

**Tribhuvan University “Questions Bank” solution for computer programming
2076**



1.What do you mean by software?Explain about generation of Programming language.Design A flowchart to check whether an input number is even or odd.

Answer Number 1.

Part-1

Software is a set of programs, which is designed to perform a well-defined function. A program is a sequence of instructions written to solve a particular problem.

There are two types of software -

- System Software
- Application Software

System Software: The system software is a collection of programs designed to operate, control, and extend the processing capabilities of the computer itself. System software is generally prepared by the computer manufacturers.Eg:Windows, Linux, Android OS, Ios

Application Software:Application software products are designed to satisfy a particular need of a particular environment. All software applications prepared in the computer lab can come under the category of Application software.

Part-2

There are five generations of Programming languages. They are:

First Generation Languages : These are low-level languages like machine language. //0 1

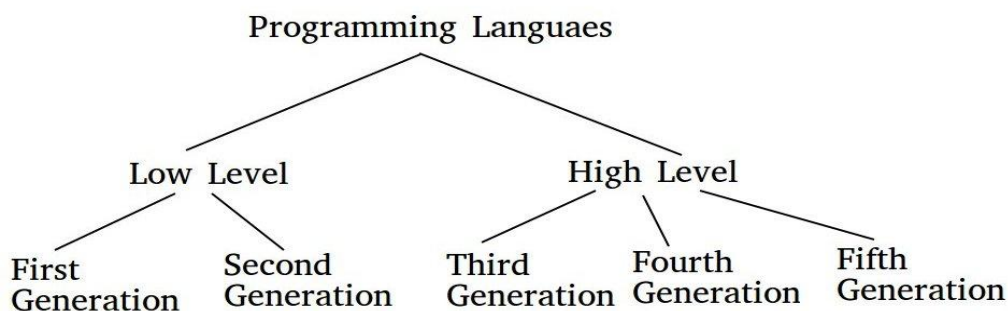
Second Generation Languages : These are low-level assembly languages used in kernels and hardware drives.// `ADD A B`

Third Generation Languages : These are high-level languages like C, C++, Java, Visual Basic, and JavaScript.

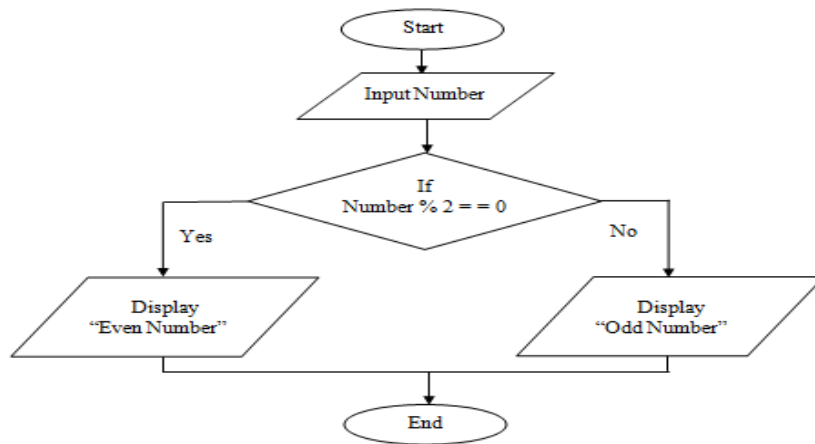
Fourth Generation Languages : These are languages that consist of statements that are similar to statements in the human language. These are used mainly in database programming and scripting. Examples of these languages include Perl, Python, Ruby, SQL, MatLab(MatrixLaboratory).

Fifth Generation Languages : These are the programming languages that have visual tools to develop a program. Examples of fifth-generation languages include Mercury, OPS5, and Prolog.

The first two generations are called low-level languages. The next three generations are called high-level languages.



Part 3: Here is the design of flowchart to check whether an input number is an even or odd.



2. What is the difference between a simple statement and a compound statement? Design an algorithm to generate the first n terms of the sequence 2 6 10 14 ...

Answer No. 2

A Statement is the smallest standalone element of a programming language which in itself is a collection of tokens of the language put in some proper sequence as per the syntax rules or grammar of the language. A sequence of one or more statements gives rise to a program. C language makes a distinction between statements and definitions, with a statement only containing executable code and a definition declaring an identifier.

The body of C functions (including the main() function) are made up of statements. These can either be **Simple Statements** or **Compound Statements**.

Simple Statements do not contain other statements and end with a semicolon called **Statement terminator**. The simplest kind of statement in C is an expression. The following statements are simple statements and most of the statements in a typical C program are Simple Statements of this form.

```
int a=5;
```

or

```
Sum= number1+number2;
```

```
or
```

```
root=sqrt(num1);|
```

Compound Statements have other statements inside them; may be simple or other compound statements. The term Compound Statement (or Block) refers to a collection of statements (both Simple and Compound) that are enclosed in braces to form a single unit. Compound statements are not terminated with semicolons. The statement shown below is a compound statement. Notice that it contains three simple statements enclosed in a pair of curly braces. Also, though individual simple statements end with semicolon, however the compound statement does not have one.

```
while(i<0){  
    printf("%d",i);  
    i--;  
}
```

Compound statements generally form the body of functions, loops, conditional statements, and so on.

Part -2 Here is the algorithm to generate the above sequence.

Step 1: Start

Step 2: Declare variables num num1,next.

Step 3: Read num from user.

Step 4: Assign num1=2.

```
Step 5:    do{  
            Display num1  
            next=num1+4;  
            num1=next;  
  
        } while( next<num)
```

Step 6: Stop

3.Why language processor is needed? Differentiate compilers from interpreters. Write a program to capture inputs and display outputs for characters. Write Comments in each line of your program.

Answer No.3

Part-1

Compilers or interpreters translate programs written in high-level languages into machine code that a computer understands. And assemblers translate programs written in low-level or assembly language into machine code. In the compilation process, there are several stages. To help programmers write error-free code, tools are available.

As Computer does not understand high level language, it needs to be converted into machine understandable code that is binary digits(0 and 1). To translate High level language into machine level language we need a language processor.

Part-2 Here is the difference between Compilers and interpreters.

S.
N
o.

Compiler

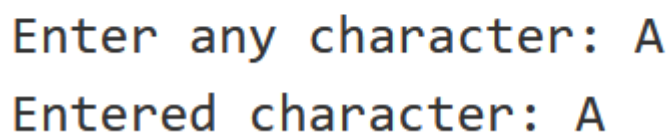
Interpreter

- | | | |
|----|--|--|
| 1. | Compiler scans the whole program in one go. | Translates program one statement at a time. |
| 2. | As it scans the code in one go, the errors (if any) are shown at the end together. | Considering it scans code one line at a time, errors are shown line by line. |
| 3. | Main advantage of compilers is it's execution time. | Due to interpreters being slow in executing the object code, it is preferred less. |
| 4. | It converts the source code into object code. | It does not convert source code into object code instead it scans it line by line |
| 5 | It does not require source code for later execution. | It requires source code for later execution. |
| Eg | C, C++, C# etc. | Python, Ruby, Perl, SNOBOL, MATLAB, etc. |
| . | | |

Part 3:

```
#include <stdio.h> //Standard input output
#include <conio.h> //console input output
int main(){ //Beginning of main function
    char ch; // Variable declaration
    printf("Enter any character: "); //Output function
    ch = getchar(); //taking input from user using getchar function
    printf("Entered character: %c\n", ch); //Displaying output
    getch();
    return 0;
}
```

Output



```
Enter any character: A
Entered character: A
```

4. What are the formal arguments in C-function? Mention the relationship between the formal arguments and actual arguments. Write a program to find the number entered by the user is prime or composite by using the concept of continue/break statement.

Answer No.4

Part 1:

Arguments which are mentioned in the definition/declaration of the function is called formal arguments. Formal arguments are very similar to local variables inside the function. The scope of formal arguments is local to the function definition in which they are used. Formal arguments are a copy of the actual arguments. A change in formal arguments would not be reflected in the actual arguments.

Part -2:

Actual Arguments

When a function is called, the values (expressions) that are passed in the function call are called the arguments or actual parameters.

These are the variables or expressions referenced in the parameter list of a subprogram call.

Actual Parameters are the parameters which are in the calling subprogram.

There is no need to specify datatype in actual parameters.

The parameters written in the function call are known as actual parameters.

Actual Parameters can be constant values or variable names.

Formal Arguments

The parameter used in function definition statement which contain data type on its time of declaration is called formal argument.

These are the variables or expressions referenced in the parameter list of a subprogram specification.

Formal Parameters are the parameters which are in called subprograms.

The datatype of the receiving value must be defined.

The parameters written in function definition are known as formal parameters.

Formal Parameters can be treated as local variables of a function in which they are used in the function header.

//Invoke (call) the method

```
int number1 = 25;  
int number2 = 47;  
int sum = add(number1, number2);
```

actual parameters
(or arguments)

//Method definition

```
public int add(int x, int y)  
{  
    return (x + y);  
}
```

formal parameters

Part 3:


```
#include <stdio.h>  
#include <conio.h>  
int main(){  
    int num,i;  
    printf("Please enter the number\n");  
    scanf("%d",&num);  
    for(i=2;i<num;i++){  
        if(num%i==0){  
            break;  
        }  
    }  
    if(i==num){  
        printf("%d is PRIME",num);  
    }  
    else{  
        printf("%d is Composite",num);  
    }  
    getch();  
    return 0;  
}
```

Output:

Please enter the number

7

7 is PRIME

 C:\Users\DELL\Desktop\Third year\PrimeComposite.exe

Please enter the number

9

9 is Composite_