Computer Assignment

2074 Ashwin(Back)

1.What are computer programs and computer programming? Explain the steps that are required to build a computer program for solving a certain problem.

Ans: The set of instructions given to a computer which makes the computer perform a wide range of tasks are called computer programs. The set of computer programs is called computer software.

The process of writing code to facilitate specific actions in a computer, application or software program, which instructs them on how to perform any task is called computer programming.

The steps that are required to build a computer program for solving a certain program are given below:

1. Problem Analysis
2. Algorithm development
3. Flowchart development
4. Coding
5. Compilation and Execution
6. Debugging and Testing
7. Documentation

2. Explain with an example the role that precedence and associativity play in the execution of an expression. Rewrite the following program by correcting any errors, if present and also write down the output of the corrected code.

Ans: The data type and the value of an expression depends on the data types of the operands and the order of evaluation of operators which is determined by the precedence and associativity of operators. When expressions contain more than one operator, the order in which the operators are evaluated depends on their precedence levels. A higher precedence operator is evaluated before a lower precedence operator. If the precedence levels of operators are the same, then the order of evaluation depends on their associativity (or grouping).

For example:

```
int x=2, y=3, z=4, r;
```

```
r=x-y*z;
```

In this case, both – and * are arithmetic operators but as the priority of * is higher than that of – so first multiplication is carried out then the product is subtracted from x. This ordering of operators of different priority illustrates the operator precedence.

Similarly, for associativity:

Example:

int x=2, y=3, z=4, r;

r=x-y+z;

Here, both – and + have same priority so in this case, C lets us to operate either in left to right or right to left order. For this, the order is from left to right and operation is performed accordingly. This ordering of operation of same priority illustrates operator associativity.

2.Corrected Code

```
#include<stdio.h>
#define MAX 5
int main()
{

int n[MAX]={2,3,5,4,10},i,sum=0;
for (i=0;i<MAX;i+=1)
{
printf("case %d=%3.2d\n",i,n[i]);
sum+=n[i];

}
float average=sum/MAX;
printf("%5.2f",average);
return 0;
}
```

OUTPUT:

```
case 0= 02

case 1= 03

case 2= 05

case 3= 04

case 4= 10

4.00
```

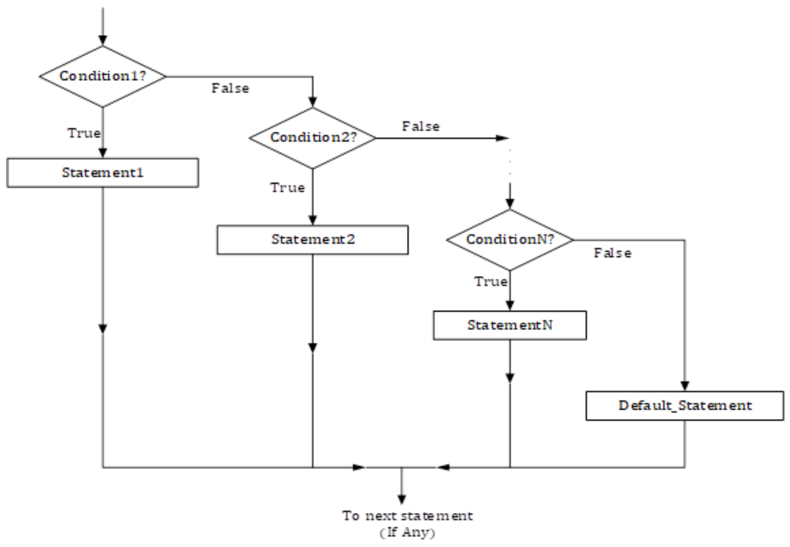
3.a. Compare if-else-if ladder and switch construct with example and flowchart.

Ans: If-else-if is another way of putting **ifs** together when multipath decisions are involved. A multipath decision is a chain of **ifs** in which the statement associated with each **else** is an **if**.

Syntax:

```
if(condition-1)
        statement-1;
else if(condition-2)
        statement-2;
        else if(condition-3)
                    statement-3;
                    ..........................
                    else if(condition-n)
                           statement-n;
                            else
                             default-statement;
 statement-x;
```

Flowchart:



Example:

WAP to identify whether a given number is positive, negative or zero.

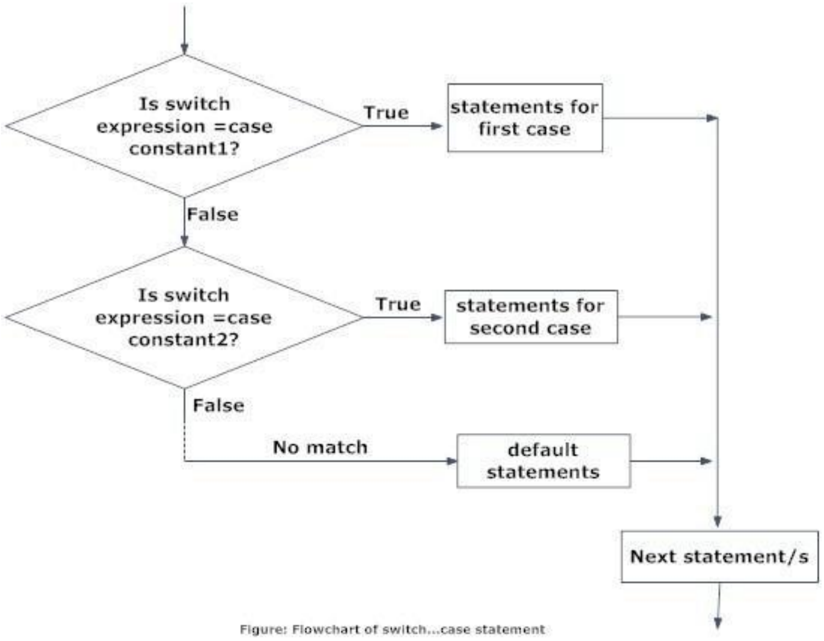
```
#include <stdio.h>
int main()
{
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);

   if (num > 0) {
      printf("The number is positive.\n");
   } else if (num < 0) {
        printf("The number is negative.\n");
    } else {
        printf("The number is zero.\n");
    }
    return 0;
}
```

Similarly, when the number of alternative increases, the complexity of program also increases, so there is a multiway decision statement known as switch which tests the value of an expression against a list of case values (integer or character constants) and when the match is found, the statements associated with that case is executed.

Syntax:

```
switch (expression)

{

            case constant1:

                      block of case constant1;

                      break;

             case constant2:

                        block of case constant2;

                        break;

               case constant3:

                         block of case constant3;

                         break;

                      ................................................

                      ................................................

                     default:

                          default block;}

next statement(s);
```

Flowchart:


Is switch expression =case constant1?
True
statements for first case
False
Is switch expression =case constant2?
True
statements for second case
False
No match
default statements
Next statement/s
Figure: Flowchart of switch...case statement

Example:

WAP that asks an arithmetic operator and two operands and performs the corresponding operation on the operands.

```
#include <stdio.h>
int main() {
   char operator;
   float operand1, operand2, result;
   printf("Enter an arithmetic operator (+, -, *, /): ");
   scanf(" %c", &operator);
   printf("Enter the first operand: ");
   scanf("%f", &operand1);
   printf("Enter the second operand: ");
   scanf("%f", &operand2);

   switch (operator) {
      case '+':
         result = operand1 + operand2;
         break;
      case '-':
         result = operand1 - operand2;
         break;
      case '*':
         result = operand1 * operand2;
         break;
      case '/':
         result = operand1 / operand2;
         break;
   }
   printf("Result: %.2f %c %.2f = %.2f\n", operand1, operator, operand2, result);

   return 0;
}
```

3.b. Write a program to generate the following pattern using unformatted input/output functions only.

```
        N
      e e e
    p p p p p
  a a a a a a a
L L L L L L L L L
```

Ans:

Code:

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
int i,j,k,l;
char s[]="nepal";

l=strlen(s);

for(i=0;i<l;i++)
{
for(j=l;j>i;j--)
{
putchar(' ');
}
for(k=0;k<2*i+1;k++)
{


if(i==0 || i==4)
{

putchar(toupper(s[i] ));
}
else{

putchar(s[i]);}
putchar(' ');
}
putchar('\n');
}
return 0;
}
```

OUTPUT:

```
        N
      e e e
    p p p p p
  a a a a a a a
L L L L L L L L L
```

4. Write a program in C to find out whether the nth term of the Fibonacci series is a prime number or not. Read the value of n from the user and display the result in the main function. Use separate user defined functions to generate the nth Fibonacci term and to check whether a number is prime or not.

Ans:

Code:

```c
#include<stdio.h>
int fibo(int n)
{
    if(n==1)
    return 0;
    if(n==2)
    return 1;
    if(n>=2)
    {
        return fibo(n-1)+fibo(n-2);

     }  }
int prime(int a)
{
  if (a<=1)
  return 0;
  int count=0;
  for(int i=1;i<=a;i++)
  {
    {if(a%i==0)
    count++;}
  }
  if (count==2)
  return 1;

}
int main()
{
    int n,res;
    printf("enter the value of n:");
    scanf("%d",&n);
```

```
   res=prime(fibo(n));
   if(res==0)
   { printf("Not Prime");}
   if (res==1)
   { printf("Prime");}
   return 0;
}
```

Output:

enter the value of n:4

Prime

5 a) How two-dimensional arrays are created in C programming? Write a program to read square matrix of size NXN and find sum of both diagonals.

Ans: Two dimensional arrays can be seen as array of arrays. It is organized in two directions: horizontal and vertical. The data items arranged in horizontal direction are referred as rows and the data items arranged in vertical direction are referred as columns. The matrices those we study in mathematics are two-dimensional arrays. A pair of indices representing the position of the element in the array can be used to access each element of the array.

It is declared as:

data_type array_name [row_size] [column_size];

They are created in C programming as shown below:

```
#include<stdio.h>
int main()
{
 int ar[3][3],i, j;
for(i=0;i<3;i++)
{
 for(j=0;j<3;j++)
{
 printf("Enter values\n");
 scanf("%d",&ar[i][j]);
}
}
printf("Printing the elements of array\n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
 printf("%d\t",ar[i][j]);
 }
 }
 return 0;
 }
```

// A program to read square matrix of size NXN and find sum of both diagonals. //

```
#include<stdio.h>
int main()
{
   int sd1,sd2,N;
   printf("Enter the size of the matrix");
   scanf("%d",&N);
   int a[N][N];
   for(int i=0;i<N;i++)
   {
      for(int j=0;j<N;j++)
      {
      printf("Enter the elements of  matrix");
      scanf("%d",&a[i][j]);
      }
   }
   sd1=0;
   sd2=0;
   for(int i=0;i<N;i++)
   {
      for(int j=0;j<N;j++)
      {
         if (i==j)
         {
          sd1=sd1+a[i][j];
         }
         if (i+j==N-1)
         {
         sd2=sd2+a[i][j];
         }
      }
   }
printf("The sum of first doagonal is %d\n",sd1);
printf("The sum of second diagonal is %d\n",sd2);
return 0;

}
```

Q.No.5.b.
Write a program in C to check whether a given string is palindrome or not using user defined function.

```
#include <stdio.h>
#include <string.h>
void palindrome(char s[])
{
    int l = strlen(s);
    int i, c;
  for (i = 0; i < l; i++)
    {
        if (s[i] == s[l - i - 1])
        {
            c++;
        }
    }
    if (c == i)
    {
        printf("the string is palindrome");
    }
    else
    {
        printf("the string isnot palindrome");
    }
}

int main()
{
    char s[100];
    printf("enter any string");
    gets(s);
    palindrome(s);
}
```

Output:

```
enter any stringmadam
the string is palindrome
```

Q.No.6 What are the advantages of using pointers in C programming? Write a program in C to find the second largest elements from an array containing N elements using the concept of pointer.

A fundamental idea in the C programming language, pointers come with a number of benefits that make them strong and practical tools. Some of the main benefits of using pointers in C are as follows:

Dynamic Memory Allocation: Pointers enable the use of operations like malloc, calloc, and realloc to dynamically allocate memory while a program is running. This makes it possible to effectively manage memory and generate data structures of various sizes.

Passing by Reference: If you feed a function a pointer, it allows you to change the original data right there in the code. This is a useful method for passing huge data structures to functions without having to replicate the full structure beforehand.

```
#include<stdio.h>
int main(){
    int num[10];
    int n,temp;
    int *p;
    printf("enter the number of elements you want");
    scanf("%d",&n);
for(int i=0;i<n;i++){
   printf("enter the element %d\t",i+1);
   scanf("%d",&num[i]);
}
for(int i=0;i<n;i++){
   for(int j=i+1;j<n;j++){
if(*(num+i)<(*(num+j))){
   temp=num[i];
   num[i]=num[j];
   num[j]=temp;
}
}
}
printf("the second largest array element is %d",num[1]);

}
```

Output:

```
enter the number of elements you want5
enter the element 1     45
enter the element 2     43
enter the element 3     23
enter the element 4     56
enter the element 5     76
the second largest array element is 56
```

Q.No.7.Explain structure and nested structures. Create a structure to hold any complex number x+iy. Write a program that used the structure to read two complex numbers and display the third complex number which is the multiplication of the entered complex numbers.

In C programming, a structure is a composite data type that allows you to group together variables of different data types under a single name. Each variable within a structure is referred to as a "member".Structures provide a way to create more organized and meaningful data structures for representing complex entities.

Syntax:

```
Struct  struct_name {
 datatype_1 member1;
Datatype_1 member2;
};
```

A nested structure is a structure that is a member of another structure. This allows you to create more complex data structures that represent relationships between entities.

```
struct Address {
   char street[50];
   char city[30];
   char state[20];
};

struct Employee {
   char name[50];
   int id;
   struct Address address; { Nested structure}
   float salary;
}
```

Program:

```
#include<stdio.h>
struct complex{
    int real;
    int im;
};
int main(){
    struct complex c1;
    struct complex c2;
    struct complex r3;
    printf("enter the real and imaginary part of first complex number");
    scanf("%d%d",&c1.real,&c1.im);
    printf("enter the real and imaginary part if second complex number");
    scanf("%d%d",&c2.real,&c2.im);
    r3.real=(c1.real*c2.real)-(c1.im*c2.im);
    r3.im=(c1.real*c2.im)+(c2.real*c1.im);
    printf("the multiplication of two complex numbers is %d + %d i",r3.real,r3.im);
    }
```

Output:

```
enter the real and imaginary part of first complex number2
3
enter the real and imaginary part if second complex number4
5
the multiplication of two complex numbers is -7 + 22 i
```

Q.No.8.a What are the different input/output functions used with data files in C?.Explain with syntax and examples.

In C, there are several input/output functions used to work with data files. These functions are part of the <stdio.h> library and allow you to read from and write to files.

1.fopen(): This function is used to open any existing file or create a new one,

ptr = fopen("fileopen","mode");

2. fscanf(): This function reads a certain set of data from the existing file.

fscanf(FILE*fptr, “format specifiers”,“argument”)

3.fprintf(): Thr fprintf function is used to write set of characters into file.

It sends formatted output to a steam.

Fprintf(FILE*fptr, “format specifiers”,“argument”)

4.fclose():This function closes the file.

5.getc():This function reads a character from a file.

6.putc():This function writes a character to a file.

Q.No.8.b Write a program,, in C to read integers from the user until the user says “no”. After reading the data write all the odd numbers to a file name odd.txt and all the even numbers to file named even.txt.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
   FILE *fe;
   FILE *fo;
   int num;
   char s[5];
   fo = fopen("odd.txt", "   w+");
   if (fo == NULL)
   {
      printf("EOF");
   }
   fe = fopen("even.txt", "w+");
   if (fe == NULL)
   {
      printf("EOF");
   }
   do
   {
      printf("enter your number\t");
      scanf("%d", &num);
      if (num % 2 != 0)
      {
         fprintf(fo, "%d\n", num);
      }
      else
      {
         fprintf(fe, "%d\n", num);
      }
      printf("Do you want to write more ?if not type no\t");
      fflush(stdin);
      gets(s);
   } while (strcmp(s, "no") != 0);
   fclose(fe);
   fclose(fo);
}
```

Output:

```
enter your number       23
Do you want to write more ?if not type no       y
enter your number       45
Do you want to write more ?if not type no       y
```

```
enter your number       33
Do you want to write more ?if not type no       y
enter your number       78
Do you want to write more ?if not type no       y
enter your number       80
Do you want to write more ?if not type no       no
```

Q.No.9 When can you use recursive functions? Why do we need control statements in computer programs? Differentiate between do. while and for statements.

Recursion is solving a problem with a function that calls itself. It helps you break down bit problems into smaller ones. Often, the recursive solution can be simpler to read than the iterative one. Recursion can be used to program different problems such as Fibonacci series, factorial, sum to natural numbers etc. In all these problems, common process repeated tasks is done which can be achieved by recursion.

Control statements in C help the computer execute a certain logical statement and decide whether to enable the control of the flow through a certain set of statements or not. Also, it is used to direct the execution of statements under certain conditions. The various types of control statements are:

Decision making control statements

1.if statement
2. If-else statements
3. Nested if-else statements
4. else-if ladder

- Conditional statements
- Goto statements in C
- Loop control statements in C

1. While Loop
2. Do-while Loop
3. For Loop

For loop	**Do while loop**
Syntax: For(initialization; condition; increament), {. Statements;}	Syntax: Do { . Statements; } While(condition);
It is an entry controlled loop.	It is an exit controlled loop.
Initializing and updating is the part of the syntax.	Initializing and updating isn’t the part of the syntax.
If the conditions don’t match, the control will never enter the loop.	Even if the condition is true for the first time the control will enter in a loop.

Q.No.10 What are the characteristics of FORTAN programming ? Write a program in FORTAN programming ro calculate the value of π by evaluating the following for the first 25 terms.

Fortran (short for Formula Translation) is a programming language that was initially designed for scientific and engineering computations. It has evolved over the years, and different versions of the language have been released. Here are some key characteristics of Fortran programming

Numerical Computation: Fortran was designed with a focus on numerical and scientific computing. It provides extensive support for arithmetic operations, mathematical functions, and array manipulation.
Strongly Typed: Fortran is a strongly typed language, which means that variables must be declared with specific data types before they can be used. This ensures type safety and reduces the likelihood of runtime errors.
Array Operations: Fortran has built-in support for arrays and provides powerful array operations, making it well-suited for tasks involving large datasets or matrix computations.
Procedural: Fortran is a procedural programming language, meaning that programs are structured as a series of procedures (subroutines and functions) that define the steps to be executed.
CODE:
```
program pi
implicit none
 real :: p,sign,term
 integer :: n
 sign =1.0
p=0.0
 do n= 1,25
 term =sign/(2*n-1)
 p=p+term
 sign=-sign
 end do
 p=4*p
 write(*,*) p
 stop
 end
```