ID:11757

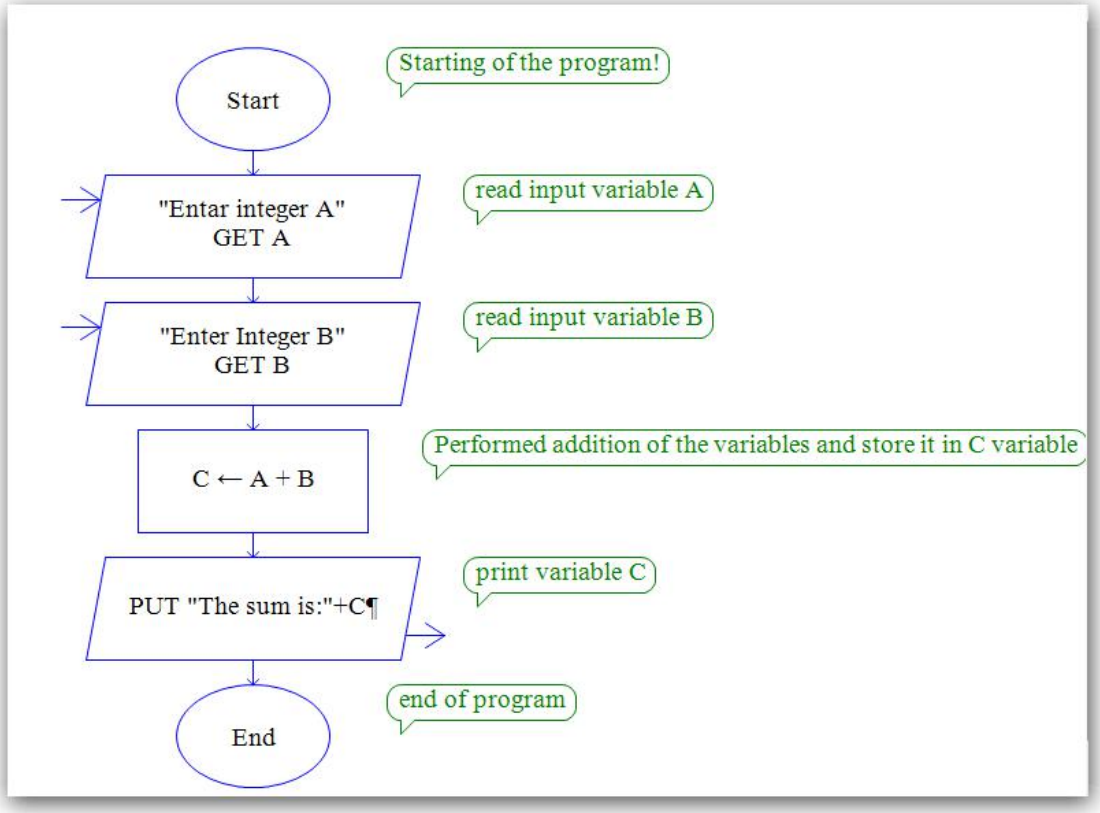
NAME:SALMAN KHAN

SUBJECT:PROGRAMMING FUNDAMENTALS

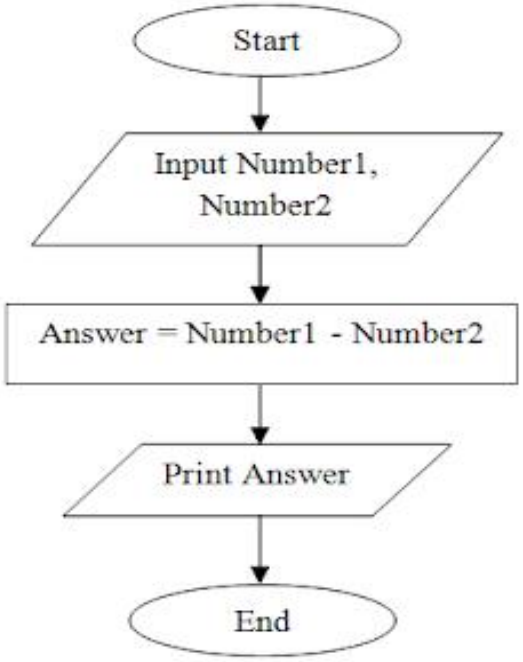
TEACHER: DR. FAZAL-E- MALIK

DATE:25.08.2020

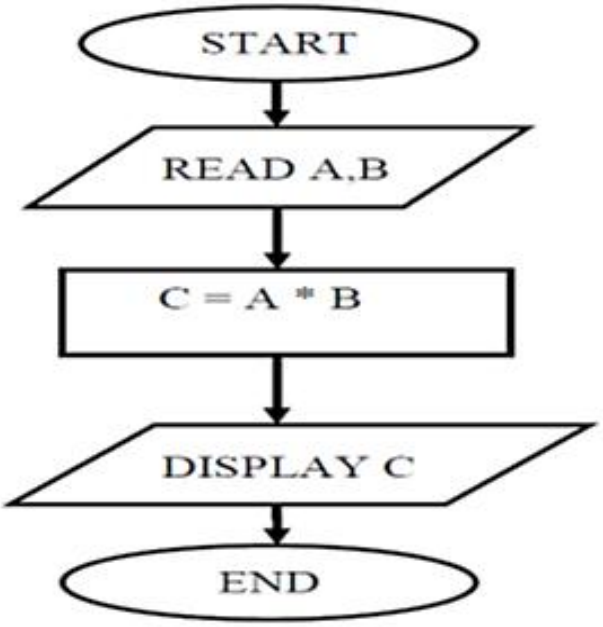
ANS:1 PART(A):SUM OF TWO NUMBER:



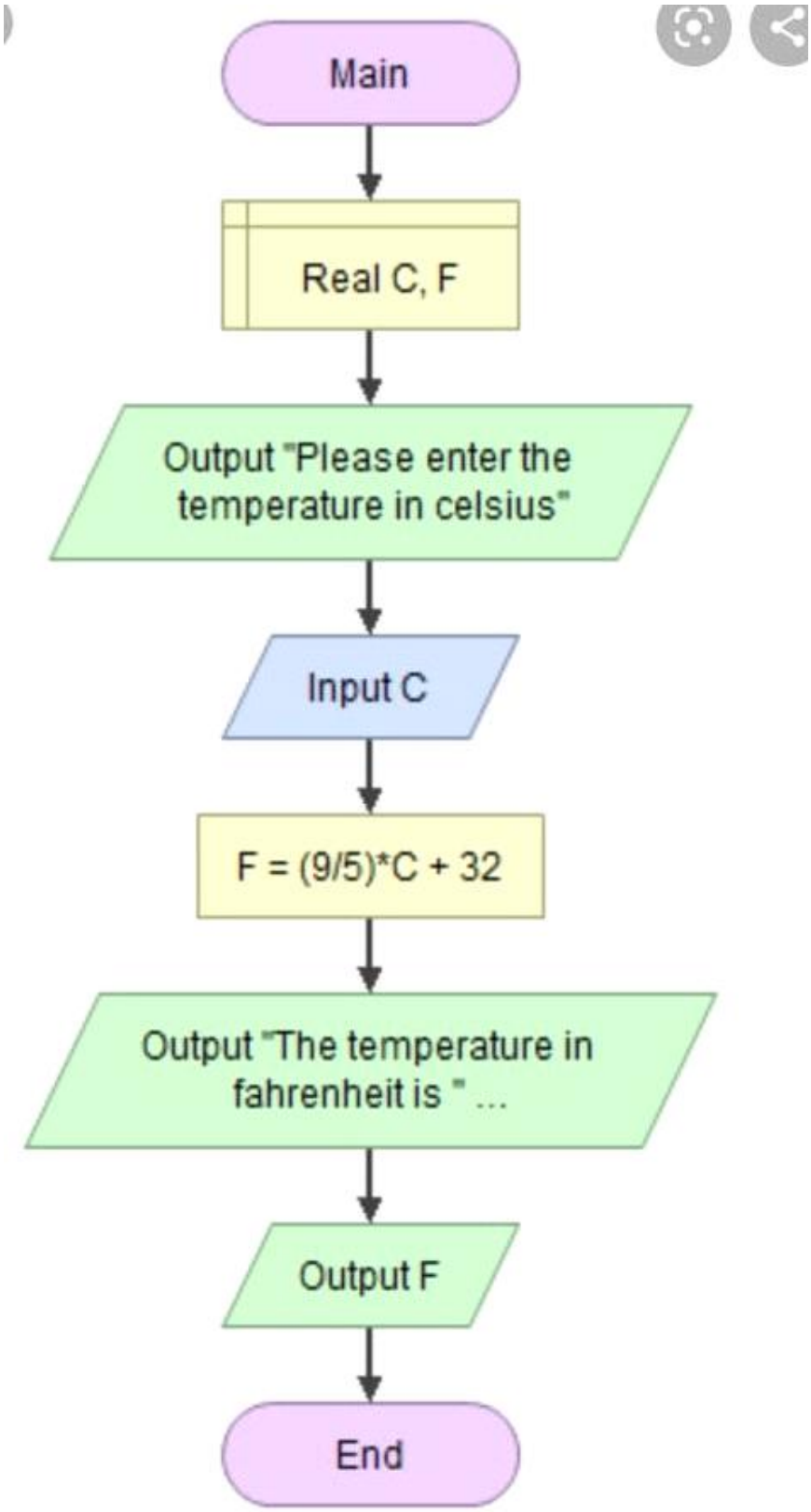
DIFFERENCE OF TWO NUMBERES:



PRODUCT OF TWO NUMBERS:



PART(B):

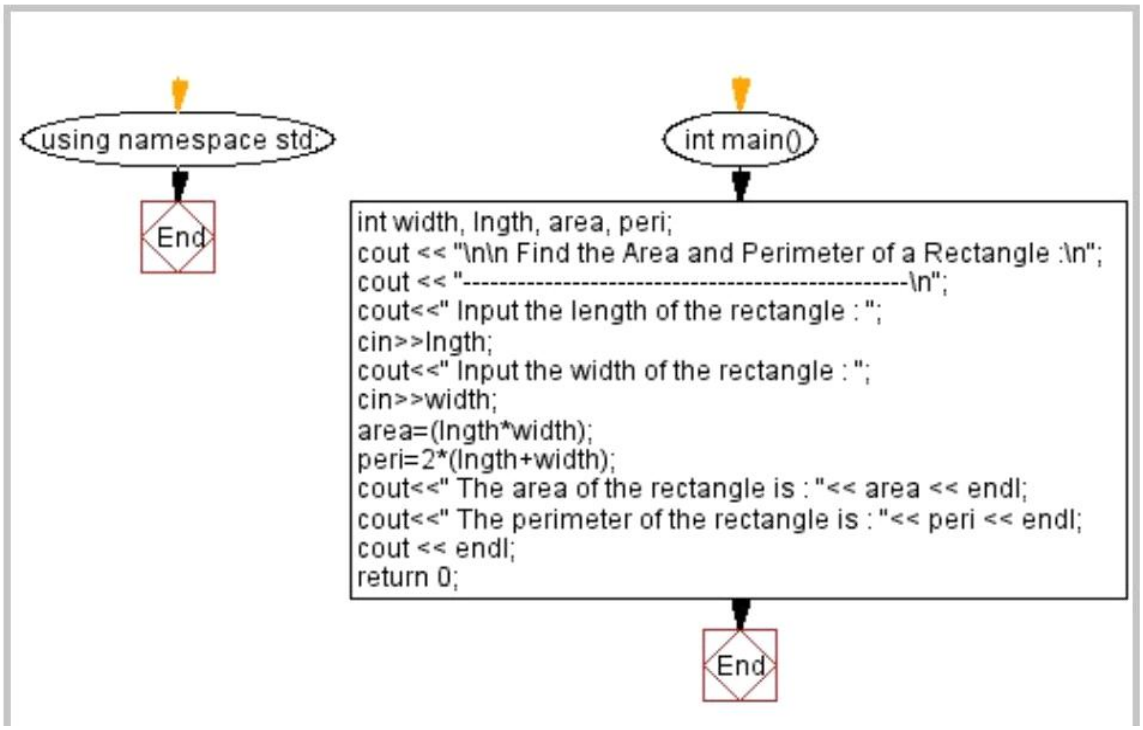

Main
Real C, F
Output "Please enter the temperature in celsius"
Input C
F = (9/5)*C + 32
Output "The temperature in fahrenheit is " ...
Output F
End

ANS:2 PATR(A): #include <stdio.h>

```
/* height and width of a rectangle in inches */
int width;
int height;
int area;
int perimeter;
int main() {
        height = 7;
        width = 5;
      perimeter = 2*(height + width);
        printf("Perimeter of the rectangle = %d inches\n", perimeter);
        area = height * width;
        printf("Area of the rectangle = %d square inches\n", area);
return(0);
```

FLOWCHART:

Flowchart:



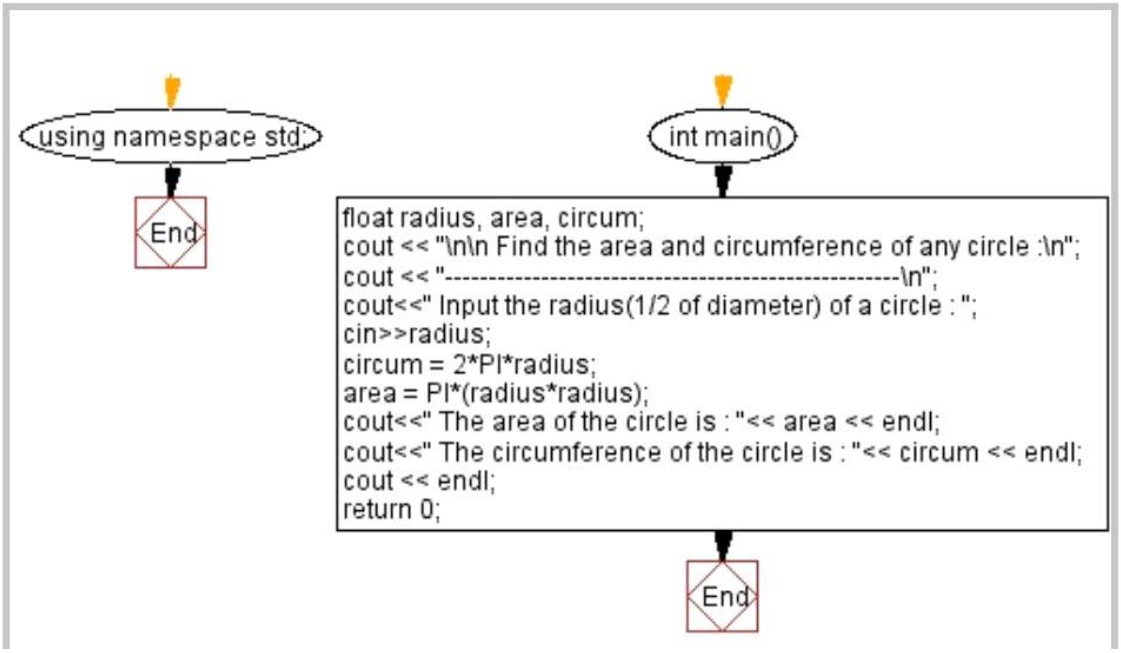
PART(B):

```
#include <iostream>
#define PI 3.14159
using namespace std;
    int main()
    {
       float radius, area, circum;
                cout << "\n\n Find the area and circumference of any circle :\n";
                cout << "-----\n";
          cout<<" Input the radius(1/2 of diameter) of a circle : ";
       cin>>radius;
                circum = 2*PI*radius;
                area = PI*(radius*radius);
```

```
    cout<<" The area of the circle is : "<< area << endl;
    cout<<" The circumference of the circle is : "<< circum << endl;
    cout << endl;
    return 0;
}
```

FLOWCHART:

Flowchart:



ANS:3PART(A):PROGRAMMING LANGUAGE:Programming languages specially developed so that you could pass your data and instructions to the computer to do specific job

There are two major types of programming languages,

Low Level Languages

High Level Languages

Low Level languages are further divided in to Machine language and Assembly language

High Level Languages are, for scientific application FORTRAN and C languages are used. On the other hand COBOL is used for business applications. A programming language is a notation designed to connect instructions to a machine or a computer. Programming languages are mainly used to control the performance of a machine or to express algorithms. At present, thousand programming languages have been implemented. In the computer field, many languages need to be stated in an imperative form, while other programming languages utilize declarative form. The program can be divided into two forms such as syntax and semantics. Some languages are defined by an SO standard like C language.

MACHINE LANGUAGE: Machine Language is the only language that is directly understood by the computer. It does not need any translator program

The only advantage is that program of machine language run very fast

There is nothing "below" machine language – only hardware.

Impossible for humans to read. Consists of only 0's and 1's.

0001001111110000

In the earliest days of computers, the only programming languages available were machine languages. Each computer had its own machine language, which was made of streams of 0s and 1s.

ASSEMBLY LANGUAGE: The next evolution in programming came with the idea of replacing binary code for instruction and addresses with symbols. Because they used symbols, these languages were first known as symbolic languages. The set of these mnemonic languages were later referred to as assembly languages.

It is the first step to improve the programming structure, you should know that computer can handle numbers and letter.

The set of symbols and letters forms the Assembly Language and a translator program is required to translate the Assembly Language to machine language

This translator program used for Assembly Language is called Assembler

To program in assembly you need to understand concepts behind machine language and execution-fetch cycle of CPU.

Assembly is a machine specific language.

Although Assembly and machine language might look similar, they are in fact two different types of languages.

Assembly consists of both binary and simple words

Machine code composed only of 0's and 1's

HIGH LEVEL LANGUAGE: Although assembly languages greatly improved programming efficiency, they still required programmers to concentrate on the hardware they were using. Working with symbolic languages was also very tedious, because each machine instruction had to be individually coded. The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high- level languages.

Assembly and machine level languages require deep knowledge of computer hardware where as in higher language you have to know only the instructions in English words and logic of the problem.

Higher level languages are simple languages that use English and mathematical symbols like +, -, %, / etc. for its program construction

Any higher level language has to be converted to machine language for the computer to understand

For example COBOL (Common Business Oriented Language), FORTRAN (Formula Translation) and BASIC (Beginners All-purpose Symbolic Instruction Code) are high level languages

ADVANTAGE OF HIGH LEVE LANGUAGE: Higher level languages have a major advantage over machine and assembly languages that higher level languages are easy to learn and use (similar to the languages used by us in our day to day life.

TYPES OF PRGRAMING LANGUAGE: Procedural Programming Language:

The procedural programming language is used to execute a sequence of statements which lead to a result. Typically, this type of programming language uses multiple variables, heavy loops and other elements, which separates them from functional programming languages. Functions of procedural language may control variables, other than function's value returns. For example, printing out information.

Functional Programming Language:

Functional programming language typically uses stored data, frequently avoiding loops in favor of recursive functions.The functional programing's primary focus is on the return values of functions, and side effects and different suggests that storing state are powerfully discouraged. For example, in an exceedingly pure useful language, if a function is termed, it's expected that the function not modify or perform any o/p. It may, however, build algorithmic calls and alter the parameters of these calls. Functional

languages are usually easier and build it easier to figure on abstract issues, however, they'll even be "further from the machine" therein their programming model makes it difficult to know precisely, but the code is decoded into machine language (which are often problematic for system programming).

Object-oriented Programming Language:

This programming language views the world as a group of objects that have internal data and external accessing parts of that data. The aim this programming language is to think about the fault by separating it into a collection of objects that offer services which can be used to solve a specific problem. One of the main principle of object oriented programming language is encapsulation that everything an object will need must be inside of the object. This language also emphasizes reusability through inheritance and the capacity to spread current implementations without having to change a great deal of code by using polymorphism.

Scripting Programming Language:

These programming languages are often procedural and may comprise object-oriented language elements, but they fall into their own category as they are normally not full-fledged programming languages with support for development of large systems. For example, they may not have compile-time type checking. Usually, these languages require tiny syntax to get started.

Logic Programming Language

These types of languages let programmers make declarative statements and then allow the machine to reason about the consequences of those statements. In a sense, this language doesn't tell the computer how to do something, but employing restrictions on what it must consider doing.

To call these groups " types of language " is really a bit confusing. It's easy to program in an object-oriented style in C language. In truth, most of the languages include ideas and features from various domains, which only helps to increase the usefulness of these types of languages. Nevertheless, most of the programming languages do not best in all styles of programming.

PART(B): TRANSLATORS: Computers only understand machine code (binary), this is an issue because programmers prefer to use a variety of high and low-level programming languages instead.

To get around the issue, the high-level and low-level program code (source code) needs to pass through a translator.

A translator will convert the source code into machine code (object code).

There are several types of translator programs, each able to perform different tasks.

compiler: compiler are used to translate a program written in a high-level language into machine code (object code).

Once compiled (all in one go), the translated program file can then be directly used by the computer and is independently executable.Compiling may take some time but the translated program can be used again and again without the need for recompilation.

An error report is often produced after the full program has been translated. Errors in the program code may cause a computer to crash. These errors can only be fixed by changing the original source code and compiling the program again.

INTERPRETER: Interpreter programs are able to read, translate and execute one statement at a time from a high-level language program.

The interpreter stops when a line of code is reached that contains an error.

Interpreters are often used during the development of a program. They make debugging easier as each line of code is analysed and checked before execution.Interpreted programs will launch immediately, but your program may run slower then a complied file.

No executable file is produced. The program is interpreted again from scratch every time you launch it.

ASSEMBLER: Assemblers are used to translate a program written in a low-level assembly language into a machine code (object code) file so it can be used and executed by the computer.

Once assembled, the program file can be used again and again without re-assembly.