TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

EXAMINATION CONTROL DIVISION

2067 ASHADH

1) What is high level language? What are the different types of high level languages? How computer programing language C is different from FORTAN?

ANS:

High Level programming language is language that is more user friendly, to some extent platform-independent and abstract from low-level computer processor operations such as memory access.

The different types of high level languages are:

1. Procedure Oriented Language
2. Object Oriented languages
3. Problem Oriented Languages
4. Natural Languages

Computer programming language C is different from FORTAN in the following ways:

1. Origin and History:

   a. C: Developed in the early 1970s at Bell Labs by Dennis Ritchie, C was designed as a general-purpose programming language to write operating systems and utility programs.

   b. Fortran: Developed in the 1950s by IBM, Fortran (short for "Formula Translation") was one of the earliest high-level programming languages. It was primarily designed for scientific and engineering computations.

2. Syntax and Structure:

   a. C: C has a more compact and flexible syntax. It uses curly braces {} to define blocks of code and relies heavily on semicolons to terminate statements.

   b. Fortran: Fortran has a more traditional syntax. It uses columns to denote statement continuation and indentation is less important.

3. Applications:

   a. C: C is widely used in systems programming, embedded systems, game development, and software development where performance and low-level control are crucial.

   b. Fortran: Fortran is commonly used in scientific and engineering fields, such as numerical simulations, computational fluid dynamics, and weather modeling, where array-based calculations are central.

4. Language Features:

   a. C: C provides features like pointers, manual memory management, and explicit control over hardware, making it a powerful language for system-level programming.

   b. Fortran: Fortran has specialized features for mathematical computations, such as array broadcasting, intrinsic mathematical functions, and parallel processing support.

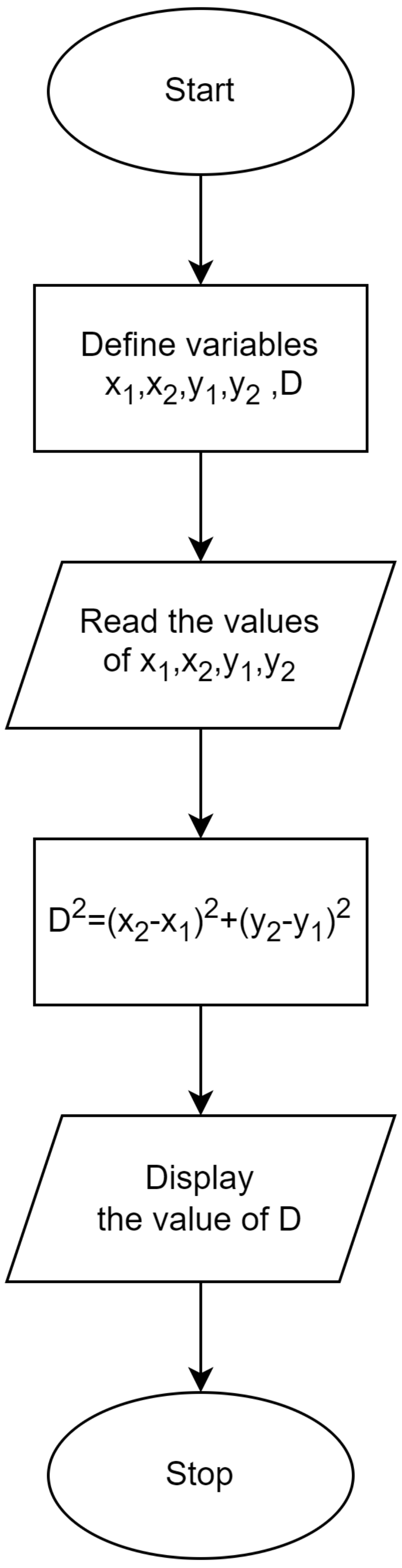
2) Write an algorithm and flowchart of the distance between two points (x1,y1) and (x2,y2),governed by the formula D2=(x2-x1)2+(y2-y1)2.Where, x1,x2,y1,y2are given by the user but should not be zero.

ANS:

ALGORITHM:

1. Start
2. Print "Enter the values of x1,x2,y1,y2 but the values should not be zero"
3. Read the values.
4. Calculate the distance between two points using formula, D2=(x2-x1)2+(y2-y1)2.
5. Display D as a distance between the points.
6. Stop.

FLOWCHART


Start
Define variables x_1,x_2,y_1,y_2 ,D
Read the values of x_1,x_2,y_1,y_2
$D^2=(x_2-x_1)^2+(y_2-y_1)^2$
Display the value of D
Stop

3) Write a syntax used in C programming language for the followings:

a) scanf()

b) while

c) struct

d) if else

e) static

ANS:

SYNTAX:

a) scanf() :

```
scanf("control string", arg1, arg2, ...., argn);
```

b) while :

```
while (condition)
          {
               // Code to be executed while the condition is true
          }
```

c)struct:

```
struct structure_name
          {
            data_type member1;
             data_type member2;
               .......
               .......
            data_ type membern;
          };
```

d) if else:

```
if(condition)
     {
          //code to be executed when condition is true
     }
else
     {
          //code to be executed when condition is false
     }
```

e) static:

```
static data_type variable_name;
```

4) What are the significant meanings of ‘&’ and ‘*’ established in C programming? How can you differentiate between ‘called by value’ and ‘called by reference’ with example in C programming?

Ans:

In C programming, the ‘&’ (ampersand) and ‘*’ (asterisk) symbols have significant meanings related to pointers and memory manipulation:

i. Address-of operator(&):
   - The ‘&’ operator is used to get the memory address of a variable. It returns a pointer to the memory location where the variable is stored.
   - It is often used when working with pointers, functions that require passing memory addresses, and memory management.

ii. Dereference operator(*):
   - The ‘*’ operator is used to access the value stored at memory location pointed by the pointer. This is also known as “dereferencing” the pointer.
   - When applied to a pointer, it retrieves the value stored in the memory location pointed to by the pointer.

Call by reference and call by value are the two methods of passing argument to the function. In call by value, the value of the actual parameter is passed. The original value of passed variable is not modified. The value of argument in the calling function is not changed even if they are changed in called function.

For example:

```
#include<stdio.h>


void swap(int, int);

int main()

{

 int x, y;

 printf("Enter 2 numbers");

 scanf("%d%d",&x,&y);

 printf("Before swapping :x=%d and y=%d",x,y);

 swap(x,y);

 printf("\nAfter swapping :x=%d and y=%d",x,y);

 return0;

}

void swap(inta, intb)

{

 int temp;

 temp = b;

 b=a;

 a=temp;

}
```

Output:

```
Enter 2 numbers

1

2
```

```
Before swapping :x=1 and y=2

After swapping :x=1 and y=2
```

Here, the values of a and b are swapped in the function. However, in the main function, the value of x and y are not swapped. Thus, in call by value method, any change in a or b is not reflected in x and y.

In call by reference, rather than the actual value, the address of variable is passed as argument. Pointers are used to point to the memory location of the variables. Any change in the variable in called function is reflected in the calling function.

For example:

```
#include<stdio.h>

void swap(int *,int *);

int main()

{

 intx, y;

 printf("Enter 2 numbers\n");

 scanf("%d%d",&x,&y);

 printf("Before swapping :x=%d and y=%d",x,y);

 swap(&x,&y);

 printf("\nAfter swapping :x=%d and y=%d",x,y);

 return0;

}

void swap(int*a, int*b)

{

 int temp;

 temp = *b;

 *b=*a;

 *a=temp;

}
```

OUTPUT

```
Enter 2 numbers

1

2

Before swapping :x=1 and y=2

After swapping :x=2 and y=1
```

Here, address of the variable is passed to the called function. Here addresses of x and y are passed to the function swap. These addresses are copied to a and b respectively. Any operation done inside the swap is actually done on the memory location pointed by the address. Thus, swapping done on *p and *q is reflected on x and y.

5) State with example, how switch( ) function differs from user defined function in computer programming language C.

ANS:

We can differentiate switch() function from the user-defined function based on the following category:

Purpose:

- Switch case: Used to make decisions based on the value of an expression.
- User-defined function: Used to encapsulate and modularize a specific piece of functionality for reuse.

Usage:

- Switch case: Primarily used for conditional branching.
- User-defined function: Used for encapsulating tasks into reusable blocks of code.

Control Flow:

- Switch case: Directs program flow based on matching values of an expression.
- User-defined function: Is invoked explicitly by calling its name.

Return Value:

- Switch case: Does not return a value itself; it only executes code based on the matched case.
- User-defined function: Can return a value to the caller using the return statement.

Arguments:

- Switch case: Typically doesn't take arguments; it relies on the value of the expression being evaluated.
- User-defined function: Accepts arguments (parameters) to perform operations on.

6) Write a program to find the standard deviation of an array of the values in C programming.. The array elements are read from the terminal. Use user defined functions to calculate the standard deviation and mean.

ANS:

```
#include<stdio.h>

#include<string.h>

#include<math.h>



float SD(int arr[],float,int);

float m(float,int);

int main()

{

  int n;

  float sum=0,mean;

  printf("Enter how many datas do you want to enter\n");

  scanf("%d",&n);

  int data[n];

  for(int i=0;i<n;i++)

  {
        scanf("%d",&data[i]);

        sum=sum+data[i];

     }

     mean=m(sum,n);

     printf("Mean of the given numbers is %.2f\n",mean);

     printf("The standard deviation of the given numbers is %.2f",SD(data,mean,n));

  }



  float m(float sum,int n)

  {

     return sum/n;

  }



  float SD(int arr[],float mean,int n)

  {

     float sum=0;

     for(int i=0;i<n;i++)

     {

        sum=sum+pow(arr[i]-mean,2);

     }

     return sqrt(sum/n);

  }
```

7)Write a program in C programming language according to the output displayed below:(to open a file named RECORD.txt for the n number of data where cost, service charge 5%, VAT 15% and Total Cost must be calculated by program itself)

Item code	Description	Rate	Quantity	Cost
001CT	Computer	22,000.00	5	110000.00
007M	Cell Phone	8000.00	10	80000.00
VAT 15%				
Service Charge 5%				
Total Cost				

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

```
#include<string.h>


struct find_cost

{

   char i_code[10],item[50];

   int quantity;

   float rate,cost,sum;

};


int main()

{

   int n;

   float sum=0;

   printf("How many data do you want to store\n");

   scanf("%d",&n);

   struct find_cost f[n];

   FILE *fp;

   fp=fopen("record.txt","w+");

   fprintf(fp,"Item Code\tDescription\tRate\tQuantity\tCost\n");

   for(int i=0;i<n;i++)

   {   printf("For item %d\n",i+1);

      printf("Enter the item code\n");

      scanf("%s",f[i].i_code);

      printf("Enter name of the item\n");

      scanf("%s",f[i].item);

      printf("Enter quantity of the item\n");

      scanf("%d",&f[i].quantity);

      printf("Enter rate of the item\n");

      scanf("%f",&f[i].rate);

      f[i].cost=f[i].rate*f[i].quantity;

      sum=sum+f[i].cost;

      fprintf(fp,"%s\t\t%s\t\t%d\t\t%.2f\t\t%.2f\n",f[i].i_code,f[i].item,f[i].quantity,f[i].rate,f[i].cost);

      if(i==(n-1))

      {

         float vat=0.15*sum;

         float service_charge=0.05*sum;

         fprintf(fp,"\nVAT=%.2f",vat);
```

```
            fprintf(fp,"\nService Charge=%.2f",service_charge);

            fprintf(fp,"\nTotal Cost=%.2f",sum+vat+service_charge);



          }

        }

      }
```

8) Rewrite the program correctly and write output of the given program written in C programming language below:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

   char ar1[11]={'I','o','E',',','P','U','L','C','H','O','W','K'};

   char ar2[15]="IoE,PULCHOWK";

   char ar3[11]={

      {'I'},{'o'},{'E'},{','},{'P'},{'U'},{'L'},{'C'},{'H'},{'O'},{'W'},{'K'}};

   printf("Array1=%c\n",ar1);

   printf("Array2=%c\n",ar2);

   printf("Array3=%c\n",ar3);

   return 0;

}
```

ANS:

```
      #include<stdio.h>

      #include<conio.h>

      #include<string.h>

      void main()

      {

         char ar1[]={'I','o','E',',','P','U','L','C','H','O','W','K','\0'};

         char ar2[15]="IoE,PULCHOWK";

         char ar3[][1]={{'I'},{'o'},{'E'},{','},{'P'},{'U'},{'L'},{'C'},{'H'},{'O'},{'W'},{'K'},{'\0'}};

         printf("Array1=%s\n",ar1);

         printf("Array2=%s\n",ar2);

         printf("Array3=%s\n",ar3);

      }
```

9) Describe the formatted input and output statement in FORTAN programming language with it's syntax.

ANS:

In the Fortran programming language, formatted input and output statements are used to read and write data to and from files or the standard input/output devices (usually the screen and keyboard). These statements provide a way to control the appearance and arrangement of data being read from or written to files. The format descriptors in these statements define how data is formatted.

Here's the basic syntax for formatted input and output statements in Fortran:

```
read(UNIT=unit_no,FMT=format_no) list_of_variables_separated_by_comma

write(UNIT=unit_no,FMT=format_no) list_of_variables_separated_by_comma
```

We can specify some particular input or output format. The specification of the types of the data and it's size is called FORMAT specification.

SYNTAX:

```
label format(format_specification1,format_specification2,........)
```

where, lablel is used to identify the correct format specification while reading or writing a particular data. The different format specifications are separated by commas in single format statement.

The most common format letters are:

I- integer,

A- text string,

F- real numbers, fixed point format,

X-horizontal skip(space) e.t.c

In the previous syntax of i/o, we pass the label of the format into the format_no to achieve the required format.

10)write a program in Fortan to evaluate the following series:

series=1/1^2 +1/2^2 +1/3^2+.......+1/n^2

Ans:

```
      program SeriesEvaluation

      real sum=0

      integer i, n

      write(*,*), 'How many terms do you want'

      read(*,*), n

      do i=1,n,1

         sum=sum+1/(i**2)

         end do

      write(*,900), sum

900     format('sum is ',F10.3)

      stop

      end
```