("%d\n",::a);
::a=a;
printf("%d\n",a);
printf("%d\n",::a);
getch();
}
```

Output:

10

5

15

5

15

15

Q.1.e) Differentiate between structure and union.

Ans:

Sr. NO.	Structure	Union
1.	Memory allotted to structure is equal to the space require collectively by all the members of the structure.	Memory allotted for a union is equal to the space required by the largest memory of that union
2.	Data is more secured in structure.	Data can be corrupted in a union.
3.	Structure provide ease of programming.	Unions are comparatively difficult for programming.
4.	Structures require more memory.	Unions require less memory.
5.	Structure must be used when information of all the member elements of a structure are to be stored.	Unions must be used when only one of the member elements of the union is to be stored.

Q.2.a) What is the need of computer programming. What do you mean by structured programming? Develop an ALGORITHM and FLOWCHART to find reverse of a number.

Ans:

1. Programming is to give the machine a list of steps to perform a particular task.
2. If the system to which programming is done is a computer than it is called as Computer programming.
3. A programming of any system has to be done in the language understood by that system.
4. Programming languages are an interface between humans and computer, they allow us to communicate with computers in order to do awesome things.
5. Structured programming (SP) is a technique devised to improve the reliability and clarity of programs
6. In SP, control of program flow is restricted to three structures, sequence, IF THEN ELSE, and DO WHILE, or to a structure derivable from a combination of the basic three. Thus, a structured program does not need to use GO TOs or branches (unless it is written in a language that does not have statement forms corresponding to the SP structures, in which case, GO TOs may be used to simulate the structures).
7. The result is a program built of modules that are highly independent of each other.
8. In turn, this allows a programmer to be more confident that the code contains fewer logic errors and will be easier to debug and change in the future.
9. However, SP may be less efficient than an unstructured counterpart.

Algorithm:

1. START

2. PRINT “Enter a number”

3. INPUT n.

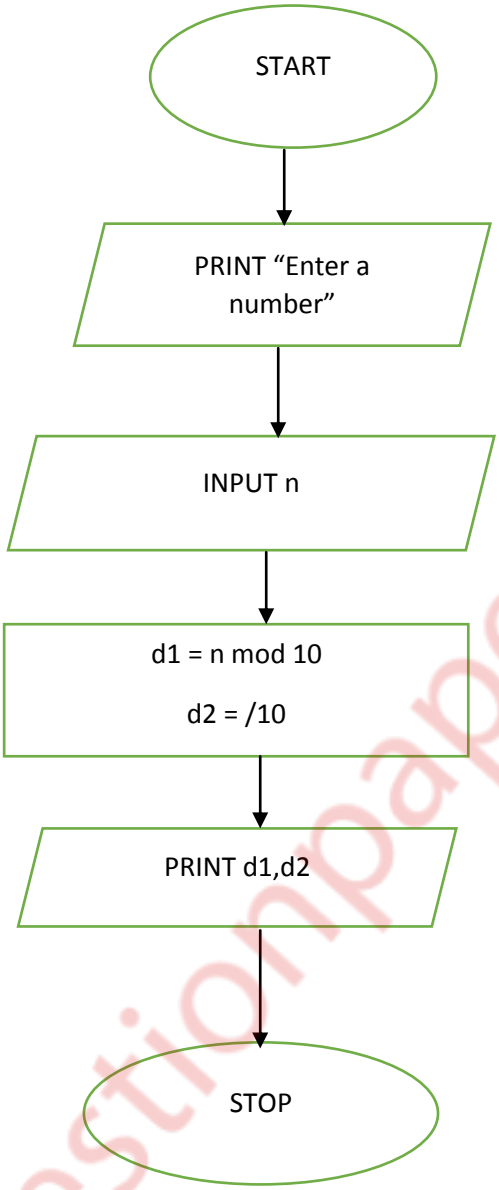
4. d1 = n mod 10

5. d2 = n/10

6. PRINT d1, d2.

7. STOP.

Flowchart:



Q.2.b) Write a menu driven program to perform arithmetic operations on two integers. The menu should be repeated until user selects "STOP" option. Program should have independent user defined functions for each case.

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int operator,ans;
double firstNumber,secondNumber;
clrscr();
 do
{
printf("Enter a operation:\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Divison");
scanf("%d", &operator);
printf("Enter two operands: ");
scanf("%lf %lf",&firstNumber, &secondNumber);
switch(operator)
{
  case 1:
    printf("%.1lf + %.1lf = %.1lf",firstNumber, secondNumber, firstNumber +
secondNumber);
    break;
  case 2:
    printf("%.1lf - %.1lf = %.1lf",firstNumber, secondNumber, firstNumber -
secondNumber);
    break;
  case 3:
    printf("%.1lf * %.1lf = %.1lf",firstNumber, secondNumber, firstNumber *
secondNumber);
    break;
```

```
    case 4:
        printf("%.1lf / %.1lf = %.1lf",firstNumber, secondNumber, firstNumber /
secondNumber);
        break;
    default:
        printf("Error! operator is not correct");
  }
  printf("\nDo you want to continue 1.YES 2. NO");
  scanf("%d",&ans);
  }
  while(ans==1);
  getch();
}
```

Output:

Enter an Operation:

1. Addition
2. Subtraction
3. Multiplication
4. Division

1
Enter two operands 10 20
10.0 + 20.0 = 30.0
Do you want to continue 1.YES 2.NO 1

Enter an operation

1. Addition
2. Subtraction
3. Multiplication
4. Division

2
Enter two operands: 50 60
50.0 – 60.0 = -10.0
Do you want to continue 1.YES 2.NO 2

Q.3.a) Write a program to create two integer of size 8 and 7. Initialize the arrays with random values. Sort the arrays in ascending order with the help of user defined function namely "sort array". Merge these arrays with the help of another user defined function named "merge arrays" which returns a new array. Program should display the arrays before and after sorting, also the merged arrays.

Ans:

```
#include<conio.h>
#include<stdio.h>
void main()
{
    int a[25],b[25],sum[50],i,j,k=1,n,m,s,temp;
    clrscr();
    printf("Enter the number of element in first array :");
    scanf("%d",&n);
    printf("\nEnter the element of array :\n");
    for(i=1;i<=n;i++)
    scanf("%d",&a[i]);
    printf("\nEnter the number of element in second array :");
    scanf("%d",&m);
    printf("\nEnter the element of array :\n");
    for(i=1;i<=m;i++)
    scanf("%d",&b[i]);
    s=m+n;
    for(i=1;i<=s;i++)
    {
        if(i<=n)
        {
            sum[i]=a[i];
        }
        else
```

```
        {
                sum[i]=b[k];
                k=k+1;
        }
}
printf("\n Array before sorting is\n");
for(i=1;i<=s;i++)
printf("%d\t",sum[i]);
for(i=1;i<=s;i++)
{
        for(j=1;j<=s;j++)
        {
                if(sum[i]<=sum[j])
                {
                        temp=sum[i];
                        sum[i]=sum[j];
                        sum[j]=temp;
                }
        }
}

printf("\nElement of array after sorting is :\n");
for(i=1;i<=s;i++)
printf("%d\t",sum[i]);
getch();
}
```

Output:

Enter the number of elements in first array: 8

Enter the element of array:

1 4 7 8 9 22 66 11

Enter the number of elements in second array: 7

33 55 88 14 16 18 79

Array before sorting is

1 4 7 8 9 22 66 11 33 55 88 14 16 18 79

Array after sorting is

1 4 7 8 9 11 14 16 18 22 33 55 66 79 88

Q.3.b) What are advantages and disadvantages of recursion? Comment on conditions to be considered while writing a recursive function. Write a program to print Fibonacci series up to N terms using a recursive function.

Ans:

Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.

Advantages:

1. Reduce unnecessary calling of functions.
2. Through Recursion one can solve problems in easy way while its iterative solution is very big and complex.
3. Extremely useful when applying the same solution.

Disadvantages:

1. Recursive solution is always logical and it is very difficult to trace.

2. In recursive we must have an if statement somewhere to force the function to return without the recursive call being executed, otherwise the function will never return.

3. Recursion uses more processor time.

i. A recursive function must definitely have a condition that exits from calling a function again.

ii. Hence there must be condition that calls the function itselfif that condition is true.

iii. If the condition is false then it will exit from the loop of calling itself again.

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,i;
 int fibo(int,int,int);
 clrscr();
 printf(“Enter the number of elements”);
 scanf(“%d”,&n);
 printf(“0\n”);
 for(i=1;i<=n-1;i++)
 {
   printf(“%d\n”,fibo(0,1,i));
 }
 getch();
}
int fibo(int a, int b int i)
{
 if(i==1)
 return b;
 else
 return(fibo(b,a+b,--i));
}
```

Output:

```
Enter the number of elements:10
0
1
1
2
3
5
8
13
21
34
```

Q.4.a) What are structures? Comment on nested structures. Write a program to read Title, Author and price of 10 books using array of structures. Display the records in ascending order of Price.

Ans:

Structure:

1. Structure is a collection of multiple data elements that can be of different data types.

2. Array is a collection of data items of same type, while structure can have data items of different types.

3. The memory space required to store one variable of structure is equal to the memory space required by all the elements independently to be stored in memory.

4. syntax

```
struct structure_name
{
  data_type variable_name;
  data_type variable_name;
  -
  -
}
```

Program:

```
#include <stdio.h>

struct book
{
   int price;
   char title[80];
   char author[80];
};

void accept(struct book list[80], int s);
void display(struct book list[80], int s);
```

```
void bsortDesc(struct book list[80], int s);

void main()
{

   struct book data[20];
   int n;
   clrscr();

   accept(data,10);
   bsortDesc(data, 10);
   display(data, 10);
   getch();
}

void accept(struct book list[80], int s)
{
   int i;
   for (i = 0; i < s; i++)
   {

         printf("\nEnter title : ");
         scanf("%s", &list[i].title);

         printf("Enter Author name: ");
         scanf("%s",&list[i].author);
         printf("Enter price : ");
         scanf("%d", &list[i].price);
   }
```

```
}

void display(struct book list[80], int s)
{
   int i;

   printf("\n\nTitle\t\tAuthor\t\tprice\n");
   printf("-------------------------------------\n");
   for (i = 0; i < s; i++)
   {
         printf("%s\t\t%s\t\t%d\n", list[i].title, list[i].author, list[i].price);
   }
}

void bsortDesc(struct book list[80], int s)
{
   int i, j;
   struct book temp;

   for (i = 0; i < s ; i++)
   {
         for (j = 0; j < (s -i); j++)
         {
            if (list[j].price > list[j + 1].price)
            {
                  temp = list[j];
                  list[j] = list[j + 1];
                  list[j + 1] = temp;
            }
```

```
        }
    }
}
```

Output:

Enter Title: physics

Enter Author name: Einstein

Enter Price:1000

Enter Title:english

Enter Author name:willey

Enter Price:900

Enter Title:chemistry

Enter Author name:narayan

Enter Price:800

Enter Title:history

Enter Author name:george

Enter Price:700

Enter Title:geography

Enter Author name:bailey

Enter Price:600

Enter Title:AOA

Enter Author name:Koreman

Enter Price:500

Enter Title:SPA

Enter Author name:Harish

Enter Price:400

Enter Title:Maths

Enter Author name:kumbhojkar

Enter Price:300

Enter Title:CG

Enter Author name:hern

Enter Price:200

Enter Title:python

Enter Author name:naryan

Enter Price:100

Title	Author	price
Python	Narayan	100
CG	hern	200
Maths	kumbhojkar	300
SPA	Harish	400
AOA	koreman	500
Geography	bailey	600
History	George	700
Chemistry	Narayan	800
English	willey	900
Physics	Einstein	1000

Q.4.b. i) Explain gets() and puts() functions of C language. Comment on their parameters and return values.

Ans:

gets():

1. gets() function is used to scan a line of text from a standard input device.
2. This function will be terminated by a newline character.
3. The newline character won't be included as part of the string. The string may include white space characters.

   Syntax :

   char *gets(char *s);

4. This function is declared in the header file stdio.h.
5. It takes a single argument. The argument must be a data item representing a string. On successful completion, shall return a pointer to string s.

puts():

The C library function int puts(const char *str) writes a string to stdout up to but not including the null character. A newline character is appended to the output.

Example: int puts(char const*str)

Parameters: str-this is the C string to be written

Return value: If successful, non-negative value is returned. On error, the function returns EOF.

Q.4.b. ii) Enlist bitwise operator in c language. Explain any 2 with examples.

Ans.

The bitwise operators are listed below

1. ~ to perform bitwise NOT operation.
2. & to perform bitwise AND operation.
3. | to perform bitwise OR operation.
4. ^ to perform bitwise EXOR operation.
5. << to perform bitwise left shift operation.
6. >> to perform bitwise right shift operation.

AND operator
Example .

5 & 3 = 1

$(5)_{10}$ = $(0\ 1\ 0\ 1)_2$
$(3)_{10}$ = $(0\ 0\ 1\ 1)_2$

$(0\ 0\ 0\ 1)_2 = (1)_{10}$

OR operator

Example.

12 | 9 = 13

$(12)_{10} = (1\ 1\ 0\ 0)_2$
$(9)_{10}$ = $(1\ 0\ 0\ 1)_2$

$(1\ 1\ 0\ 1)_2 = (2)_{10}$

Q.5.a) Write a programs to print following patterns for N lines.

1.
```
5 4 3 2 *
5 4 3 * 1
5 4 * 2 1
5 * 3 2 1
* 4 3 2 1
```

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j;
    clrscr();
    for(i=1;i<=5;i++)
        {
        for(j=5;j>=1;j--)
            {
            if(i==j)
             printf("*");
            else
              printf("%d",j);
            }
        printf("\n");
         }
      getch();
    }
```

Output:

5 4 3 2 *

5 4 3 * 1

5 4 * 2 1

5 * 3 2 1

* 4 3 2 1

Q.5.b) 2.

5

5 4

5 4 3

5 4 3 2

5 4 3 2 1

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i,j;
 clrscr();
 for(i=5;i>=1;i--)
  {
      for(j=1;j<i;j++)
       printf(" ");
      for(j=5;j>=i;j--)
          printf("%d",j);
        printf("\n");
     }
      getch();
```

}

Output:

. 5

5 4

5 4 3

5 4 3 2

5 4 3 2 1

Q.5.b) Why files are needed? Explain all the opening modes. Write a program to create copy of the user specify names of source and destination files.

Ans. Need of files:

1. When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.

2. If you have to enter a large number of data, it will take a lot of time to enter them all. However if you have a file containing all the data, you can easily access the contents of the file using few commands in c.

3. You can easily move your data from one computer to another without any changes.
4. The various modes in which a file can be opened are as listed below:
   i. "r" indicates that the file is to be opened indicates in the read mode.
   ii. "w" indicates that the file is to be opened indicates in the write mode.

   When a file is opened in write mode the data written in the file overwrites the previously stored data in the file.

   iii. "a" indicates that the file is to be opened in the append mode.In this mode the data written into the file is appended towards the end of the already stored data in the file.The earlier stored data remains as it is.
   iv. "w+" indicates that the file is to be opened in write and read mode.
   v. "r+" indicates that the file is to be opened in read and write mode.

```
//program to  create a copy of file.

#include <stdio.h>
```

```
#include<conio.h>
#include <stdlib.h>

void main()
{
  char ch, source_file[20], target_file[20];
  FILE *source, *target;
 clrscr();
  printf("Enter name of file to copy\n");
  gets(source_file);

  source = fopen(source_file, "r");

  if( source == NULL )
  {
    printf("Press any key to exit...\n");
    exit(EXIT_FAILURE);
  }

  printf("Enter name of target file\n");
  gets(target_file);

  target = fopen(target_file, "w");

  if( target == NULL )
  {
    fclose(source);
    printf("Press any key to exit...\n");
    exit(EXIT_FAILURE);
```

```
}

while( ( ch = fgetc(source) ) != EOF )
  fputc(ch, target);

printf("File copied successfully.\n");

fclose(source);
fclose(target);

getch();
}
```

Output:

Enter name of file to copy

Factorial.c

Enter name of targt file

Factorial-copy.c

File copied successfully

Q.6.a) Comment on dynamic memory allocation. Write a program to read and store N integers in a Array, where value of N is defined by user. Find minimum and maximum members from the Array.

Ans. Dynamic memory allocation:

1. Dynamic Memory Allocation refers to managing system memory at runtime.
2. Dynamic memory management in C programming language is performed via a group four functions namely malloc(), calloc().
3. These four dynamic memory allocation function of the C programming language are defined in the C standard library header file <stdlib.h>. Dynamic memory allocation uses the heap space of the system memory.

malloc()

1. malloc() is used to allocate a block of memory on the heap.
2. Specifically, malloc() allocates the user a specified number of bytes but does not initialize.
3. Once allocated, the program accesses this block of memory via a pointer that malloc() returns.
4. The default pointer returned by malloc() is of the type void but can be cast into a pointer of any data type.
5. However, if the space is insufficient for the amount of memory requested by malloc().
6. Then the allocation fails and a NULL pointer is returned.
7. The function takes a single argument, which is the size of the memory chunk to be allocated.

```
//program to find maimum and minimum element from the array
  #include<stdio.h>
  #include<conio.h>
  #define MAX_SIZE 100
  void main()
 {
   int arr[MAX_SIZE];
   int i, max, min, size;
  clrscr();
  printf("Enter size of the array: ");
   scanf("%d", &size);
  printf("Enter elements in the array: ");
  for(i=0; i<size; i++)
  {
   scanf("%d", &arr[i]);
   }
 max = arr[0];
 min = arr[0];
 for(i=1; i<size; i++)
 {
```

```
        if(arr[i] > max)
        {
         max = arr[i];
         }
        if(arr[i] < min)
        {
         min = arr[i];
        }
      }
       printf("Maximum element = %d\n", max);
      printf("Minimum element = %d", min);
      getch();
}
```

Output:

```
Enter size of the array: 10
Enter elements in the array:
6
4
3
8
9
12
65
87
54
24
Maximum element=87
Minimum element=3
```

Q.6.b) Explain any 4 functions from string.h header file with suitable examples.

Ans:

Functions from string.h header file are as follows

strlen() function

1. This function returns an integer value that is the length of the string passed to the function.
2. When returning the length of the string it does not consider the space required for null Character.
3. Hence it returns the exact length of the string neglecting these space required for null character.

Example:

```
#include<conio.h>
#include <stdio.h>
#include<string.h>
 {
    Int l;
   char a[100];
   clrscr();
   printf("enter a string\n");
     gets(a);
    l=strlen(a);
    printf("the length of the entered string is: %d",l);
     getch();
 }
```

Output:

Enter a string

Hello

The length of the entered string is 5

2. strcpy() function

This function copies the second string into the first string passed to the function. The second string remains unchanged. Only the first string is changed and gets a copy of the second string.for e.g. strcpy(str1,str2), will copy the string “str2” into the string “str1”.

Example:

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
  char a[100],b[100];
   clrscr();
   printf(“enter a string\n”);
   gets(a);
  strcpy(b,a);
  printf(“the new string is %s”,b);
 getch();
 }
```

Output:

Enter a string

Hello, how are you?

The new string is Hello, how are you?

3.Strcmp() function

This function compares the two string variables passed to it.it returns an integer value equal to:

1. 0(zero),if the two strings are equal.
2. Negative value ,if the first string is smaller than the second string.
3. Positive value,if the first string is greater than the second string.

Both the strings remain unchanged.

The string is smaller means its alphabetical sequence is smaller. For

Example: ”Hello” is lesser than “Hi”; because the first character “H” is same, but the second character “e” is smaller than “I”. ”e” is smaller than “I” because “e” comes before “i” in the alphabets i.e. A,B,C, ……Z. the function compares the ASCII value of the characters.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    char    a[100],b[100];
    clrscr();
    printf(“enter two strings:\n”);
    gets(a);
    gets(b);
    if(strcmp(a,b)==0)
        printf(“strings are equal “);
    else
        printf(“%s string is greater “,a);
    else
        printf(“%s string is greater”,b);
    getch();
```

}

Output:

Enter two strings:

Hello

Hi

Hi string is greater

4. strcat() function

This function concatenates (joins) the two string variables passed to it. It returns a string of the combination of the two in the first string variable.

Example:

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
   char a[100],b[100];
   clrscr();
   printf("enter two strings:\n");
   gets(a);
   gets(b);
   strcat(a,b);
   printf("the concatenated strig is %s",a);
   getch();
}
```

Output:

Enter two strings:

Mumbai

University

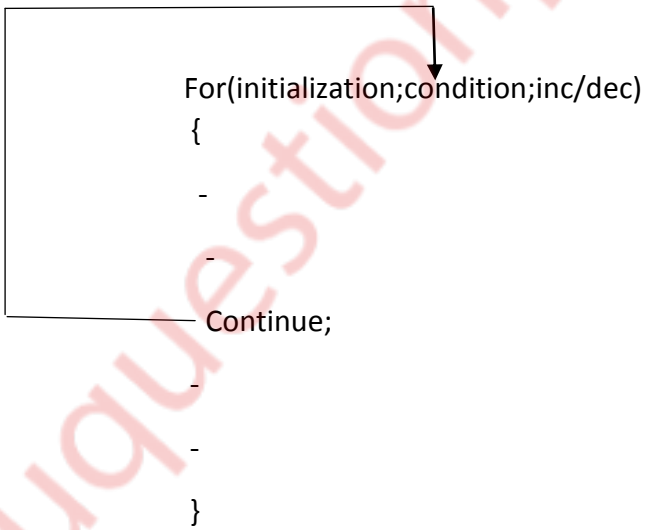
The concatenated string is MumbaiUniversity

Q.6.c) Explain continue and break statements with the help of suitable examples.

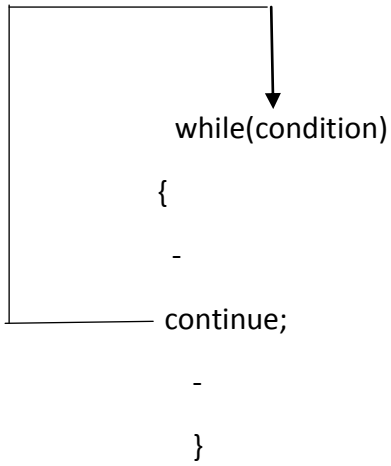
Ans.

1. continue statement-the continue statement also neglects the statements after it in the Loop and transfers the control back to the starting of the loop for next iteration.

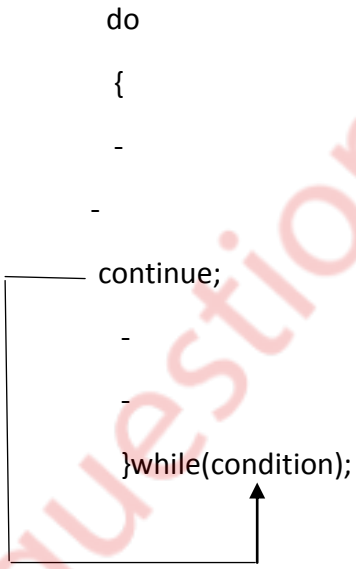
***operation of continue statement in a for loop.**



***operation of continue statement in a while loop**



***operation of continue statements in a do-while loop**

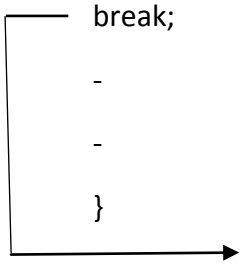


2.Break statement:

The break statement neglects the statement after it in the loop and transfers the control outside the loop

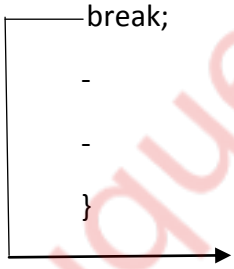
***operation of break statement in a for loop**

```
for(initialization;condition;inc/dec)
{
  -
-
break;
-
-
}
```

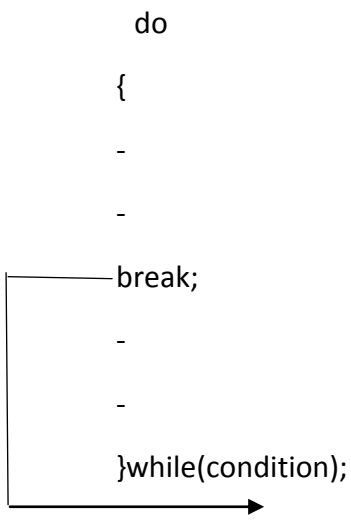


***operation of break statement in a while loop**

```
while(condition)
{
-
-
break;
-
-
}
```



***operation of break statement in a do-while loop**



Q. P. Code : 23993

Q1. a) Define union. Compare Structure and Union.
Ans.

Union :- A **union** is a special data type available in **C** that allows storing different data types in the same memory location.

	Structure	Union
Keyword	The keyword **'struct'** is used to define a structure.	The keyword **'union'** is used to define a union.
Size	When a variable associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members.	When a variable associated with a union, the compiler allocates the memory by considering the size of the largest memory. Hence, The size of union is equal to the size of largest members.
Memory	Each member within a structure is assigned and unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initializing Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

Q1. b) What is an error ? Explain different types of errors occurred in program.
Ans.

Error :-

While writing c programs, errors also known as bugs in the world of programming may occur unwillingly which may prevent the program to compile and run correctly as per the expectation of the programmer.

Types of errors :-

Basically there are three types of errors in c programming:

1. Runtime Errors :

C runtime errors are those errors that occur during the execution of a c program and generally occur due to some illegal operation performed

in the program. For example,

- Dividing a number by zero
- Trying to open a file which is not created
- Lack of free memory space

2. Compile Errors :-

Compile errors are those errors that occur at the time of compilation of the program. C compile errors may be further classified as:

2.1 Syntax Errors :

When the rules of the c programming language are not followed, the compiler will show syntax errors.

2.2 Semantic Errors :

Semantic errors are reported by the compiler when the statements written in the c program are not meaningful to the compiler.

3. Logical Errors :-

Logical errors are the errors in the output of the program. The presence of logical errors leads to undesired or incorrect output and are caused due to error in the logic applied in the program to produce the desired output. Also, logical errors could not be detected by the compiler, and thus, programmers have to check the entire coding of a c program line by line.

Q1. c) Explain switch case and if-else ladder with example.

Ans.

Switch Case :-

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

Example :

Code :

```
// Following is a simple program to demonstrate syntax of switch.
#include <stdio.h>
#include <conio.h>
int main()
{
  int x = 2;
```

```
switch (x)
 {
   case 1: printf("Choice is 1");
         break;
   case 2: printf("Choice is 2");
          break;
   case 3: printf("Choice is 3");
         break;
   default: printf("Choice other than 1, 2 and 3");
          break;
 }
 return 0;
}
```

Output :

```
Choice is 2
```

If-else ladder :-

The if else ladder statement in C programming language is used to test set of conditions in sequence. An if condition is tested only when all previous if conditions in if-else ladder is false. If any of the conditional expression evaluates to true, then it will execute the corresponding code block and exits whole if-else ladder.

Example :

Code :

```
#include<stdio.h>
#include<conio.h>
 void main()
{
     int a;
     printf(“Enter a number : “);
     scanf(“%d”,&a);
     if(a > 0)
     {
          printf(“\nThe number is positive “);
     }
     else if(a<0)
```

```
        {
                printf("\n The number is negative");
        }
    else
        {
                printf("Number is zero");
        }
     getch();
    }
```

Output :

```
Enter a number : 25
The number is positive.
```

Q1. d) Explain any four standard library functions from sting.h ?
Ans.

Standard Library Functions from string.h :-

In the C Programming Language, the Standard Library Functions are divided into several header files.

The following is a list of functions found within the <string.h> header file:

I. Comparison functions
   1. memcmp - Compare Memory Blocks
   2. strcmp - String Compare
   3. strcoll - String Compare Using Locale-Specific Collating sequence.
   4. strncmp - Bounded String Compare
   5. strxfrm - Transform Locale-Specific String

II. Concatenation functions
   1. strcat - String Concatenation
   2. strncat - Bounded String Concatenation

III. Copying functions
   1. memcpy - Copy Memory Block
   2. memmove - Copy Memory Block
   3. strcpy - String Copy
   4. strncpy - Bounded String Copy

IV. Search functions
   1. memchr - Search Memory Block for Character
   2. strchr - Search String for Character
   3. strcspn - Search String for Intial Span of Characters Not in Set

4. strpbrk - Search String for One of a Set of Characters
5. strrchr - Search String in Reverse for Character
6. strspn - Search String for Initial Span of Characters in Set
7. strstr - Search String for Substring
8. strtok - Search String for Token

V. Miscellaneous functions

1. memset - Initialize Memory Block
2. strerror - Convert Error Number to String
3. strlen - String Length

Q1. e) Explain break and continue statement with example.

Ans.

It is sometimes desirable to skip some statements inside the loop or terminate the loop immediately without checking the test expression. In such cases, break and continue statements are used.

break Statement :-

The break statement terminates the loop (for, while and do...while loop) immediately when it is encountered. The break statement is used with decision making statement such as if...else. In C programming, break statement is also used with switch...case statement.

Syntax of break statement :

```
break;
```

Example :-

Code :

```
// Program to calculate the sum of maximum of 10 numbers
// Calculates sum until user enters positive number
# include <stdio.h>
# include <conio.h>
int main()
{
int i;
double number, sum = 0.0;
for (i=1; i <= 10; ++i)
        {
printf("Enter a n%d: ",i);
```

```
scanf("%lf",&number);
// If user enters negative number, loop is terminated
if(number < 0.0)
        {
                break;
        }
        sum += number; // sum = sum + number;
 }
 printf("Sum = %.2lf",sum);
 return 0;
}
```

Output :

```
Enter a n1: 2.4
Enter a n2: 4.5
Enter a n3: 3.4
Enter a n4: -3
Sum = 10.30
```

continue Statement :-

The continue statement skips some statements inside the loop. The continue statement is used with decision making statement such as if...else.

Syntax of continue Statement :

```
continue;
```

Example :-

Code :

```
// Program to calculate sum of maximum of 10 numbers
// Negative numbers are skipped from calculation
# include <stdio.h>
# include <conio.h>
int main()
{
   int i;
   double number, sum = 0.0;
```

```
    for(i=1; i <= 10; ++i)
    {
        printf("Enter a n%d: ",i);
        scanf("%lf",&number);
        // If user enters negative number, loop is terminated
        if(number < 0.0)
        {
            continue;
        }
        sum += number; // sum = sum + number;
    }
    printf("Sum = %.2lf",sum);
    return 0;
}
```

Output :

```
Enter a n1: 1.1
Enter a n2: 2.2
Enter a n3: 5.5
Enter a n4: 4.4
Enter a n5: -3.4
Enter a n6: -45.5
Enter a n7: 34.5
Enter a n8: -4.2
Enter a n9: -1000
Enter a n10: 12
Sum = 59.70
```

Q2. a) Define Algorithm. Write Algorithm to check whether given number is Armstrong number or not also mention input and output specifications to algorithm.

Ans.

Algorithm :-

An Algorithm is a sequence of steps to solve a problem. Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.

Characteristics of Algorithm :-

1) ***Finiteness*** - An algorithm must always terminate after a finite number of steps.
2) ***Definiteness*** - Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case.
3) ***Input*** - An algorithm has zero or more inputs, i.e, quantities which are given to it initially before the algorithm begins.
4) ***Output*** - An algorithm has one or more outputs i.e, quantities which have a specified relation to the inputs.
5) ***Effectiveness*** - An algorithm is also generally expected to be effective. This means that all of the operations to be performed in the algorithm must be sufficiently basic that they can in principle be done exactly and in a finite length of time.

Algorithm to check whether given number is Armstrong or not :-

Step I : Start.
Step II : Input n, sum, rem, temp.
Step III : sum = 0, rem=0.
Step IV : Print "Enter an integer number : "
Step V : Read n.
Step VI : temp = n.
Step VII : If temp is less than equal to zero Then,
Go to *Step IX*.
Else
rem = temp mod 10
sum = sum + (rem X rem X rem)
temp = temp / 10
Step VIII : Go to *Step VII*.
Step IX : If sum is equal to n Then,
Print "Number n is an Armstrong number."
Else
Print "Number n is not an Armstrong number."
Step X : Stop

Q2. b) Explain various storage classes with example.
Ans.

Storage Classes :-

Storage Classes are used to describe about the features of a variable/function. These features basically include the scope, visibility and life-time which help us to trace the existence of a particular variable during the runtime of a program. To specify the storage class for a variable, the following syntax is to be followed:

Syntax:

storage_class var_data_type var_name;

C language uses 4 storage classes, namely:

i. Automatic storage class :

This is the default storage class for all the variables declared inside a function or a block. Hence, the keyword auto is rarely used while writing programs in C language. Auto variables can be only accessed within the block/function they have been declared and not outside them (which defines their scope). Of course, these can be accessed within nested blocks within the parent block/function in which the auto variable was declared. However, they can be accessed outside their scope as well using the concept of pointers given here by pointing to the very exact memory location where the variables resides. They are assigned a garbage value by default whenever they are declared.

ii. External storage class :

Extern storage class simply tells us that the variable is defined elsewhere and not within the same block where it is used. Basically, the value is assigned to it in a different block and this can be overwritten / changed in a different block as well. So an extern variable is nothing but a global variable initialized with a legal value where it is declared in order to be used elsewhere. It can be accessed within any function/block. Also, a normal global variable can be made extern as well by placing the 'extern' keyword before its declaration/definition in any function/block. This basically signifies that we are not initializing a new variable but instead we are using/accessing the global variable only. The main purpose of using

extern variables is that they can be accessed between two different files which are part of a large program. For more information on how extern variables work, have a look at this link.

iii. Static storage class :

This storage class is used to declare static variables which are popularly used while writing programs in C language. Static variables have a property of preserving their value even after they are out of their scope! Hence, static variables preserve the value of their last use in their scope. So we can say that they are initialized only once and exist till the termination of the program. Thus, no new memory is allocated because they are not re-declared. Their scope is local to the function to which they were defined. Global static variables can be accessed anywhere in the program. By default, they are assigned the value 0 by the compiler.

iv. Register storage class :

This storage class declares register variables which have the same functionality as that of the auto variables. The only difference is that the compiler tries to store these variables in the register of the microprocessor if a free register is available. This makes the use of register variables to be much faster than that of the variables stored in the memory during the runtime of the program. If a free register is not available, these are then stored in the memory only. Usually few variables which are to be accessed very frequently in a program are declared with the register keyword which improves the running time of the program. An important and interesting point to be noted here is that we cannot obtain the address of a register variable using pointers.

Example :-

Code :

```
// A C program to demonstrate different storage classes
#include <stdio.h>
 #include <conio.h>
extern int x = 9;
int z = 10;
int main()
{
```

```
    auto int a = 32;
    register char b = 'G';
    extern int z;
    printf("Hello World!\n");
    printf("\nThis is the value of the auto " " integer 'a': %d\n",a);
    printf("\nThese are the values of the" " extern integers 'x' and 'z'" "
                respectively: %d and %d\n", x, z);
    printf("\nThis is the value of the " "register character 'b': %c\n",b);
    x = 2;
    z = 5;
    printf("\nThese are the modified values " "of the extern integers 'x'
                and " "'z' respectively: %d and %d\n",x,z);
    printf("\n'y' is a static variable and its " "value is NOT initialized to
                5 after" " the first iteration! See for" " yourself :)\n");
    while (x > 0)
    {
       static int y = 5;
       y++;
       printf("The value of y is %d\n",y);
       x--;
    }
    printf("\nBye! See you soon.\n");
    return 0;
}
```

Output :

```
Hello World!
This is the value of the auto  integer 'a': 32
These are the values of the extern integers 'x' and 'z'
respectively: 9 and 10
This is the value of the register character 'b': G
These are the modified values of the extern integers 'x'
and 'z' respectively: 2 and 5
'y' is a static variable and its value is NOT initialized
to 5 after the first iteration! See for yourself :)
The value of y is 6
The value of y is 7
Bye! See you soon.
```

Q3. a) Explain Nested Structure. Write a program using nested structure to create an Array of structure to store the details of N students.
The details are,

1. **Student name**
2. **Student roll no**
3. **Marks of Physics, Chemistry, Maths.**

Calculate total of P-C-M. Display the data in the format
Name Roll no Total marks
Ans.

Nested structure :-

Nested structure in C is nothing but structure within structure. One structure can be declared inside other structure as we declare structure members inside a structure. The structure variables can be a normal structure variable or a pointer variable to access the data. Nested structure in c language can have another structure as a member. There are two ways to define nested structure in c language:

1) Separate structure :

We can create 2 structures, but dependent structure should be used inside the main structure as a member. Let's see the code of nested structure.

2) Embedded structure :

We can define structure within the structure also. It requires less code than previous way. But it can't be used in many structures.

The syntax of nested structure is given as :

```
struct structure_name
{
data_type variable_name;
_ _ _ _ _ _ _
struct
{
data_type variable_name;
_ _ _ _ _ _ _
internal_structure_name;
_ _ _ _ _ _ _ _
}
```

Solution Program :-

Sourse Code :

```
#include <stdio.h>
#include <conio.h>
struct students
{
        char name[30] ;
        int roll_no, total;
        struct
        {
                int physics, chemistry, maths;
        }
        marks;
};
void main( )
{
        struct students n[100];
        int n, i, j;
        clrscr( );
        printf("Enter number of students : ");
        scanf("%d", &n);
        for (i=0 ; i<=n-1 ; i++)
        {
                printf("Enter following details of student :- ");
                printf("Name : ");
                scanf("%s", & n[i].name);
                printf("Roll number : ");
                scanf("%d", & n[i].roll_no);
                printf("Marks in Physics : ");
                scanf("%d", & n[i].marks.physics);
                printf("Marks in Chemistry : ");
                scanf("%d", & n[i].marks.chemistry);
                printf("Marks in Mathematics : ");
                scanf("%d", & n[i].marks.maths);
                n[i].total = n[i].marks.physics + n[i].marks.chemistry +
                                n[i].marks.maths;
        }
        printf("\n Name \t Roll Number \t Total \n");
```

```
        printf(" _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _\n");
        for (i=0 ; i<=n-1 ; i++)
        {
            printf("%s \t %d \t %d \n", n[i].name, n[i].roll_no,
                n[i].total);
        }
getch();
}
```

Output :-

```
Enter number of students : 3
Enter following details of student :-
Name : Saurav Palande
Roll number : 45
Marks in Physics : 50
Marks in Chemistry : 60
Marks in Mathematics : 30
Enter following details of student :-
Name : Sayali Rajapurkar
Roll number : 40
Marks in Physics : 50
Marks in Chemistry : 60
Marks in Mathematics : 40
Enter following details of student :-
Name : Amrit Shah
Roll number : 65
Marks in Physics : 60
Marks in Chemistry : 60
Marks in Mathematics : 60

Name                Roll Number          Total
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Saurav Palande      45                   140
Sayali Rajapurkar   40                   150
Amrit Shah          65                   180
```

Q3. b) Define pointer and its use. Explain array of pointer with example. Write program to swap the values by using call by reference concept.
Ans.

Pointer :-

A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address. The general form of a pointer variable declaration is −

type *var-name;

Uses of pointer :-

1. Pointers reduce the length and complexity of a program.
2. They increase execution speed.
3. A pointer enables us to access a variable that is defined outside the function.
4. Pointers are more efficient in handling the data tables.
5. The use of a pointer array of character strings results in saving of data storage space in memory.

Array of Pointers :-

Just like we can declare an array of int, float or char etc, we can also declare an array of pointers, here is the syntax to do the same.

Syntax :

datatype *array_name[size];

Example :

int *arrop[5];

Here arrop is an array of 5 integer pointers. It means that this array can hold the address of 5 integer variables, or in other words, you can assign 5 pointer variables of type pointer to int to the elements of this array. arrop[i] gives the address of i th element of the array. So arrop[0] returns address of variable at position 0, arrop[1] returns address of variable at position 1 and so on. To get the value at address use indirection operator (*). So *arrop[0] gives value at address[0], Similarly *arrop[1] gives the value at address arrop[1] and so on.

Solution Program :-

Source Code :

```
//Program to swap the values by using call by reference concept.
#include <stdio.h>
#include <conio.h>
void swap( int *x, int *y )
{
        int t ;
        t = *x ;
        *x = *y ;
        *y = t ;
        printf("In function :");
        printf( "\nx = %d \t y = %d\n", *x,*y);
}
void main( )
{
        int a, b;
        printf("Enter the value of a : ");
        scanf("%d", &a);
        printf("Enter the value of b : ");
        scanf("%d", &b);
        printf("Before swapping : \n");
        printf ( "\na = %d \t b = %d\n", a, b ) ;
        swap ( &a, &b ) ;
        printf("After swapping : \n");
        printf ( "\na = %d \t b = %d\n", a, b ) ;
        getch();
}
```

Output :

```
Enter the value of a : 10
Enter the value of b : 20
Before swapping :
a=10        b=20
In function :
x=20        y=10
After swapping :
a=20        b=10
```

Q4. a) Explain recursive function. Write a program to find the GCD of a number by using recursive function.

Ans.

Recursive function :-

In C, a function can call itself. This process is known as recursion. And a function that calls itself is called as the recursive function. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function. The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

Program :-

Source Code :

```
//Program to find GCD of a number by using recursive function.
#include <stdio.h>
#include <conio.h>
int gcd(int n1, int n2);
int main( )
{
        int n1, n2;
        printf("Enter two positive integers : \n");
        scanf("%d %d", &n1, &n2);
        g = gcd(n1,n2);
        printf("G.C.D of %d and %d is %d.", n1, n2, g);
        return 0;
}
int gcd(int n1, int n2)
{
         while (n1 != n2)
        {
                if (n1 > n2)
                        return gcd(n1 – n2, n2);
                else
                        return gcd(n1, n2 – n1);
        }
 return a;
 }
```

Output :

```
Enter two positive integers:
366
60
G.C.D of 366 and 60 is 6.
```

Q4. b) Write a program to perform matrix multiplication by passing input matrix to the function and printing resultant matrix.

Ans.

Program :-

Sourse code :

```
//Program for matrix multiplication using functions.
#include <stdio.h>
#include <stdlib.h>
void input(int m, int n, int a[m][n])
{
   for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
         printf("%d, %d : ", i, j);
         scanf("%d", &a[i][j]);
      }
   }
}

void print(int m, int n, int a[m][n])
{
   int i, j;
   for (i = 0; i < m; i++) {
      for (j = 0; j < n; j++) {
         printf("%3d ", a[i][j]);
      }
      printf("\n");
   }
}
void multiply(int m, int n, int p, int a[m][n], int b[n][p], int c[m][p])
```

```
{
   for (int i = 0; i < m; i++) {
      for (int j = 0; j < p; j++) {
         c[i][j] = 0;
         for (int k = 0; k < n; k++) {
            c[i][j] += a[i][k] * b[k][j];
         }
      }
   }
}
void main()
{
   int r1, c1, r2, c2;
   printf("Row and column for matrix #1 :\n");
   scanf("%d %d", &r1, &c1);
   printf("Row and column for matrix #2 :\n");
   scanf("%d %d", &r2, &c2);
   if (r2 != c1) {
      printf("The matrices are incompatible.\n");
      exit(EXIT_FAILURE);
   }
   int mat1[r1][c1], mat2[r2][c2], ans[r1][c2];
   printf("Enter elements of the first matrix.\n");
   input(r1, c1, mat1);
   printf("The elements of the first matrix are :\n");
   print(r1, c1, mat1);
   printf("Enter elements of the second matrix.\n");
   input(r2, c2, mat2);
   printf("The elements of the second matrix are :\n");
   print(r2, c2, mat2);
   multiply(r1, r2, c2, mat1, mat2, ans);
   printf("The product is :\n");
   print(r1, c2, ans);
   getch( );
}
```

Output :

```
Row and column for matrix #1 :
2
2

Row and column for matrix #2 :
2
2

Enter elements of the first matrix.
0, 0 : 1
0, 1 : 17
1, 0 : 25
1, 1 : 30

The elements of the first matrix are :

1  17
25  30

Enter elements of the second matrix.
0, 0 : 74
0, 1 : 89
1, 0 : 3
1, 1 : 65

The elements of the second matrix are :

74  89
3  65

The product is :

125 1194
1940 4175
```

Q5. a) Write a program to display following pattern:

1
232
34543
4567654
567898765

Ans.

Program :-

Source code :

```
#include <stdio.h>
#include <conio.h>
void main( )
{
        int i, s, r, k=0, c=0, j=0;
        printf("Enter the number of rows : ");
        scanf("%d", &r);
        for( i=1 ; i<=r ; i++ )
        {
                for( s=1 ; s<=r-i ; s++ )
                {
                        prinf(" ");
                        c++;
                }
                while( k != 2*i-1 )
                {
                        if(c<=r-1)
                        {
                                printf("%d",(i+k));
                                c++;
                        }
                        else
                        {
                                j++;
                                printf("%d",(i+k-2*j));
                        }
                        ++k;
                }
```

```
                j=c=k=0;
                printf("\n");
                }
        getch( );
}
```

Output :

```
Enter the number of rows : 5
      1
     232
    34543
   4567654
567898765
```

Q5. b) Write user defined function to implement string concatenation.
Ans.

Program :-

Source code :

```
#include<stdio.h>
#include<string.h>
void concat(char[], char[]);
int main() {
  char s1[50], s2[50];
  printf("\nEnter String 1 :");
  gets(s1);
  printf("\nEnter String 2 :");
  gets(s2);
  concat(s1, s2);
  printf("\nConcated string is :%s", s1);
  return (0);
}
void concat(char s1[], char s2[])
{
  int i, j;
  i = strlen(s1);
```

```
    for (j = 0; s2[j] != '\0'; i++, j++)
        {
            s1[i] = s2[j];
        }

    s1[i] = '\0';
}
```

Output :

```
Enter String 1 : Prathamesh
Enter String 2 : Padave
Concated string is : PrathameshPadave
```

Q5. c) Explain need of file data and various modes of files also write program to create edit copy of file.

Ans.

File Handling :-

File handling is an important part for any language. File handling helps you to store data (any type of data) in a controlled manner like for cookies, for any type of configurations etc. A file represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. The fopen() function is used to create a new file or to open an existing file.

General Syntax:

*fp = FILE *fopen(const char *filename, const char *mode);

Here, *fp is the FILE pointer (FILE *fp), which will hold the reference to the opened(or created) file. Filename is the name of the file to be opened and mode specifies the purpose of opening the file. Mode can be of following types,

- ✓ r - opens a text file in reading mode
- ✓ w - opens or create a text file in writing mode.
- ✓ a - opens a text file in append mode
- ✓ r+ - opens a text file in both reading and writing mode
- ✓ w+ - opens a text file in both reading and writing mode

- ✓ a+ - opens a text file in both reading and writing mode
- ✓ rb - opens a binary file in reading mode
- ✓ wb - opens or create a binary file in writing mode
- ✓ ab - opens a binary file in append mode
- ✓ rb+ - opens a binary file in both reading and writing mode
- ✓ wb+ - opens a binary file in both reading and writing mode
- ✓ ab+ - opens a binary file in both reading and writing mode

Program :-

Source code :

```
/* Program to create edit copy of file */
# include <stdio.h>
#include <conio.h>
void main( )
{
        FILE *fp ;
        char sub[50], temp[20];
        int s, t;
        clrscr( );
        fp=fopen("Exam.txt","w+");
        printf( "Enter seat number and subject of the candidate : " ) ;
        scanf("%d %s", &s, &sub);
        printf(" Seat No \t Subject \n");
        rewind(fp);
        fscanf(fp,"%s %d", &temp, &t);
        printf("%s %d", temp, t);
        fclose(fp) ;
        return 0;
}
```

Output :-

```
Enter Seat number and Subject of the student : 2001
Chemistry
Seat number          Subject
   2001              Chemistry
```

Q6. a) Write a program to sort given array in ascending order.
Ans.

Program :-

Source code :

```
//Program to sort given array in ascending order
#include <stdio.h>
#include <conio.h>
void main()
{
        int i, j, a, n, array[50];
        printf("How many numbers in an array…? \n");
        scanf("%d", &n);
        printf("Enter the numbers. \n");
        for (i = 0; i < n; i++)
                scanf("%d", &array[i]);
        for (i = 0; i < n; i++)
        {
                for (j = i + 1; j < n; j++)
                        {
                                if (number[i] > number[j])
                                {
                                        a =  number[i];
                                        number[i] = number[j];
                                        number[j] = a;
                                }
                        }
        }
        printf("The numbers arranged in ascending order are given
                below \n");
        for (i = 0; i < n; ++i)
                printf("%d\n", number[i]);
        getch( );
}
```

Output :

```
How many numbers in an array…?
5
Enter the numbers.
3
1
456
200
150
The numbers arranged in ascending order are given below
1
3
150
200
456
```

Q6. b) Write a program for finding sum of series 1+2+3+4+…….upto n terms.
Ans.

Program :-
Source code :

```
//program to calculate sum of series up to n terms 1 + 2 + 3 +…+n.
#include <stdio.h>
#include <conio.h>
void main( )

{
    int n,i;
    int sum=0;
    printf("Enter the n i.e. max value of series: ");
    scanf("%d",&n);
    sum = (n * (n + 1)) / 2;
    printf("Sum of the series: ");
    for (i =1 ; i <= n ; i++)
    {
        if (i!=n)
```

```
                printf("%d + ",i);
            else
                printf("%d = %d ",i,sum);
        }
        getch( );
}
```

Output :-

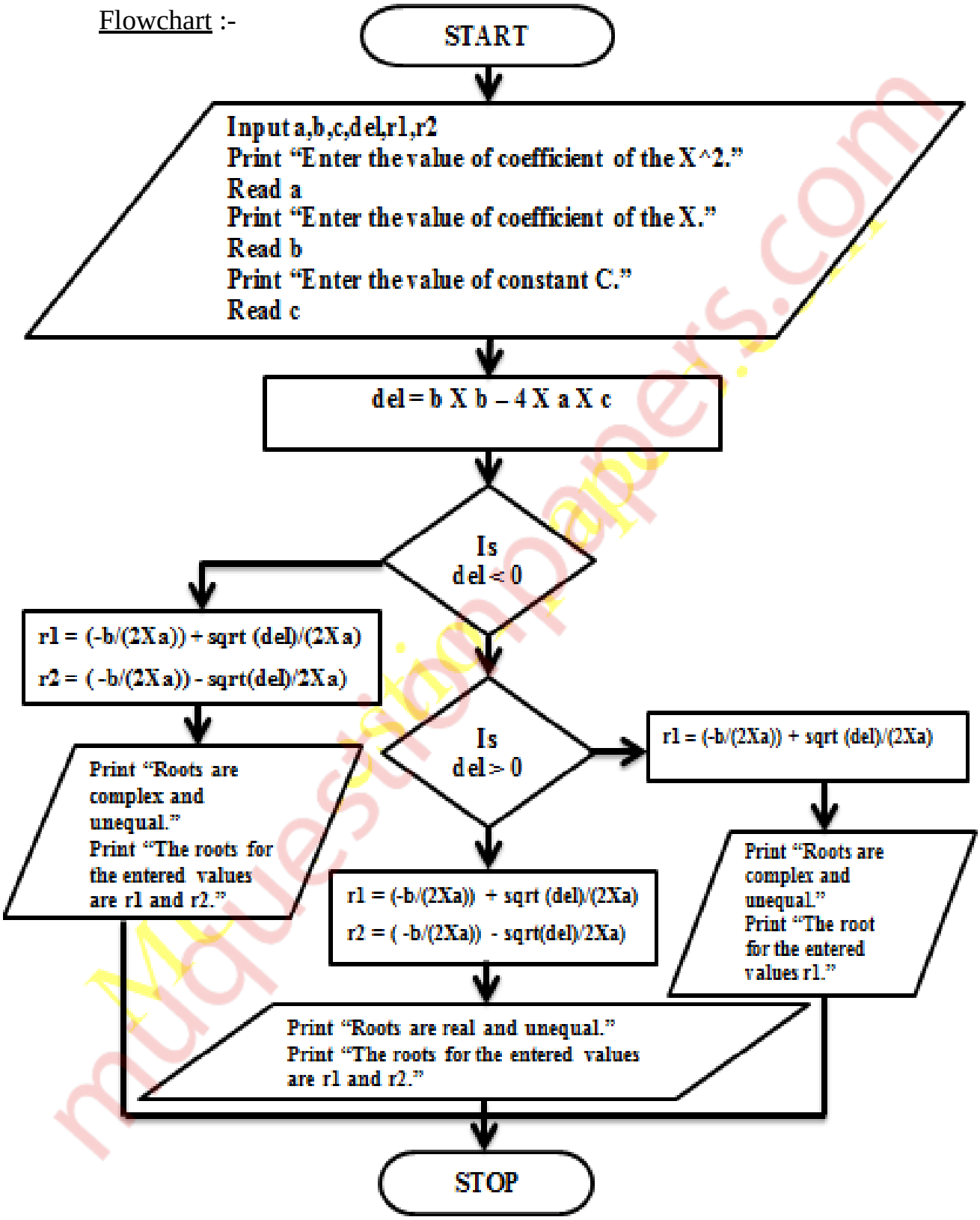
```
Enter the n i.e. max value of series: 5
Sum of the series: 1 + 2 + 3 + 4 + 5 = 15
```

Q6. c) Draw the flow chart to find roots of a quadratic equation.
Ans.

Algorithm :-

Step I : Start.
Step II : Input a,b,c,del,r1,r2.
Step III : Print "Enter the value of coefficient of the X^2."
Step IV : Read a.
Step V : Print "Enter the value of coefficient of the X."
Step VI : Read b.
Step VII : Print "Enter the value of constant C."
Step VIII : Read c.
Step IX : del = b X b – 4 X a X c
Step X : If del is less than zero Then,
r1 = (-b / (2 X a)) + sqrt (del) / 2 X a
r2 = (-b / (2 X a)) - sqrt (del) / 2 X a
Print "Roots are complex and unequal."
Print "The roots for the entered values are r1 and r2."
Else - if del is greater than zero Then,
r1 = (-b / (2 X a)) + sqrt (del) / 2 X a
r2 = (-b / (2 X a)) - sqrt (del) / 2 X a
Print "Roots are real and unequal."
Print "The roots for the entered values are r1 and r2."
Else,
r1 = (-b / (2 X a)) + sqrt (del) / 2 X a
Print "Roots are real and equal."
Print "The root for the entered values is r1."
Step XI : Stop

Flowchart :-



F.E. Sem - 2 CBCGS SPA – MAY 18

Q.1.a) Select the correct option from multiple choice questions. (10 M)

i. **Which bitwise operator is used to multiply the number by 2^n where n is number of bits.**
A]Bitwise-OR B]Bitwise-AND C]Bitwise Left shift D] Bitwise Right shift
Ans:- C

ii. **Which operator has the lowest priority?**
A] ++ b] % C] + D] ||
Ans:-D

iii. **Which of these is a valid variable declaration?**
A]in temp salary; B] float marks_student; C] float roll-no; D] int main;
Ans:-B

iv. **What will be the output of the following program?**

```
Void main() {
Double x=28;
Int r;
r=x%5;
printf("\n r=%d",r);}
```

A] r= 3 B] Run time Error C] Compile time Error D] None of the above
Ans:-A

v. **What will be the output of following program**

```
Void main() {
 int x[]={10,20,30,40,50};
printf("\n%d%d%d%d",x[4], 3[x], x[2], 1[x], x[0]); }
```

A]Error B}10 20 30 40 50 C] 50 40 30 20 10 D] None of these
Ans:-D

vi. **Which of the following is not a keyword of 'C'?**
A]auto B]register C]int D]function
Ans:-D

vii. **What will be the output?**

```
Void main() {
Int y;
y=0x10+010+10;
printf("\ny=%x",y); }
```

A] y=34 B] x=34 C] y=22 D] Error

Ans:C

viii. **Study the following C program**

```
Void main() {
int a=0;
for( ;a;);
a++; }
```

what will be the value of the variable a, on the execution of above program

A] 1 B] 0 C] -1 D] None of these

Ans:A

ix. **Which of the following is used as a string termination character?**

A] 0 B]\0 C] /0 D]None of these

Ans:B

x.

```
void main() {
char a[]= "Hello world";
char *p;
p=a;
printf("\n%d%d%d%d",sizeof(a),sizeof(p),strlen(a),strlen(p) ); }
```

A] 11 11 10 10 B] 10 10 10 10 C] 12 12 11 11 D] 12 2 11 11

Ans:D

Q.1.b) State True or False with reason. (10 M)

i. Size of pointer variable is equal to the datatype it points to. True

ii. A float constant cannot be used as a case constant in a switch statement. True

iii. The statement void p; is valid. True

iv. while(0); is an infinite loop. False

Ans: when the condition is while(0); the control will never enter into the loop hence there will not be any infinite loop.

v. scanf() function is used to input string having multiple words. True

vi. A function can have any number of return statements. True

vii. In a union, space is allocated to every member individually. False

Ans: A union shares space among its member.

viii. An algorithm is a graphical representation of the logic of a program. False

Ans: An algorithm is set of structured instruction used to execute the code.

ix. Comments in the program make debugging of the program easier. True

x.There is no difference between '\0' and '0'. False

Ans: '\0' is used as a string terminator whereas '0' is number for the interpreter.

Q.2.a.i) How to create array of structure variable and assign values to its members? (5 M)

Ans:

1. An array of structure can be declared in the same way as declaring array of any other data type.
2. For example, an array of the structure student can be declared as shown:
   struct student s[100];
   This array can now store information of 100 students.
3. Array of structure must be used when many variables of a structure are required.
4. Syntax:

```
Struct structure_name
{
    data_type member1;
    data_type member2;
    -
    -
    data_type member;
};
```

Q.2.a.ii) Differentiate between struct and union. When is union preferred over struct? Give on example of each. (5 M)

Ans:

Sr. NO.	Structure	Union
1.	Memory allotted to structure is equal to the space require collectively by all the members of the structure.	Memory allotted for a union is equal to the space required by the largest memory of that union
2.	Data is more secured in structure.	Data can be corrupted in a union.
3.	Structure provide ease of programming.	Unions are comparatively difficult for programming.
4.	Structures require more memory.	Unions require less memory.
5.	Structure must be used when information of all the member elements of a structure are to be stored.	Unions must be used when only one of the member elements of the union is to be stored.

1. Union is preferred over struct when only one of the member elements of the union is stored.
   Example of structure:

```
#include<conio.h>
#include<stdio.h>
Struct student
{
    char name[20];
    int roll_no;
    float fees;
};
void main()
{
    Struct student s1;
    clrscr();
    printf("Enter the student's name, roll number ");
    gets(s1.name);
    scanf("%d",&s1.roll_no);
    printf("The student details are as follows: \nName:%s\n
Roll number:%d",s1.name,s1.roll_no);
    getch();
```

```
}
```

Output:

```
Enter the student's name, roll number:John
20
The student details are as follows:
Name:John
Roll number:20
```

Example of union:

```
#include<stdio.h>
#include<conio.h>
Union info
{
    char name[20];
};
void main()
{
 Union info i1;
int choice;
clrscr();
printf("Enter your name");
scanf("%s",i1.name);
printf("Your name is %s",i1.name);
getch();
}
```

Output:

```
Enter your name:John
Your name is John
```

Q.2.b.i) WAP to print the sum of following series. (5 M)

$1+2^2+3^3+\ldots\ldots+n^n$

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n,sum=0;
 clrscr();
 printf("Enter a number ");
 scanf("%d",&n);
 while(n!=0)
 {
  sum=sum+(n*n);
  n--;
 }
 printf("The sum of the series is %d",sum);
 getch ();
}
```

Output:

```
Enter a number 3
The sum of the series is 14
```

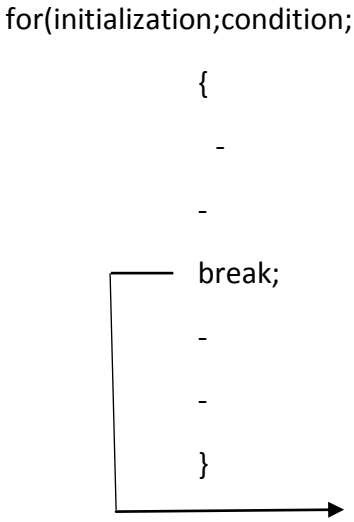
Q.2.b.ii) Compare the following **(5 M)**

i. break and continue statements

Ans:

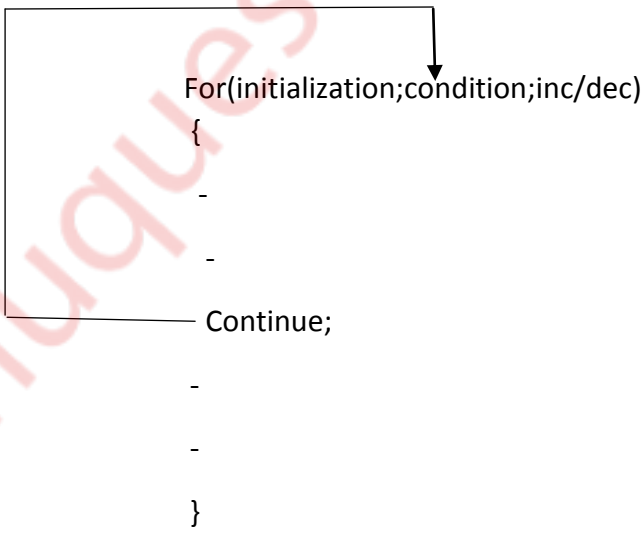
break statement: The break statement neglects the statement after it in the loop and transfer the control outside the loop.

Operation of break statement in for loop:



Continue statement: The continue statement also neglects the statement after it in the loop and transfer the control back to the starting of the loop for next iteration.

Operation of continue statement in for loop:



ii. if-else and switch statements:

Ans:

Sr.no	If……else	Switch
1.	In this we can test only one condition.	In this we can test multiple condition.
2.	It can have values based on constraints.	It can have values based on user choice.
3.	In this we cannot have different conditions.	In this case we can have only one expression but various value of the same expression.
4.	If else statement is used to evaluate a condition to be true or false	A switch case statement is used to test for multiple values of the same variable or expression like 1,2,3 etc.

Q.3.a) Write a program to calculate number of vowels (a, e, i, o, u) separately in the entered string. (6 M)

Ans:

```
#include <stdio.h>
#include<conio.h>

void main()
{
 int c = 0, count = 0,a=0,i=0,e=0,o=0,u=0,A=0,I=0,E=0,O=0,U=0;
 char s[1000];
 clrscr();

 printf("Input a string\n");
 gets(s);

 while (s[c] != '\0')
 {
  if (s[c] == 'a')
   {
```

```
 a++;
 printf("\na=%d",a);
}
else  if (s[c] == 'A')
{
 A++;
 printf("\nA=%d",A);
}
else  if (s[c] == 'e')
{
 e++;
 printf("\ne=%d",e);
}
else  if (s[c] == 'E')
{
 E++;
 printf("\nE=%d",E);
}
else  if (s[c] == 'i')
{
 i++;
 printf("\ni=%d",i);
}
else  if (s[c] == 'I')
{
 I++;
 printf("\nI=%d",I);
}
else  if (s[c] == 'o')
```

```
    {
     o++;
     printf("\no=%d",o);
    }
    else if (s[c] == 'O')
    {
     O++;
     printf("\nO=%d",O);
    }
    else  if (s[c] == 'u')
    {
     u++;
     printf("\nu=%d",u);
    }
    else  if (s[c] == 'U')
    {
     U++;
     printf("\nU=%d",U);
    }

    count++;
   c++;
  }
  getch();
}
```

Output:

Input string University

U=1

i=2

e=1

Q.3.b) Predict output of following program segment. (4 M)

i.

```
main()
{
int a,b,*p1,*p2,x,y;
 clrscr();
 a=48;b=10;p1=&a;p2=&b;
 x=*p1**p2-8;
 *p1=*p1+*p2;
 y=(*p1/ *p2)+20;
 printf("%d%d%d%d%d%d",*p1,*p2,a,b,x,y);
}
```

Output:

5810581047225

ii)

```
main()
{
 int x=4, y=9,z;
 clrscr();
 z=x++ + --y+y;
 printf("\n%d%d%d",x,y,z);
```

```
z= --x+x+y--;
printf("\n%d%d%d",x,y,z);
getch();
}
```

Output:

5820

4716

Q.3.c) An electronic component vendor supplies three products: transistors, resistors and capacitors. The vendor gives a discount of 10% on order of transistor if the order is more than Rs. 1000. On order of more than Rs. 100 for resistors, a discount of 5% is given and discount of 10% is given on order for capacitors of value more than Rs. 500. Assume numeric code 1, 2 and 3 used for transistors, capacitors and resistors respectively. Write a program that reads the product code and order amount and prints out the net amount that the customer is required to pay after discount. (Note: Use switch-case) (10 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int choice,n,dis;
 clrscr();
 printf("\n1.Transistor\n2.Capacitor\n3.Resistor\nEnter your choice");
 scanf("%d",&choice);
 printf("Enter the price");
 scanf("%d",&n);
 switch(choice)
 {
  case 1:if(n>1000)
            {
             dis=n/10;
```

```
        dis=n-dis;
        printf("The total price after discount is %d",dis);
       }
       else
       {
        dis=n;
        printf("The total price is %d",dis);
       }
       break;
case 2:if(n>500)
       {
        dis=n/10;
        dis=n-dis;
        printf("The total price after discount is %d",dis);
       }
       else
       {
        dis=n;
        printf("The total price is %d",dis);
       }
       break;
case 3:if(n>100)
       {
        dis=n/5;
        dis=n-dis;
        printf("The total price after discount is %d",dis);
       }
       else
       {
```

```
            dis=n;
            printf("The total price is %d",dis);
           }
           break;
 }
getch();
}
```

Output:

1.Transistor

2.Capacitors

3.Resistor

Enter your choice 1

Enter the price 1050

The total price after discount is 945

Q.4.a) What is recursion? WAP using recursion to find the sum of array elements of size n.

(10 M)

Ans:

1. Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.
2. A recursive function must definitely have a condition that exits from calling the function again.
3. Hence there must be a condition that calls the function itself if that condition is true.
4. If the condition is false then it will exit from the loop of calling itself again.

Program:

```
#include<conio.h>
#include <stdio.h>
#define MAX_SIZE 100

int sum(int arr[], int start, int len);
```

```
void main()
{
    int arr[MAX_SIZE];
    int N, i, sumof_array;
    clrscr();
    printf("Enter size of the array: ");
    scanf("%d", &N);
    printf("Enter elements in the array: ");
    for(i=0; i<N; i++)
    {
     scanf("%d", &arr[i]);
    }


    sumof_array = sum(arr, 0, N);
    printf("Sum of array elements: %d", sumof_array);

    getch();
}
int sum(int arr[], int start, int len)
{
    if(start >= len)
     return 0;

    return (arr[start] + sum(arr, start + 1, len));
}
```

Output:

Enter size of the array: 10

Enter elements in the array: 1 2 3 4 5 6 7 8 9 10

Sum of array Elements: 55

Q.4.b) Write a C program to **(10 M)**

i. **Create a 2D array [Matrix] [in main function]**
ii. **Write a function to read 2D array[Matrix]**
iii. **Write a function that will return true(1) if entered matrix is symmetric or false(0) is not symmetric.**
iv. **Print whether entered matrix is symmetric or not [in main function]**

Ans:

Program:

```
#include<stdio.h>

void main()
{
  int m, n, c, d, matrix[10][10], transpose[10][10];
  clrscr();
  printf("Enter the number of rows and columns of matrix\n");
  scanf("%d%d", &m, &n);
  printf("Enter elements of the matrix\n");
  for (c = 0; c < m; c++)
   for (d = 0; d < n; d++)
     scanf("%d", &matrix[c][d]);
  for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
      transpose[d][c] = matrix[c][d];
  if (m == n)
```

```
 {
   for (c = 0; c < m; c++)
   {
     for (d = 0; d < m; d++)
     {
          if (matrix[c][d] != transpose[c][d])
            break;
     }
     if (d != m)
          break;
   }
   if (c == m)
     printf("The matrix is symmetric.\n");
   else
     printf("The matrix isn't symmetric.\n");
 }
 else
   printf("The matrix isn't symmetric.\n");

 getch();
}
```

Output:

Enter the number of rows and columns of matrix

2 2

Enter elements of matrix

1 2 3 4

The matrix isn’t symmetric.

Q.5.a) implement string copy function STRCOPY (str1,str2) that copies a string str1 (source) to another string str2 (destination) without using library function. (5 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 char s1[100],s2[100],i;
clrscr();
 printf("enter string s1: ");
scanf("%s",s1);
for(i=0;s1[i]!='\0';++i)
 {
        s2[i]=s1[i];
 }
  s2[i]='\0';
  printf("string s2: %s",s2);
  getch();
}
```

Output:

Enter string s1:program

String s2: program

Q.5.b) Explain file handling in c in details.[Note: Mention file type, file modes, file related functions and its use) (8 M)

Ans:

There are various types of files like: text file stored with the extension

“txt”, binary files stored with the extension “bin” etc. For file handling or accessing the contents of file, there are certain predefined functions available in the c programming language,

A file represents a sequence of bytes on the disk where a group of Related data is stored. File is created for permanent storage of data. C programming Language can handle files as stream-oriented data (text) files and system oriented Data (binary) files.

An important thing required to access files is the “FILE pointer”. This pointer is used to point to the values stored in the file. A file pointer is hence to be created for accessing the files. The syntax for creating a file pointer is a given below:

FILE *<identifier for pointer>;

For e.g. FILE *fp;

Hence in every program we write in this section to access files, we will use this kind of pointer declaration. This pointer is used to point the data to be accessed in the file i.e. whenever a data is read or written in the file, it is from the location pointed by the file pointer “fp”.

Function and there use:

1. fopen(): This function is used to open a file to be accessed in the program. The file to be opened is to be passed as a string parameter to the function and also the mode of opening the file is to be passed as the string to the function.
   Hence the syntax of the function with parameters is as given below:
   <file pointer identifier>=fopen<”file name>”,<mode of opening the file>”)
   For e.g. fp=fopen(“test.txt”,”w”);
   The various modes in which file can be opened are as follows
   1. “r” indicates that the file is to be opened indicates in the read mode.
   2. “w” indicates that the file is to be opened indicates in the write mode.

      When a file is opened in write mode the data written in the file overwrites the previously stored data in the file.

   3. ”a” indicates that the file is to be opened in the append mode. In this mode the data written into the file is appended towards the end of the already stored data in the file. The earlier stored data remains as it is.
   4. “w+” indicates that the file is to be opened in write and read mode.
   5. “r+” indicates that the file is to be opened in read and write mode.

2. fclose(): The function is used to close the file opened using the file pointer passed to the function. The syntax with parameters to call this function is as given below:
   fclose(<file pointer identifier>);
   for e.g. fclose(fp);
   this statement closes the file opened using the file pointer variable “fp”. It closes the file opened using the function fopen().
3. feof(): This function returns true or false based on whether the pointer pointing to the file has reached the end of the file or not. The pointer used to point the file has to be passed as a parameter to the function feof(). Also the file has to be opened pointed using the pointer “fp”,before using this function. The syntax of the function with the parameters is as shown below:
   feof(<file pointer identifier>);
4. fputc(): This function is used to put a character type data into the opened file using the open() function, pointed by a file pointer. The character to be put into the file as well as the pointer are to be passed as the parameters to this function. The syntax to call this function along with the parameters to be passed is as shown below:
   fputc<char type data>,<file pointer identifier>);

   for e.g.: fputc(c,fp);

   this example will store the character value of the char type data variable. ”c” into the opened file and pointed by the pointer fp at the position pointed by the pointer fp in the file.

5. getc(): This function is used to get a character from the file pointed by the corresponding file pointer passed to the function. It is exactly opposite the fputc() function. This function brings the character from the file opened and pointed by the file pointer variable passed to the function. The syntax of the function call with the parameters to be passed is as given below:
   getc(<file pointer identifier>);
   For e.g. getc(fp);
   This function brings the character type data from the opened file using the pointer fp; from the location pointed by the pointer “fp” in that file.
6. rewind() : this function is used to rewind or bring the file pointer variable to the point to the beginning of the file from wherever it is currently pointing in the file. The syntax of the function call with the parameters to be passed is as given below:
   rewind(<file pointer identifier>);
   for e.g.rewind(fp);
   this function rewinds or brings back the pointer “fp” to the beginning of the file from wherever it was pointing in the file opened using the pointer “fp”.
7. Fprintf():This function is used to store the different data types in the file as the fputc() function is used to store the character in the file. This can be used to store

integers , float, string etc. types of data into the file opened. The function is similar to printf(),except that it writes to the file instead of the monitor. The syntax shown below explains the concept:
fprintf(<file pointer identifier>,"<format specifiers>",<variable names>);
For e.g.: fprintf(fp,"%d",x);
This function will print or store the value of integer variable "x" in the file opened using the pointer "fp". The data will be stored at the location pointed by the pointer variable "fp".

8. Fscanf(): The function is used to read the different types of the data as the getc() Function is used to read a character from the file. This function can be used to read an integer, float string etc. types of data into the file opened. The function is similar to scanf(),except that it reads from the file instead of the keyboard. The syntax shown below explains the concept:
fscanf(<file pointer variable>,"<format specifiers>",<address of the variables in which the data is to be read>);
for e.g.: fscanf(fp,"%d",&x);

Q.5.c) WAP to print all possible combination of 1,2,3 using nested loops.(7 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,k;
clrscr();
for(i=1;i<=3;i++)
        {
            for(j=1;j<=3;j++)
                    {
                            for(k=1;k<=3;k++)
                            printf("\n%d%d%d",i,j,k);
                    }
        }
getch();
```

}

Output:

111

112

113

121

122

123

131

132

133

211

212

213

221

222

223

.

.

.

.

(and so on.)

Q.6.a) WAP to print following pattern for n lines. [Note: range of n is 1-9] **(5 M)**

1

121

12321

1234321

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j,n;
    clrscr();
    printf("Enter the number of lines:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n-i;j++)
        {
            printf(" ");
        }
        for(j=1;j<=i;j++)
        {
            printf("%d",j);
        }
        for(j=i-1;j>=1;j--)
        {
            printf("%d",j);
        }
```

```
            printf("\n");
        }
        getch();
    }
```

Output:

```
1
121
12321
1234321
```

Q.6.b) WAP to print binary equivalent of entered decimal no. (5 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    int n,i;
    clrscr();
    printf("Enter a number: ");
    scanf("%d",&n);
    printf("Binary form is: ");
    for(i=15;i>=0;i--)
    {
        printf("%d",n/(int)(pow(2,i));
```

```
            n=n%(int)(pow(2,i));
        }
        getch();
    }
```

Output:

```
Enter a number:12
Binary form is:0000000000001100
```

Q.6.c) what is significance of storage classes? Explain it with relevant examples. (10 M)

Ans:

The different locations in the computer where we can store data and their accessibility, initial values etc. very based on the way they are declared. These different ways are termed as different storage classes.

In C there are for storage classes, namely

1. Automatic
2. Register
3. Static
4. External or global

Let us see these storage classes one by one

1. Automatic storage class
   In this case data is stored in memory
   The initial value of such a variable is garbage
   The scope of the variable is local i.e. limited to the function in which it is defined.
   The life of such variables is till the control remains in the particular function where it is defined.
   For e.g.:
   Int i; or auto int i;

2. Register storage class
   In this case data is stored in CPU register
   The initial value of such a variable is garbage.
   The scope of the variable is local i.e. limited to the function in which it is defined

The life of such variables is till the control remains in the particular function where it is defined.
For e.g.:
Register int I;
In this case the data is stored in a small memory inside the processor called its registers.
The advantage of such storage class is that since the data is in the processor itself, its access and operation on such data is faster.
There is limitation on the size of the data that can declared to be register storage class. The data should be such that it doesn't require more than 4 bytes. Hence double and long double data types cannot be declared as a register.
Also there is a limitation on the maximum number of variables in a function that can be a register class. The limitation is that a maximum of 3 register class variable can be declared in a function.

3. Static storage class
   In this case data is stored in a memory
   The initial value of such a variable is zero
   The scope of the variable is local i.e. limited to the function in which it is defined
   The life of such variable is till the program is alive.
   For e.g.:
   Static int I;
   If a variable is declared static, its value remains unchanged even If the function execution is completed.
   When the execution to that function returns, the previous value is retained.
   Thus it says the initialization is only once. If you have an initialization statement of a static member, it will be executed only once i.e. for the first time when this function is called.
4. External or global storage class
   In this case data is stored in memory
   The initial value of such a variable is zero.
   The scope of the variable is global i.e. it is accessible from anywhere in the program.
   The life such a variable is till the program is alive.

F.E. Sem - 2 CBCGS SPA – MAY 18

Q.1.a) Select the correct option from multiple choice questions. (10 M)

i. **Which bitwise operator is used to multiply the number by 2^n where n is number of bits.**
A]Bitwise-OR B]Bitwise-AND C]Bitwise Left shift D] Bitwise Right shift
Ans:- C

ii. **Which operator has the lowest priority?**
A] ++ b] % C] + D] ||
Ans:-D

iii. **Which of these is a valid variable declaration?**
A]in temp salary; B] float marks_student; C] float roll-no; D] int main;
Ans:-B

iv. **What will be the output of the following program?**

```
Void main() {
Double x=28;
Int r;
r=x%5;
printf("\n r=%d",r);}
```

A] r= 3 B] Run time Error C] Compile time Error D] None of the above
Ans:-A

v. **What will be the output of following program**

```
Void main() {
 int x[]={10,20,30,40,50};
printf("\n%d%d%d%d",x[4], 3[x], x[2], 1[x], x[0]); }
```

A]Error B}10 20 30 40 50 C] 50 40 30 20 10 D] None of these
Ans:-D

vi. **Which of the following is not a keyword of 'C'?**
A]auto B]register C]int D]function
Ans:-D

vii. **What will be the output?**

```
Void main() {
Int y;
y=0x10+010+10;
printf("\ny=%x",y); }
```

A] y=34 B] x=34 C] y=22 D] Error

Ans:C

viii. **Study the following C program**

```
Void main() {
int a=0;
for( ;a;);
a++; }
```

what will be the value of the variable a, on the execution of above program

A] 1 B] 0 C] -1 D] None of these

Ans:A

ix. **Which of the following is used as a string termination character?**

A] 0 B]\0 C] /0 D]None of these

Ans:B

x.

```
void main() {
char a[]= "Hello world";
char *p;
p=a;
printf("\n%d%d%d%d",sizeof(a),sizeof(p),strlen(a),strlen(p) ); }
```

A] 11 11 10 10 B] 10 10 10 10 C] 12 12 11 11 D] 12 2 11 11

Ans:D

Q.1.b) State True or False with reason. (10 M)

i. Size of pointer variable is equal to the datatype it points to. True

ii. A float constant cannot be used as a case constant in a switch statement. True

iii. The statement void p; is valid. True

iv. while(0); is an infinite loop. False

Ans: when the condition is while(0); the control will never enter into the loop hence there will not be any infinite loop.

v. scanf() function is used to input string having multiple words. True

vi. A function can have any number of return statements. True

vii. In a union, space is allocated to every member individually. False

Ans: A union shares space among its member.

viii. An algorithm is a graphical representation of the logic of a program. False

Ans: An algorithm is set of structured instruction used to execute the code.

ix. Comments in the program make debugging of the program easier. True

x.There is no difference between '\0' and '0'. False

Ans: '\0' is used as a string terminator whereas '0' is number for the interpreter.

Q.2.a.i) How to create array of structure variable and assign values to its members? (5 M)

Ans:

1. An array of structure can be declared in the same way as declaring array of any other data type.
2. For example, an array of the structure student can be declared as shown:
   struct student s[100];
   This array can now store information of 100 students.
3. Array of structure must be used when many variables of a structure are required.
4. Syntax:

```
Struct structure_name
{
    data_type member1;
    data_type member2;
    -
    -
    data_type member;
};
```

Q.2.a.ii) Differentiate between struct and union. When is union preferred over struct? Give on example of each. (5 M)

Ans:

Sr. NO.	Structure	Union
1.	Memory allotted to structure is equal to the space require collectively by all the members of the structure.	Memory allotted for a union is equal to the space required by the largest memory of that union
2.	Data is more secured in structure.	Data can be corrupted in a union.
3.	Structure provide ease of programming.	Unions are comparatively difficult for programming.
4.	Structures require more memory.	Unions require less memory.
5.	Structure must be used when information of all the member elements of a structure are to be stored.	Unions must be used when only one of the member elements of the union is to be stored.

1. Union is preferred over struct when only one of the member elements of the union is stored.
   Example of structure:

```
#include<conio.h>
#include<stdio.h>
Struct student
{
    char name[20];
    int roll_no;
    float fees;
};
void main()
{
    Struct student s1;
    clrscr();
    printf("Enter the student's name, roll number ");
    gets(s1.name);
    scanf("%d",&s1.roll_no);
    printf("The student details are as follows: \nName:%s\n
Roll number:%d",s1.name,s1.roll_no);
    getch();
```

```
}
```

Output:

```
Enter the student's name, roll number:John
20
The student details are as follows:
Name:John
Roll number:20
```

Example of union:

```
#include<stdio.h>
#include<conio.h>
Union info
{
    char name[20];
};
void main()
{
 Union info i1;
int choice;
clrscr();
printf("Enter your name");
scanf("%s",i1.name);
printf("Your name is %s",i1.name);
getch();
}
```

Output:

```
Enter your name:John
Your name is John
```

Q.2.b.i) WAP to print the sum of following series. (5 M)

$1+2^2+3^3+\ldots\ldots+n^n$

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n,sum=0;
 clrscr();
 printf("Enter a number ");
 scanf("%d",&n);
 while(n!=0)
 {
  sum=sum+(n*n);
  n--;
 }
 printf("The sum of the series is %d",sum);
 getch ();
}
```

Output:

```
Enter a number 3
The sum of the series is 14
```

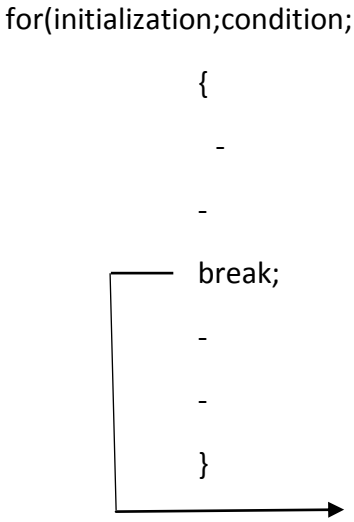
Q.2.b.ii) Compare the following **(5 M)**

i. break and continue statements

Ans:

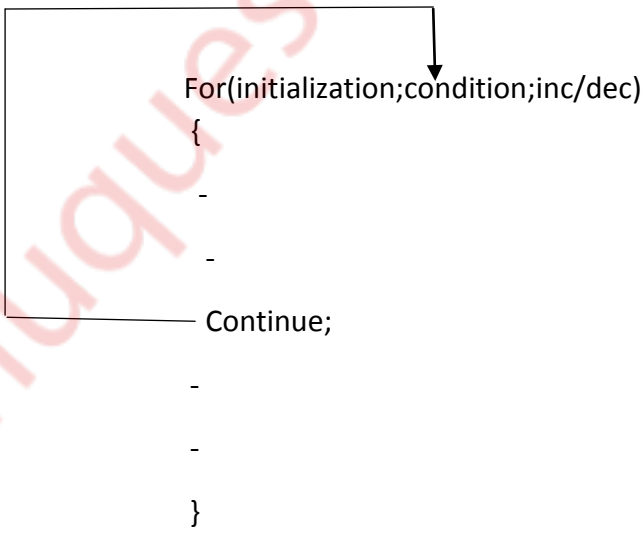
break statement: The break statement neglects the statement after it in the loop and transfer the control outside the loop.

Operation of break statement in for loop:



Continue statement: The continue statement also neglects the statement after it in the loop and transfer the control back to the starting of the loop for next iteration.

Operation of continue statement in for loop:



ii. if-else and switch statements:

Ans:

Sr.no	If……else	Switch
1.	In this we can test only one condition.	In this we can test multiple condition.
2.	It can have values based on constraints.	It can have values based on user choice.
3.	In this we cannot have different conditions.	In this case we can have only one expression but various value of the same expression.
4.	If else statement is used to evaluate a condition to be true or false	A switch case statement is used to test for multiple values of the same variable or expression like 1,2,3 etc.

Q.3.a) Write a program to calculate number of vowels (a, e, i, o, u) separately in the entered string. (6 M)

Ans:

```
#include <stdio.h>
#include<conio.h>

void main()
{
 int c = 0, count = 0,a=0,i=0,e=0,o=0,u=0,A=0,I=0,E=0,O=0,U=0;
 char s[1000];
 clrscr();

 printf("Input a string\n");
 gets(s);

 while (s[c] != '\0')
 {
  if (s[c] == 'a')
   {
```

```
 a++;
 printf("\na=%d",a);
}
else  if (s[c] == 'A')
{
 A++;
 printf("\nA=%d",A);
}
else  if (s[c] == 'e')
{
 e++;
 printf("\ne=%d",e);
}
else  if (s[c] == 'E')
{
 E++;
 printf("\nE=%d",E);
}
else  if (s[c] == 'i')
{
 i++;
 printf("\ni=%d",i);
}
else  if (s[c] == 'I')
{
 I++;
 printf("\nI=%d",I);
}
else  if (s[c] == 'o')
```

```
    {
     o++;
     printf("\no=%d",o);
    }
    else if (s[c] == 'O')
    {
     O++;
     printf("\nO=%d",O);
    }
    else  if (s[c] == 'u')
    {
     u++;
     printf("\nu=%d",u);
    }
    else  if (s[c] == 'U')
    {
     U++;
     printf("\nU=%d",U);
    }

    count++;
   c++;
  }
  getch();
}
```

Output:

Input string University

U=1

i=2

e=1

Q.3.b) Predict output of following program segment. (4 M)

i.

```
main()
{
int a,b,*p1,*p2,x,y;
 clrscr();
 a=48;b=10;p1=&a;p2=&b;
 x=*p1**p2-8;
 *p1=*p1+*p2;
 y=(*p1/ *p2)+20;
 printf("%d%d%d%d%d%d",*p1,*p2,a,b,x,y);
}
```

Output:

5810581047225

ii)

```
main()
{
 int x=4, y=9,z;
 clrscr();
 z=x++ + --y+y;
 printf("\n%d%d%d",x,y,z);
```

```
z= --x+x+y--;
printf("\n%d%d%d",x,y,z);
getch();
}
```

Output:

5820

4716

Q.3.c) An electronic component vendor supplies three products: transistors, resistors and capacitors. The vendor gives a discount of 10% on order of transistor if the order is more than Rs. 1000. On order of more than Rs. 100 for resistors, a discount of 5% is given and discount of 10% is given on order for capacitors of value more than Rs. 500. Assume numeric code 1, 2 and 3 used for transistors, capacitors and resistors respectively. Write a program that reads the product code and order amount and prints out the net amount that the customer is required to pay after discount. (Note: Use switch-case) (10 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int choice,n,dis;
 clrscr();
 printf("\n1.Transistor\n2.Capacitor\n3.Resistor\nEnter your choice");
 scanf("%d",&choice);
 printf("Enter the price");
 scanf("%d",&n);
 switch(choice)
 {
  case 1:if(n>1000)
            {
             dis=n/10;
```

```
        dis=n-dis;
        printf("The total price after discount is %d",dis);
       }
       else
       {
        dis=n;
        printf("The total price is %d",dis);
       }
       break;
case 2:if(n>500)
       {
        dis=n/10;
        dis=n-dis;
        printf("The total price after discount is %d",dis);
       }
       else
       {
        dis=n;
        printf("The total price is %d",dis);
       }
       break;
case 3:if(n>100)
       {
        dis=n/5;
        dis=n-dis;
        printf("The total price after discount is %d",dis);
       }
       else
       {
```

```
            dis=n;
            printf("The total price is %d",dis);
           }
           break;
 }
getch();
}
```

Output:

1.Transistor

2.Capacitors

3.Resistor

Enter your choice 1

Enter the price 1050

The total price after discount is 945

Q.4.a) What is recursion? WAP using recursion to find the sum of array elements of size n.

(10 M)

Ans:

1. Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.
2. A recursive function must definitely have a condition that exits from calling the function again.
3. Hence there must be a condition that calls the function itself if that condition is true.
4. If the condition is false then it will exit from the loop of calling itself again.

Program:

```
#include<conio.h>
#include <stdio.h>
#define MAX_SIZE 100

int sum(int arr[], int start, int len);
```

```
void main()
{
    int arr[MAX_SIZE];
    int N, i, sumof_array;
    clrscr();
    printf("Enter size of the array: ");
    scanf("%d", &N);
    printf("Enter elements in the array: ");
    for(i=0; i<N; i++)
    {
     scanf("%d", &arr[i]);
    }


    sumof_array = sum(arr, 0, N);
    printf("Sum of array elements: %d", sumof_array);

    getch();
}
int sum(int arr[], int start, int len)
{
    if(start >= len)
     return 0;

    return (arr[start] + sum(arr, start + 1, len));
}
```

Output:

Enter size of the array: 10

Enter elements in the array: 1 2 3 4 5 6 7 8 9 10

Sum of array Elements: 55

Q.4.b) Write a C program to **(10 M)**

i. **Create a 2D array [Matrix] [in main function]**
ii. **Write a function to read 2D array[Matrix]**
iii. **Write a function that will return true(1) if entered matrix is symmetric or false(0) is not symmetric.**
iv. **Print whether entered matrix is symmetric or not [in main function]**

Ans:

Program:

```
#include<stdio.h>

void main()
{
  int m, n, c, d, matrix[10][10], transpose[10][10];
  clrscr();
  printf("Enter the number of rows and columns of matrix\n");
  scanf("%d%d", &m, &n);
  printf("Enter elements of the matrix\n");
  for (c = 0; c < m; c++)
   for (d = 0; d < n; d++)
     scanf("%d", &matrix[c][d]);
  for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
      transpose[d][c] = matrix[c][d];
  if (m == n)
```

```
 {
   for (c = 0; c < m; c++)
   {
     for (d = 0; d < m; d++)
     {
          if (matrix[c][d] != transpose[c][d])
            break;
     }
     if (d != m)
          break;
   }
   if (c == m)
     printf("The matrix is symmetric.\n");
   else
     printf("The matrix isn't symmetric.\n");
 }
 else
   printf("The matrix isn't symmetric.\n");

 getch();
}
```

Output:

Enter the number of rows and columns of matrix

2 2

Enter elements of matrix

1 2 3 4

The matrix isn’t symmetric.

Q.5.a) implement string copy function STRCOPY (str1,str2) that copies a string str1 (source) to another string str2 (destination) without using library function. (5 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 char s1[100],s2[100],i;
clrscr();
 printf("enter string s1: ");
scanf("%s",s1);
for(i=0;s1[i]!='\0';++i)
 {
        s2[i]=s1[i];
 }
  s2[i]='\0';
  printf("string s2: %s",s2);
  getch();
}
```

Output:

Enter string s1:program

String s2: program

Q.5.b) Explain file handling in c in details.[Note: Mention file type, file modes, file related functions and its use) (8 M)

Ans:

There are various types of files like: text file stored with the extension

“txt”, binary files stored with the extension “bin” etc. For file handling or accessing the contents of file, there are certain predefined functions available in the c programming language,

A file represents a sequence of bytes on the disk where a group of Related data is stored. File is created for permanent storage of data. C programming Language can handle files as stream-oriented data (text) files and system oriented Data (binary) files.

An important thing required to access files is the “FILE pointer”. This pointer is used to point to the values stored in the file. A file pointer is hence to be created for accessing the files. The syntax for creating a file pointer is a given below:

FILE *<identifier for pointer>;

For e.g. FILE *fp;

Hence in every program we write in this section to access files, we will use this kind of pointer declaration. This pointer is used to point the data to be accessed in the file i.e. whenever a data is read or written in the file, it is from the location pointed by the file pointer “fp”.

Function and there use:

1. fopen(): This function is used to open a file to be accessed in the program. The file to be opened is to be passed as a string parameter to the function and also the mode of opening the file is to be passed as the string to the function.
   Hence the syntax of the function with parameters is as given below:
   <file pointer identifier>=fopen<”file name>”,<mode of opening the file>”)
   For e.g. fp=fopen(“test.txt”,”w”);
   The various modes in which file can be opened are as follows
   1. “r” indicates that the file is to be opened indicates in the read mode.
   2. “w” indicates that the file is to be opened indicates in the write mode.

      When a file is opened in write mode the data written in the file overwrites the previously stored data in the file.

   3. ”a” indicates that the file is to be opened in the append mode. In this mode the data written into the file is appended towards the end of the already stored data in the file. The earlier stored data remains as it is.
   4. “w+” indicates that the file is to be opened in write and read mode.
   5. “r+” indicates that the file is to be opened in read and write mode.

2. fclose(): The function is used to close the file opened using the file pointer passed to the function. The syntax with parameters to call this function is as given below:
   fclose(<file pointer identifier>);
   for e.g. fclose(fp);
   this statement closes the file opened using the file pointer variable “fp”. It closes the file opened using the function fopen().
3. feof(): This function returns true or false based on whether the pointer pointing to the file has reached the end of the file or not. The pointer used to point the file has to be passed as a parameter to the function feof(). Also the file has to be opened pointed using the pointer “fp”,before using this function. The syntax of the function with the parameters is as shown below:
   feof(<file pointer identifier>);
4. fputc(): This function is used to put a character type data into the opened file using the open() function, pointed by a file pointer. The character to be put into the file as well as the pointer are to be passed as the parameters to this function. The syntax to call this function along with the parameters to be passed is as shown below:
   fputc<char type data>,<file pointer identifier>);

   for e.g.: fputc(c,fp);

   this example will store the character value of the char type data variable. ”c” into the opened file and pointed by the pointer fp at the position pointed by the pointer fp in the file.

5. getc(): This function is used to get a character from the file pointed by the corresponding file pointer passed to the function. It is exactly opposite the fputc() function. This function brings the character from the file opened and pointed by the file pointer variable passed to the function. The syntax of the function call with the parameters to be passed is as given below:
   getc(<file pointer identifier>);
   For e.g. getc(fp);
   This function brings the character type data from the opened file using the pointer fp; from the location pointed by the pointer “fp” in that file.
6. rewind() : this function is used to rewind or bring the file pointer variable to the point to the beginning of the file from wherever it is currently pointing in the file. The syntax of the function call with the parameters to be passed is as given below:
   rewind(<file pointer identifier>);
   for e.g.rewind(fp);
   this function rewinds or brings back the pointer “fp” to the beginning of the file from wherever it was pointing in the file opened using the pointer “fp”.
7. Fprintf():This function is used to store the different data types in the file as the fputc() function is used to store the character in the file. This can be used to store

integers , float, string etc. types of data into the file opened. The function is similar to printf(),except that it writes to the file instead of the monitor. The syntax shown below explains the concept:
fprintf(<file pointer identifier>,"<format specifiers>",<variable names>);
For e.g.: fprintf(fp,"%d",x);
This function will print or store the value of integer variable "x" in the file opened using the pointer "fp". The data will be stored at the location pointed by the pointer variable "fp".

8. Fscanf(): The function is used to read the different types of the data as the getc() Function is used to read a character from the file. This function can be used to read an integer, float string etc. types of data into the file opened. The function is similar to scanf(),except that it reads from the file instead of the keyboard. The syntax shown below explains the concept:
fscanf(<file pointer variable>,"<format specifiers>",<address of the variables in which the data is to be read>);
for e.g.: fscanf(fp,"%d",&x);

Q.5.c) WAP to print all possible combination of 1,2,3 using nested loops.(7 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,k;
clrscr();
for(i=1;i<=3;i++)
        {
            for(j=1;j<=3;j++)
                    {
                            for(k=1;k<=3;k++)
                            printf("\n%d%d%d",i,j,k);
                    }
        }
getch();
```

}

Output:

111

112

113

121

122

123

131

132

133

211

212

213

221

222

223

.

.

.

.

(and so on.)

Q.6.a) WAP to print following pattern for n lines. [Note: range of n is 1-9] **(5 M)**

1

121

12321

1234321

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j,n;
    clrscr();
    printf("Enter the number of lines:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n-i;j++)
        {
            printf(" ");
        }
        for(j=1;j<=i;j++)
        {
            printf("%d",j);
        }
        for(j=i-1;j>=1;j--)
        {
            printf("%d",j);
        }
```

```
            printf("\n");
        }
        getch();
    }
```

Output:

```
1
121
12321
1234321
```

Q.6.b) WAP to print binary equivalent of entered decimal no. (5 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    int n,i;
    clrscr();
    printf("Enter a number: ");
    scanf("%d",&n);
    printf("Binary form is: ");
    for(i=15;i>=0;i--)
    {
        printf("%d",n/(int)(pow(2,i));
```

```
            n=n%(int)(pow(2,i));
        }
        getch();
    }
```

Output:

```
Enter a number:12
Binary form is:0000000000001100
```

Q.6.c) what is significance of storage classes? Explain it with relevant examples. (10 M)

Ans:

The different locations in the computer where we can store data and their accessibility, initial values etc. very based on the way they are declared. These different ways are termed as different storage classes.

In C there are for storage classes, namely

1. Automatic
2. Register
3. Static
4. External or global

Let us see these storage classes one by one

1. Automatic storage class
   In this case data is stored in memory
   The initial value of such a variable is garbage
   The scope of the variable is local i.e. limited to the function in which it is defined.
   The life of such variables is till the control remains in the particular function where it is defined.
   For e.g.:
   Int i; or auto int i;

2. Register storage class
   In this case data is stored in CPU register
   The initial value of such a variable is garbage.
   The scope of the variable is local i.e. limited to the function in which it is defined

The life of such variables is till the control remains in the particular function where it is defined.
For e.g.:
Register int I;
In this case the data is stored in a small memory inside the processor called its registers.
The advantage of such storage class is that since the data is in the processor itself, its access and operation on such data is faster.
There is limitation on the size of the data that can declared to be register storage class. The data should be such that it doesn't require more than 4 bytes. Hence double and long double data types cannot be declared as a register.
Also there is a limitation on the maximum number of variables in a function that can be a register class. The limitation is that a maximum of 3 register class variable can be declared in a function.

3. Static storage class
   In this case data is stored in a memory
   The initial value of such a variable is zero
   The scope of the variable is local i.e. limited to the function in which it is defined
   The life of such variable is till the program is alive.
   For e.g.:
   Static int I;
   If a variable is declared static, its value remains unchanged even If the function execution is completed.
   When the execution to that function returns, the previous value is retained.
   Thus it says the initialization is only once. If you have an initialization statement of a static member, it will be executed only once i.e. for the first time when this function is called.
4. External or global storage class
   In this case data is stored in memory
   The initial value of such a variable is zero.
   The scope of the variable is global i.e. it is accessible from anywhere in the program.
   The life such a variable is till the program is alive.

STUCTURED PROGRAMMING APPROACH (MAY 19)

Q.1)

a) Attempt the Multiple Choice Questions (6 M)

i.
```
#include<stdio.h>
int main()
{
    int a=0;
    a=a++ + a++ - a++ + ++a;
    printf("%d",a);
    return 0;

}
```

The output of above program is
(a) 2
(b) 90
(c) 3
(d) Error

Ans: a) 2

ii. Which of the following operator can be used to access value at address stored in a pointer variable?
(a) *
(b) @
(c) &
(d) &&

Ans: a) *

iii. In C programming which of the operator have the highest precedence
(a) Relational
(b) Arithmetic
(c) Bitwise
(d) Logical

Ans: b) Arithmetic

iv. Which of the following are themselves a collection of different data types?
(a) String
(b) 2D arrays

(c) Structures
(d) Char

Ans: c) Structures

v. If x and b are the float variables what is the value of x when x=sizeof(b).
(a) 2
(b) 3
(c) 4
(d) Null

Ans: c) 4

vi. The default value of static storage class variable is
(a) Zero
(b) Garbage
(c) One
(d) None of the above

Ans: a) Zero

---

b) Find the output of following **(4 M)**

i.
```
#include<stdio.h>
int main()
{
        int a=1;
        do{
                a++;
                ++a;
        }while(a++>25);
        printf("%d\n",a);
        return 0;
}
```

Ans: 4

ii.
```
#include<stdio.h>
int main()
{
        int x;
        x=10;
```

```
        if(x>10)
                x-=10;
        else if(x>=0)
                x+=00;
        else if(x)
                x+=10;
        else
                x-=10;
        printf("%d\n",x);
        return 0;
}
```

Ans: 10

---

c) Convert the following **(6 M)**

i. **153 from base 10 to Hexadecimal.**

Ans:

16	153
9	9
	9

Hence, $(153)_{10} = (99)_{16}$

ii. **1100 1101 1001 1011 from base 2 to decimal.**

Ans:

$1*2^{15}+1*2^{14}+0*2^{13}+0*2^{12}+1*2^{11}+1*2^{10}+0*2^{9}+1*2^{8}+1*2^{7}+0*2^{6}+0*2^{5}+1*2^{4}+1*2^{3}+0*2^{2}+1*2^{1}+1*2^{0}$

= 32768+16384+0+0+2048+1024+0+256+128+0+0+16+8+0+2+1

=52635

iii. **143 from base 8 to decimal.**

Ans:

$(143)_8 = 1*8^2 + 4*8^1 + 3*8^0$

= 64+32+3

$= 99$

$(143)_8 = (99)_{10}$

---

d) Differentiate between Structure and Union

Ans:

Sr. NO.	Structure	Union
1.	Memory allotted to structure is equal to the space require collectively by all the members of the structure.	Memory allotted for a union is equal to the space required by the largest memory of that union
2.	Data is more secured in structure.	Data can be corrupted in a union.
3.	Structure provide ease of programming.	Unions are comparatively difficult for programming.
4.	Structures require more memory.	Unions require less memory.
5.	Structure must be used when information of all the member elements of a structure are to be stored.	Unions must be used when only one of the member elements of the union is to be stored.

Q.2)

a) What is recursion? Write a program using recursion to calculate value of $Z=X^Y$. (8 M)

Ans:

1. Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.
2. A recursive function must definitely have a condition that exits from calling the function again.
3. Hence there must be a condition that calls the function itself if that condition is true.
4. If the condition is false then it will exit from the loop of calling itself again.

Program:

```
#include <stdio.h>

int power(int n1, int n2);

int main()
{
   int base, powerRaised, result;
   printf("Enter base number: ");
   scanf("%d",&base);
   printf("Enter power number: ");
   scanf("%d",&powerRaised);
   result = power(base, powerRaised);
   printf("%d^%d = %d", base, powerRaised, result);
   return 0;
}
int power(int base, int powerRaised)
{
   if (powerRaised != 0)
      return (base*power(base, powerRaised-1));
   else
      return 1;
}
```

Output:

```
Enter base number: 3
Enter power number: 4
3^4 = 81
```

---

b) Write a function to reverse a 3 digit number. **(6 M)**

Ans:

Program:

```
#include <stdio.h>
int main()
{
  int n, reverse = 0;
  printf("Enter a number to reverse:\n");
```

```
  scanf("%d", &n);
  while (n != 0)
  {
    reverse = reverse * 10;
    reverse = reverse + n%10;
    n       = n/10;
  }
  printf("Reverse of entered number is = %d\n", reverse);
  return 0;
}
```

Output:

Enter a number to reverse : 123
Reverse of entered number is = 321

---

c) Write a program to display following pattern **(6 M)**

```
        0
      0  1
    0  1  0
  0  1  0  1
0  1  0  1  0
```

Ans:

Program:

```
#include<stdio.h>
int main()
{
        int i,j,k;
        for(i=0 ; i<=4 ; i++)
        {
                for(j=4 ; j > i ; j--)
                printf(" ");
                for(k=0; k<=i ; k++)
                {
                if(k%2 == 0)
                printf("0");
```

```
                else
                printf("1");// else 1
                }
                printf("\n");
        }
        return 0;
}
```

Output:

```
          0
        0  1
      0  1  0
    0  1  0  1
  0  1  0  1  0
```

Q.3)

a) Write a program to find the frequency of digit in a set of numbers and remove duplicates from an array. For ex. Array A={1,2,3,4,2,5,2} frequency of 2 is 3 and resultant array is A={1,2,3,4,5} (10 M)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[7]={1,2,3,4,2,5,2};
int b[10];
int i,j,count=0;
clrscr();
for(i=0;i<6;i++)
{
 for(j=0;j<count;j++)
 {
  if(a[i]==b[j])
  break;
 }
 if(j==count)
 {
```

```
b[count] = a[i];
count++;
}
}
for(i=0;i<count;i++)
printf("%d",b[i]);
getch();
}
```

Output:

12345

---

b) Explain the concept of nested structure? Declare a structure to enter employee Information like name, id, salary, date of joining. Use nested structure to get the address of an employee. Write a program to read 10 records and display them. (10 M)

Ans:

- If one of the members of structure is also a structure, then such a structure is called as a nested structure.
- The structure variables can be a normal structure variable or a pointer variable to access the data.
- The syntax of a nested structure can be as given below :

```
struct structure_name
{
data_type variable_name;
-
struct
{
data_type variable_name;
-
-
} internal_structure_name;
-
-
};
```

Program:

```
#include<stdio.h>
#include<conio.h>
struct Employee
{
 name[50];
 int id;
 int salary;
 char doj[50];
 struct
 {
  char add[50];
 }address;
};
void main()
{
 struct Employee e[100];
 int n,i;
 clrscr();
 printf("Enter the number of Employee");
 scanf("%d",&n);
 for(i=0;i<=n-1;i++)
 {
  printf("\nEnter ID,Name,Date of joining, Salary, Address of employee");

scanf("%d%s%s%d%s",&e[i].id,&e[i].name,&e[i].doj,&e[i].salary,&e[i].addre
ss.add);
 }
  printf("\nID\t\tName\t\tDOJ\t\tSalary\t\tAddress\n");
  printf("--------------------------------------------------------------------------------");
  for(i=0;i<=n-1;i++)
  {

 printf("\n%d\t\t%s\t\t%s\t\t%d\t\t%s\n",e[i].id,e[i].name,e[i].doj,e[i].salary,e[i].
 address.add);
  }
  getch();
}
```

Output:

```
Enter number of employee 10

Enter ID, Name, Date of joining, Salary, Address of employee 101
John
22Aug2017
20000
London

Enter ID, Name, Date of joining, Salary, Address of employee 102
Sam
20Aug2017
20000
U.S.

Enter ID, Name, Date of joining, Salary, Address of employee 103
Miller
25Aug2017
20000
U.S.

Enter ID, Name, Date of joining, Salary, Address of employee 104
Dale
22Sept2017
20000
Dubai

Enter ID, Name, Date of joining, Salary, Address of employee 105
Smith
12Aug2017
20000
London

Enter ID, Name, Date of joining, Salary, Address of employee 106
Jam
22Aug2017
20000
U.K.

Enter ID, Name, Date of joining, Salary, Address of employee 107
```

David
22Oct2017
20000
Australia

Enter ID, Name, Date of joining, Salary, Address of employee 108
Dhoni
22Dec2017
20000
India

Enter ID, Name, Date of joining, Salary, Address of employee 109
Johny
22Aug2017
20000
Dubai

Enter ID, Name, Date of joining, Salary, Address of employee 110
Kohli
22Jan2018
50000
India

Enter ID, Name, Date of joining, Salary, Address of employee 101
Rohit
22Nov2018
20000
India

ID	Name	Date of joining	Salary	Address
101	John	22Aug2017	20000	London
102	Sam	20Aug2017	20000	U.S.
103	Miller	25Aug2017	20000	U.S.
104	Dale	22Sept2017	20000	Dubai
105	Smith	12Aug2017	20000	London
106	David	22Oct2017	20000	Australia
107	Dhoni	22Dec2017	20000	India
108	Johny	22Aug2017	20000	Dubai
109	kohli	22Jan2018	50000	India
110	Rohit	22Nov2017	20000	India

Q.4)

a) Explain strcat() & strcpy() with example. Also write a program to check whether entered string is palindrome or not without using inbuilt functions.

(10 M)

Ans:

strcat() function:

- This function concatenates (joins) the two string variables passed to it. It returns a string of the combination of the two in the first string variable.

Syntax: strcat(str1,str2)

Example: strcat(a,b); This will concatenate string a & b.

strcpy() function:

- This function copies the second string into the first string passed to the function.
- The second string remains unchanged. Only the first string is changed and gets a copy of the second string.

Syntax: strcpy(str1,str2)

Example: strcpy(b,a); This will copy the string from a to b.

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char text[100];
  int begin, middle, end, length = 0;
  clrscr();
  printf("Enter a String");
  gets(text);
  while (text[length] != '\0')
    length++;
  end = length - 1;
  middle = length/2;
  for (begin = 0; begin < middle; begin++)
```

```
  {
    if (text[begin] != text[end])
    {
      printf("Entered string is not a palindrome.\n");
      break;
    }
    end--;
  }
  if (begin == middle)
    printf("Entered string is Palindrome.\n");
  getch();
}
```

Output:
Enter a string nitin
Entered string is Palindrome.

---

b) Differentiate between call by value and call by reference. Write a program to calculate factorial of a number using call by reference. **(10M)**

Ans:

Call by Value	Call by Reference
In this case the value of the parameters is passed to the caller function.	In this case the reference of the variable is passed to the function by passing the address of parameters.
In this case the actual parameters are not accessible by the called function.	In this case, since the address of the variables are available, the called function can access the actual parameters.
This is implemented by using simple variable names.	This is implemented by the use of pointer variables.
Hence the actual parameters remain unchanged in case of the call by value.	Hence the actual parameters can be altered if required in case of the call by reference method.

Program:

```
#include<stdio.h>
#include<conio.h>
void fact(int,int*,int*);
void main()
{
int a,fac,sqr;
clrscr();
printf("Enter the number: ");
scanf("%d",&a);
fact(a,&fac,&sqr);
printf("Factorial = %d.\n",fac);
getch();
}
void fact(int x,int *y,int *z)
{
int i;
*y=1;
for(i=1;i<=x;i++)
*y*=i;
}
```

Output:
Enter the number: 4
Factorial = 24.

Q.5)

a) What is file? Explain the functions available for reading & Writing data of files? Write a program to take student information from user and store it in file. (10 M)

Ans:

- For file handling or accessing the contents of file, there are certain predefined functions available in the C programming language.
- An important thing required to access files is the "FILE pointer". This pointer is used to point to the values stored in the file. A file pointer is hence to be created

for accessing the files. The syntax for creating a file pointer is as given below: FILE *<identifier for pointer>; For e.g. FILE *fp;

- Hence in every program we write in this section to access files, we will use this kind of pointer declaration. This pointer is used to point the data to be accessed in the file i.e. whenever a data is read or written in the file, it is from the location pointed by the file pointer "fp".
- **File operations are as follows:**
  1. **fopen()**: This function is used to open a file to be accessed in the program.
     - The file to be opened is to be passes as a string parameter to the function and also the mode of opening the file is to be passed as the string to the function.
     - Hence the syntax of the function with parameters is as given below : <file pointer identifier> = fopen("<file name>", " <mode of opening the file>")
     - For e.g. fp=fopen("test.txt","w"); This example statement opens the file "test.txt" in write mode and the pointer used along with the function to read/write is the file pointer "fp".
     - The various modes in which a file can be opened are as listed below :
       I. "r" indicates that the file is to be opened indicates in the read mode.
       II. "w" indicates that the file is to be opened indicates in the write mode. When a file is opened in write mode the data written in the file overwrites the previously stored data in the file.
       III. "a" indicates that the file is to be opened in the append mode. In this mode the data written into the file is appended towards the end of the already stored data in the file. The earlier stored data remains as it is.
       IV. "w+" indicates that the file is to be opened in write and read mode (v) "r+" indicates that the file is to be opened in read and write mode.

  2. **fclose():** This function is used to close the file opened using the file pointer passed to the function.
     - The syntax with parameters to call this function is as given below : fclose(<file pointer identifier>);
     - For e.g. fclose(fp);

  3. **fputc():** This function is used to put a character type data into the opened file using the fopen() function, pointed by a file pointer.
- The syntax to call this function along with the parameters to be passed is as shown below : fputc(<char type data>, <file pointer identifier>);

- For e.g. : fputc(c,fp); This example will store the character value of the char type data variable "c" into the opened file and pointed by the file pointer fp, at the position pointed by the pointer fp in the file.

4. **getc()**: This function is used to get a character from the file pointed by the corresponding file pointer passed to the function.
    - It is exactly opposite the fputc() function. This function brings the character from the file opened and pointed by the file pointer variable passed to the function.
    - The syntax of the function call with the parameters to be passed is as given below :getc(<file pointer identifier>);
    - For e.g. getc(fp);

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int main()
{
  int roll,i;
  FILE *fptr;
  char na[20],addr[20];
  clrscr();
  fptr = fopen("C:\\TURBOC3\\BIN\\abc.txt","w");
  if(fptr == NULL)
  {
    printf("Error!");
    exit(1);
  }

  printf("Enter Roll no., Name and Address of student: ");
  scanf("%d%s%s",&roll,&na[i],&addr[i]);

  fprintf(fptr,"Roll no: %d\n Name: %s\n, Address %s",roll,na[i],addr[i]);
  fclose(fptr);

  getch();
}
```

---

b) Write a program to calculate sum of diagonal elements of matrix.

1 2 3
5 9 8
1 4 7

Calculate the sum of diagonal elements = 3+9+1 =13. (10 M)

Ans:

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int mat[12][12];
   int i,j,row,col,sum=0;
   printf("Enter the number of rows and columns for 1st matrix\n");
   scanf("%d%d",&row,&col);
   printf("Enter the elements of the matrix\n");
   for(i=0;i<row;i++)
   {
        for(j=0;j<col;j++)
        {
           scanf("%d",&mat[i][j]);
        }
   }

   printf("The matrix\n");
   for(i=0;i<row;i++)
   {
        for(j=0;j<col;j++)
        {
           printf("%d\t",mat[i][j]);
        }
        printf("\n");
   }
    for(i=0;i<row;i++)
   {
        for(j=0;j<col;j++)
        {
           if(i==j)
           {
```

```
                sum=sum+mat[i][j];
            }
        }
    }

    printf("The sum of diagonal elements of a square matrix = %d\n",sum);
    getch();
}
```

Output:

```
Enter the number of rows and columns for matrix
3
3
Enter the elements of the matrix
1
2
3
4
5
6
7
8
9
The Matrix
1       2       3
4       5       6
7       8       9
The sum of the diagonal element of a square matrix = 15
```

Q.6)

a) Explain storage classes with example. (10 M)

Ans:

- The different locations in the computer where we can store data and their accessibility, initial values etc. vary based on the way they are declared. These different ways are termed as different storage classes.
- In C, we have four storage classes, namely

  1. Automatic

2. Register

3. Static

4. External or Global

- Let us see these storage classes one by one

1. Automatic storage class

- In this case data is stored in memory
- The initial value of such a variable is garbage.
- The scope of the variable is local i.e. limited to the function in which it is defined.
- The life of such variables is till the control remains in the particular function where it is defined.
- For e.g.:

  int i; or auto int i;

- In all our programs till now we have been using the automatic storage class for our variables.

2. Register storage class

- In this case data is stored in CPU register
- The initial value of such a variable is garbage.
- The scope of the variable is local i.e. limited to the function in which it is defined.
- The life of such variables is till the control remains in the particular function where it is defined.
- For e.g.:

  register int i;

- In this case the data is stored in a small memory inside the processor called as its registers.
- The advantage of such storage class is that since the data is in the processor itself, its access and operations on such data is faster.
- There is a limitation on the size of the data that can be declared to be register storage class. The data should be such that it doesn't require more than 4 bytes. Hence double and long double data types cannot be declared as register.
- Also there is a limitation on the maximum number of variables in a function that can be of register class. The limitation is that a maximum of 3 register class variable can be declared in a function.

3. Static storage class

- In this case data is stored in memory
- The initial value of such a variable is zero.
- The scope of the variable is local i.e. limited to the function in which it is defined.
- The life of such variables is till the program is alive.
- For e.g. :

  static int i;

- If a variable is declared static, its value remains unchanged even if the function execution is completed.
- When the execution to that function returns, the previous value is retained.
- Thus it says the initialization is only once. If you have an initialization statement of a static member, it will be executed only once i.e. for the first time when this function is called.

Example of register storage class: Addition of Two numbers

```
#include<stdio.h>
int main()
{
int num1,num2;
register int sum;
printf("\nEnter the Number 1 : ");
scanf("%d",&num1);
printf("\nEnter the Number 2 : ");
scanf("%d",&num2);
sum = num1 + num2;
printf("\nSum of Numbers : %d",sum);
return(0);
}
```

---

b) Write a program to find the greatest number among three entered number using ternary operator. (4 M)

Ans:

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
 int a,b,c,greatest;
 clrscr();
 printf("Enter three numbers:\n");
 scanf("%d%d%d",&a,&b,&c);
 greatest = a>b?(a>c ? a:c) : (b>c?b:c);
 printf("The greatest Number is: %d", greatest);
 getch();
}
```

Output:

```
Enter three numbers:
4
5
3
The greatest Number is: 5
```

---

c) Write a program to calculate the sum of series

x-x/2!+X/3!-x/4!.....x/n! **(6 M)**

Ans:

Program:

```
#include <math.h>
#include <stdio.h>
void main()
{
 int n,x,fact=1,series,i,se;
 clrscr();
 printf("Enter the value of n:");
 scanf("%d",&n);
 printf("Enter the value of x:");
 scanf("%d",&x);
 for(i=1;i<=n-1;i++)
 {
  fact = fact*i;
  if(i%2==0)
  {
   se= -x/i*fact;
   series= x+se;
```

```
  }
 }
 printf("The result of series is: %d",series);
 getch();
}
```

Output:

```
Enter the value of n: 5
Enter the value of x: 8
The result of series is: -40
```

STUCTURED PROGRAMMING APPROACH (DEC 19)

Q.P.CODE: 76907

Q.1.a) Attempt the Multiple choice Questions. (6 M)

i. The format identifier '%i' is also used for _____ data type?

a) Char b) int c) double d) float

Ans: b) int

ii. Which keyword can be used for coming out of recursion?

a) Break b) exit c) return d) all of above

Ans: b) exit

iii. What will happen if in a C program you assign a value to an array element whose subscript exceeds the size of array?

a) The element will be set to 0
b) The compiler would report an error
c) The program may crash if some important data gets overwritten
d) The array size would appropriately grow.

Ans: d) the array size would appropriately grow

iv. A pointer is

a) A keyword used to create variable
b) A variable that stores address of an instruction
c) A variable that stores address of other variable
d) All of the above

Ans: c) A variable that stores address of other variable.

v. In which order do the following gets evaluated

1. Relational
2. Arithmetic
3. Logical
4. Assignment

a) 2134 b) 1234 c) 4321 d) 3214

Ans: a) 2134

vi. Which of the following cannot a be structure member?

a) Another structure b) Function c) Array d) None of the mentioned

Ans: a) Another structure.

---

b) Find the output of the following: (4 M)

i.

```
#include<stdio.h>
Void main()
{
 int c;
 for(c=1;c<=5;)
 printf("%d",++c);
}
```

Ans:

2 3 4 5 6

ii. How many 'x' are printed by the following code?

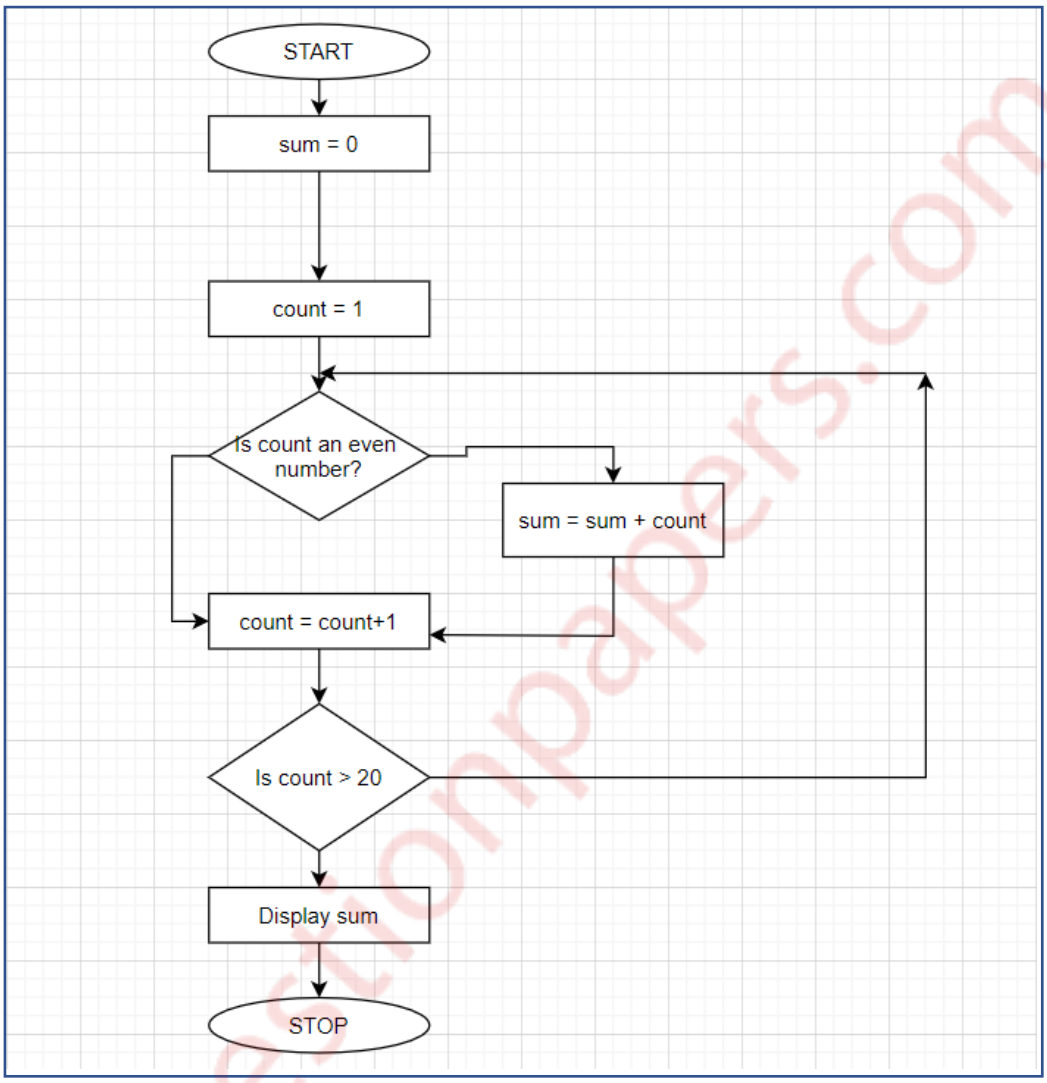
```
#include<stdio.h>
Void main()
{
 Int i=5;
 While(i-->0)
 Printf("x");
 Printf("x");
}
```

Ans: 6

---

c) Draw a flowchart for printing the sum of even terms contained within the numbers 0-20. (4 M)

Ans:

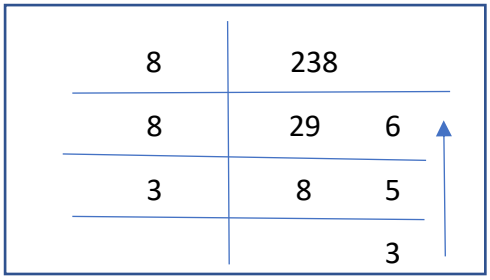


---

d) Solve the following (6 m)

i. Convert 238 decimal to octal

Ans:



Then, when we put the remainders together in reverse order, we get the answer. The decimal number 238 converted to octal is therefore: 356.

ii. Convert A3D Hexadecimal to decimal

Ans:

$(A3D)_{16} = (10 \times 16^2) + (3 \times 16^1) + (13 \times 16^0) = (2621)_{10}$

Q.2)

a) Distinguish Between (6 M)

a) While and do-while loop

Ans:

While Loop	Do-While Loop
Condition is checked first then statement(s) is executed.	Statement(s) is executed atleast once, thereafter condition is checked.
It might occur statement(s) is executed zero times, If condition is false.	At least once the statement(s) is executed.
No semicolon at the end of while. while(condition)	Semicolon at the end of while. while(condition);
If there is a single statement, brackets are not required.	Brackets are always required.
Variable in condition is initialized before the execution of loop.	variable may be initialized before or within the loop.
while loop is entry controlled loop.	do-while loop is exit controlled loop.
while(condition) { statement(s); }	do { statement(s); } while(condition);

---

ii) break and continue

Ans:

Break	Continue
A break can appear in both switch and loop (for, while, do) statements.	A continue can appear only in loop (for, while, do) statements.
A break causes the switch or loop statements to terminate the moment it is executed. Loop or switch ends abruptly when break is encountered.	A continue doesn't terminate the loop, it causes the loop to go to the next iteration. All iterations of the loop are executed even if continue is encountered. The continue statement is used to skip statements in the loop that appear after the continue.
The break statement can be used in both switch and loop statements.	The continue statement can appear only in loops. You will get an error if this appears in switch statement.

When a break statement is encountered, it terminates the block and gets the control out of the switch or loop.	When a continue statement is encountered, it gets the control to the next iteration of the loop.
A break causes the innermost enclosing loop or switch to be exited immediately.	A continue inside a loop nested within a switch causes the next loop iteration.

---

b) Write a C program that will convert a decimal number into any base.

(6 M)

Ans:

```
#include <math.h>
#include <stdio.h>
long long convert(int n);
int main() {
   int n;
   printf("Enter a decimal number: ");
   scanf("%d", &n);
   printf("%d in decimal = %lld in binary", n, convert(n));
   return 0;
}
long long convert(int n) {
   long long bin = 0;
   int rem, i = 1, step = 1;
   while (n != 0) {
      rem = n % 2;
      printf("Step %d: %d/2, Remainder = %d, Quotient = %d\n", step++, n, rem, n /
2);
      n /= 2;
      bin += rem * i;
      i *= 10;
   }
   return bin;  }
```

Output:

```
Enter a decimal number: 19
Step 1: 19/2, Remainder = 1, Quotient = 9
Step 2: 9/2, Remainder = 1, Quotient = 4
Step 3: 4/2, Remainder = 0, Quotient = 2
Step 4: 2/2, Remainder = 0, Quotient = 1
Step 5: 1/2, Remainder = 1, Quotient = 0
19 in decimal = 10011 in binary
```

---

c) Write a C program to calculate the sum of following series without pow() library function S=1-x+x*2/2!-x*3/3!+……N terms. (8 M)

Ans:

```
#include <stdio.h>
void main()
{
    float x,sum,no_row;
    int i,n;
    printf("Input the value of x :");
    scanf("%f",&x);
    printf("Input number of terms : ");
    scanf("%d",&n);
    sum =1; no_row = 1;
    for (i=1;i<n;i++)
    {
     no_row = no_row*x/(float)i;
     sum =sum+ no_row;
    }
    printf("\nThe sum  is : %f\n",sum);
}
```

Output:

```
Input the value of x :3
Input number of terms : 5
The sum  is : 16.375000
```

Q.3)

a) What is an array? What does an array name signify? Can array index be negative? Write a C program to arrange the number stored in an array in such a way that array will have the odd numbers followed by even numbers. (10 M)

Ans:

- An array is a data structure that contains a group of elements. Typically these elements are all of the same data type, such as an integer or string. Arrays are commonly used in computer programs to organize data so that a related set of values can be easily sorted or searched.
- Yes, array index can be negative. It can be used to print the array in a reverse order.

Program:

```
#include <conio.h>

 int main()
{
   int a[10000],b[10000],i,n,j,k,temp,c=0;

   printf("Enter size of the  array : ");
   scanf("%d", &n);
   printf("\nEnter elements in array : ");
   for(i=0; i<n; i++)
   {
      scanf("%d",&a[i]);
      if(a[i]%2==1)
       c++;
   }
   for(i=0; i<n-1; i++)
   {

      for(j=0; j<n-i-1; j++)
      {
        if(a[j]>a[j+1])
        {
              temp=a[j];
              a[j]=a[j+1];
              a[j+1]=temp;
                }

      }
```

```
  }

    k=0;
  j=n-c;

    for(i=0; i<n; i++)
  {
     if(a[i]%2==0)
     {
              if(k<n-c)
                b[k++]=a[i];
              }
              else
              {
                       if(j<n)
                b[j++]=a[i];
       }
  }

  printf("\narray after sorting even and odd elements separately:\n ");

  for(i=0; i<n; i++)
  {
    a[i]=b[i];
    printf("%d ",a[i]);
  }
}
```

Output:

Enter size of the array : 9
Enter elements in array : 1 3 5 7 9 2 4 6 8
array after sorting even and odd elements separately:
1 2 3 4 5 6 7 8

b) Write a program that accepts a word from the user and prints in the following way. For ex. If the word is "STUDY" the program will print it as

S
ST
STU
STUD
STUDY **(10 M)**

Ans:

```
#include <stdio.h>
#include <string.h>
int main()
{
 char* a="STUDY";
 int i,j;
 for(i=0; i<strlen(a); i++)
 {
  for(j=0; j<=i; j++)
  {
   printf("%c",a[j]);
  }
  printf(" ");
}
```

Output:

```
S
ST
STU
STUD
STUDY
```

Q.4)

a) What is string? Explain the use of gets()? Write a c program that will read a word and rewrite it in alphabetical order. For ex. If the word is "matrix" the program should print "aimrtx". (10 M)

Ans:

- Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.
- **gets():**
- Gets function is used to scan a line of text from a standard input device.
- This function will be terminated by a newline character. The nwline character won't be included as part of the string. The string may include white space characters.
- **Syntax :**

  char *gets(char *s);
- This function is declared in the header file stdio.h. It takes a single argument.
- The argument must be a data item representing a string. On successful completion, shall return a pointer to string s.

```
#include<stdio.h>
 #include<conio.h>
 int main()
 {
     char str[100],temp;
     int i,j;
     clrscr();
     printf("Enter the string :");
     gets(str);
     printf("%s in ascending order is -> ",str);
     for(i=0;str[i];i++)
     {
         for(j=i+1;str[j];j++)
         {
            if(str[j]<str[i])
            {
                temp=str[j];
                str[j]=str[i];
                str[i]=temp;
            }
         }
     }
     printf("%s\n",str);
     getch();
```

```
    return 0;
}
```

Output:
Enter the String: matrix
matrix in an ascending order is -> aimrtx

---

b) Explain recursion and its advantages? Write a recursive c program to find the factorial of a given number. (10 M)

Ans:

- Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.
- Advantages:
    a. Reduce unnecessary calling of functions.
    b. Through Recursion one can solve problems in easy way while its iterative solution is very big and complex.
    c. Extremely useful when applying the same solution.

Program:

```
#include<stdio.h>
long int multiplyNumbers(int n);
int main() {
   int n;
   printf("Enter a positive integer: ");
   scanf("%d",&n);
   printf("Factorial of %d = %ld", n, multiplyNumbers(n));
   return 0;
}
long int multiplyNumbers(int n) {
   if (n>=1)
      return n*multiplyNumbers(n-1);
   else
      return 1;
}
```

Output:
Enter a positive integer: 6
Factorial of 6 = 720

Q.5)

a) Explain the storage classes with example. (10 M)

Ans:

- The different locations in the computer where we can store data and their accessibility, initial values etc. very based on the way they are declared. These different ways are termed as different storage classes.
- In C there are for storage classes, namely
  1. Automatic
  2. Register
  3. Static
  4. External or global
- Let us see these storage classes one by one

1. Automatic storage class

In this case data is stored in memory
The initial value of such a variable is garbage
The scope of the variable is local i.e. limited to the function in which it is defined.
The life of such variables is till the control remains in the particular function where it is defined.
For e.g.:
Int i; or auto int i;

2. Register storage class

In this case data is stored in CPU register
The initial value of such a variable is garbage.
The scope of the variable is local i.e. limited to the function in which it is defined
The life of such variables is till the control remains in the particular function where it is defined.
For e.g.:
Register int I;
In this case the data is stored in a small memory inside the processor called its registers.
The advantage of such storage class is that since the data is in the processor itself, its access and operation on such data is faster.
There is limitation on the size of the data that can declared to be register storage class. The data should be such that it doesn't require more than 4 bytes. Hence double and long double data types cannot be declared as a register.
Also there is a limitation on the maximum number of variables in a function that can be a register class. The limitation is that a maximum of 3 register class variable can be declared in a function.

3. Static storage class

In this case data is stored in a memory

The initial value of such a variable is zero
The scope of the variable is local i.e. limited to the function in which it is defined
The life of such variable is till the program is alive.
For e.g.:
Static int I;
If a variable is declared static, its value remains unchanged even If the function execution is completed.
When the execution to that function returns, the previous value is retained.
Thus it says the initialization is only once. If you have an initialization statement of a static member, it will be executed only once i.e. for the first time when this function is called.

4. **External or global storage class**
   In this case data is stored in memory
   The initial value of such a variable is zero.
   The scope of the variable is global i.e. it is accessible from anywhere in the program.
   The life such a variable is till the program is alive.

---

b) Declare a structure to store the information of 10 cricketers.

i. Cricketer name
ii. Matches Played
iii. Runs Scored
iv. Strike Rate

Use a function to display the cricketer information having the maximum strike rate. (10 M)

Ans:

Program:

```
#include<stdio.h>
#include<conio.h>
struct Cricketer
{
 name[50];
 int matches;
 int runs;
 float strike;
void main()
{
 struct Cricketer e[100];
 int n,i,maximum = 0;
```

```
 clrscr();
 printf("Enter the number of Cricketers");
 scanf("%d",&n);
 for(i=0;i<=n-1;i++)
 {
  printf("\nEnter name,matches,runs, strike");
  scanf("%s%d%d%d",&e[i].name,&e[i].matches,&e[i].runs,&e[i].strike);
 }
  printf("\nName\t\tMatches\t\tRuns\t\tStrike\n");
  printf("-------------------------------------------------------------------------------");
  for(i=0;i<=n-1;i++)
  {
   printf("%s%d%d%d",&e[i].name,&e[i].matches,&e[i].runs,&e[i].strike);
  }
Maximum = e[i].strike;
For(i=0;i<=n-1;i++)
{
if(e[maximum].strike < e[i].strike)
      highest = i;
}
printf(“/nThe maximum strike rate is %d”,e[maximum].strike);
  getch();
}
```

Output:

```
Enter number of Cricketers 10
Enter Name, matches, runs, strike
John
100
20000
35.5

Enter Name, matches, runs, strike
sam
101
20200
37.5

Enter Name, matches, runs, strike
Miller
105
20080
38.5
```

Enter Name, matches, runs, strike
Dale
107
20400
36.5

Enter Name, matches, runs, strike
Smith
105
20700
39.5

Enter Name, matches, runs, strike
david
106
28000
40.5

Enter Name, matches, runs, strike
Dhoni
110
20500
42.5

Enter Name, matches, runs, strike
Johny
107
20600
35.5

Enter Name, matches, runs, strike
kohli
120
20000
36.5

Enter Name, matches, runs, strike
Rohit
103
20700
36.5

Name	Matches	Runs	Strike
John	100	20000	35.5
Sam	101	20200	37.5
Miller	105	20080	38.5
Dale	107	20400	36.5
Smith	105	20700	39.5
David	106	28000	40.5
Dhoni	110	20500	42.5
Johny	107	20600	35.5
kohli	120	20000	36.5
Rohit	103	20700	36.5

The maximum strike rate is 42.5

Q.6)

a) How do pointers differ from variable in C? Write a c program to add two pointers. (10 M)

Ans:

- Pointers are variables that are used to store the address of another variable.
- Address of a variable is the memory location number which is allotted to the variable.
  The memory addresses are 0, 1, 2, 3… and so on up to the capacity of the memory. The address is normally displayed in hexadecimal form. Hexadecimal form is a representation of number somewhat similar to binary number. Here four binary digits are combined together to form a hexadecimal number.
- Pointers unlike other variables do not store values. As stated they store the address of other variables.
- It is already mentioned in the first statement that pointers are also variables. Hence, we can also have a pointer that is pointing to another pointer.
- Syntax of pointer declaration : Data_type *ptr_name;
- Wherein “Data_type” is the data type of the variable to which the pointer is supposed to point. If we want a pointer to point to an integer than, we need to have the data type of the pointer as “int”, for a float type data pointer should also be of the “float” type and so on.
- The “ptr_name” is an identifier i.e. the name of the pointer. The same rules of identifiers apply to the pointer name as to any other variable declaration. The most important difference in the declaration of a pointer is the “*” sign given before the pointer name.
- Hence, according to the syntax seen above, if we want to declare a pointer for “int” type data then we can declare it as given in the example below: int *p; Here,

the pointer name is “p”. Hence, “p” can be used as a pointer to point to any of the variable of type “int”.

- Syntax : data_type *var_name;
- Example : int *p; char *p;

Program:

```
#include <stdio.h>
int main()
{
  int first, second, *p, *q, sum;

printf("Enter two integers to add\n");
scanf("%d%d", &first, &second);

p = &first;
q = &second;

sum = *p + *q;

printf("Sum of the numbers = %d\n", sum);

return 0;
}
```

Output:
Enter two integers to add
4
5
Sum of entered number is 9

---

b) What is file? Write a c program that include the menu that must have the following capabilities

i. Enter the several lines of text and store them in data file.

ii. Retrieve and display the particular line

iii. Delete n lines (10 M)

Ans:

- For file handling or accessing the contents of file, there are certain predefined functions available in the C programming language.

- An important thing required to access files is the "FILE pointer". This pointer is used to point to the values stored in the file. A file pointer is hence to be created for accessing the files. The syntax for creating a file pointer is as given below: FILE *<identifier for pointer>; For e.g. FILE *fp;
- Hence in every program we write in this section to access files, we will use this kind of pointer declaration. This pointer is used to point the data to be accessed in the file i.e. whenever a data is read or written in the file, it is from the location pointed by the file pointer "fp".

Program:

```
#include <stdio.h>
int main ()
{
 FILE * fptr;
 int i,n;
 char str[100];
 char fname[20]="test.txt";
 char str1;
                printf(" Input the number of lines to be written : ");
                scanf("%d", &n);
                printf("\n :: The lines are ::\n");
                fptr = fopen (fname,"w");
                for(i = 0; i < n+1;i++)
                    {
                    fgets(str, sizeof str, stdin);
                    fputs(str, fptr);
                    }
 fclose (fptr);
/*-------------- read the file -------------------------------------*/
                fptr = fopen (fname, "r");
                printf("\n The content of the file %s is  :\n",fname);
                str1 = fgetc(fptr);
                while (str1 != EOF)
                    {
                            printf ("%c", str1);
                            str1 = fgetc(fptr);
                    }
   printf("\n\n");
   fclose (fptr);
/*-------------------------------------------------------------------------------*/
     FILE *fp1, *fp2;
      char filename[40];
      char c;
      int del_line, temp = 1;
```

```
 printf("Enter file name: ");
 scanf("%s", filename);
 fp1 = fopen(filename, "r");
 c = getc(fp1);
 while (c != EOF)
 {
  printf("%c", c);
  c = getc(fp1);
 }
 //rewind
 rewind(fp1);
 printf(" \n Enter line number of the line to be deleted:");
 scanf("%d", &del_line);
 fp2 = fopen("copy.c", "w");
 c = getc(fp1);
 while (c != EOF) {
  c = getc(fp1);
  if (c == '\n')
  temp++;
  if (temp != del_line)
  {

    putc(c, fp2);
  }
 }
 fclose(fp1);
 fclose(fp2);

 remove(filename);
 rename("copy.c", filename);
 printf("\n The contents of file after being modified are as  follows:\n");
 fp1 = fopen(filename, "r");
 c = getc(fp1);
 while (c != EOF) {
    printf("%c", c);
    c = getc(fp1);
 }
 fclose(fp1);
 return 0;
}
```

Output:

```
Input the number of lines to be written : 4
 :: The lines are ::
test line 1
test line 2
test line 3
test line 4
 The content of the file test.txt is  :
test line 1
test line 2
test line 3
test line 4
Enter line number of the line to be deleted: 2
The content of the file after being modified are as follows:
test line 3
test line 4
```