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
```
