LAB ASSIGNMENT NO.1

BASICS

SHREYAS SAWANT
D2A-55

AIM: Programming using C language

THEORY:

Algorithm: An Algorithm is a sequence of steps to solve a particular problem or an algorithm is an ordered set of unambiguous steps that produce a result and terminates in a finite time.

Flowchart: Flowchart uses different symbols to design a solution to a problem. By looking at a Flowchart one can understand the sequence of operations performed in a system. It's often considered as a blueprint of a design used for solving a specific problem.

Identifiers: Identifiers are names given to various elements of a program, such as variables, functions and arrays. Identifiers consist of digits and letters, in any order but the rule is that the first character should be a letter. An identifier can start with an underscore.

Keywords: Keywords are certain reserved words that have, standard, predefined meanings in C. These keywords can be used only for their intended purpose, they cannot be used as programmer-defined identifiers.

Data types: C supports different types of data. Each of which may be represented differently within the computer's memory. Each data type requires different memory requirements which may vary from one C compiler to another. For example, int, char, float and double. Some basic data types can be augmented by using the data type qualifiers short, long, signed & unsigned.

Constant: Constant in C refers to fixed values that do not change during the execution of a program. There are two simple ways in C to define constants − using #define preprocessor or using const keyword. For each type of constant, these bounds will vary from one C compiler to another.

Variable: A quantity which may vary during program execution is called a variable. Each variable has a specific storage location in memory where its value is stored. The value of the variable at any instant during the execution of a program is equal to the number stored in the storage location identified by the name of the variable.

Operators in C:

i. *Arithmetic operator: P*erforms mathematical operations such as addition, subtraction and multiplication on numerical values (constants and variables). For example: +, -, *, %, /.
ii. *Relational operator:* Checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0. For example: ==, !=, etc.
iii. *Unary operators*: Operators that act upon a single operand to produce a new value. C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.
Other operators include logical, conditional(ternary), bitwise etc.

Operator precedence: Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; Associativity is used when two operators of the same precedence appear in an expression. Associativity can be either Left to Right or Right to Left.

Library functions: Standard library functions or simply C Library functions are inbuilt functions in C programming The prototype and data definitions of the functions are present in their respective header files and must be included in your program to access them.

Preprocessor: Before a C program is compiled in a compiler, source code is processed by a program called preprocessor. This process is called preprocessing. Commands used in preprocessor are called preprocessor directives and they begin with “#” symbol. Example #define.

Input-output operations: Input means to provide the program with some data to be used in the program and Output means to display data on screen or write the data to a printer or a file C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result. The standard input-output header file, named stdio.h contains the definition of the functions. For example, getchar(), putchar(), printf(), scanf(), gets(), puts().

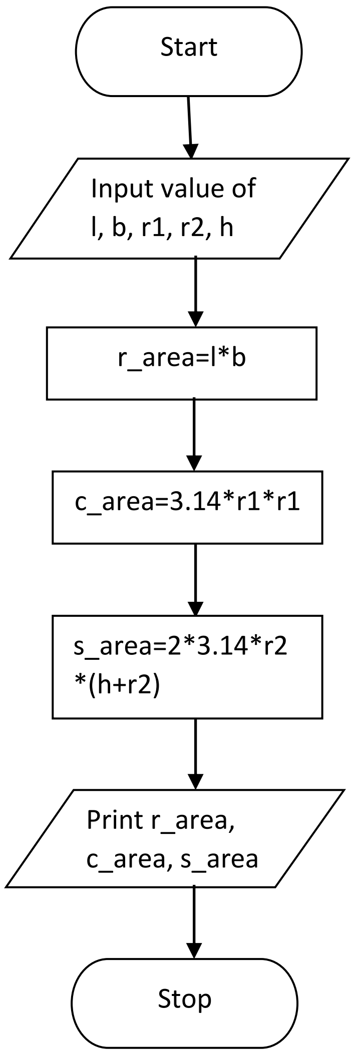
a) Write a program to find the area of the rectangle, Circle and Surface area of a Cylinder.

Variable list:
l= length of the rectangle
b = breadth of the rectangle
r1 = radius of the circle
r2 = radius of the cylinder
h = height of the cylinder
r_area = area of the rectangle
c_area = area of the circle
s_area= surface area of the cylinder

Algorithm:

```
Step 1: Start
Step 2: Input values of l, b, r1, r2, h
Step 3: r_area = l*b
Step 4: c_area= 3.14*r1*r1
Step 5: s_area = 2*3.14*r2*(h+r)
Step 6: Display r_area, c_area, s_area
Step 7: Stop
```

Flowchart:



Code:

```
#include <stdio.h>
#define pi 3.14
void main()
{       int l, b, r_area;
        double r1, r2, h, c_area, s_area;
        printf("Enter length and breadth of rectangle respectively ");
        scanf("%d%d",&l,&b);
        printf("Enter radius of cylinder ");
        scanf("%lf",&r1);
        printf("Enter radius and height of cylinder respectively ");
        scanf("%lf%lf",&r2,&h);
        r_area=l*b;
        c_area=pi*r1*r1;
        s_area= 2*pi*r2*(h+r2);
        printf("Area of rectangle= %d\n", r_area);
        printf("Area of circle= %lf\n", c_area);
        printf("Surface Area of Cylinder= %lf\n", s_area);
}
```

Output:

```
Enter length and breadth of rectangle respectively 3 4
Enter radius of cylinder 4.567
Enter radius and height of cylinder respectively 3.456 6.78
Area of rectangle= 12
Area of circle= 65.492515
Surface Area of Cylinder= 222.158868
```

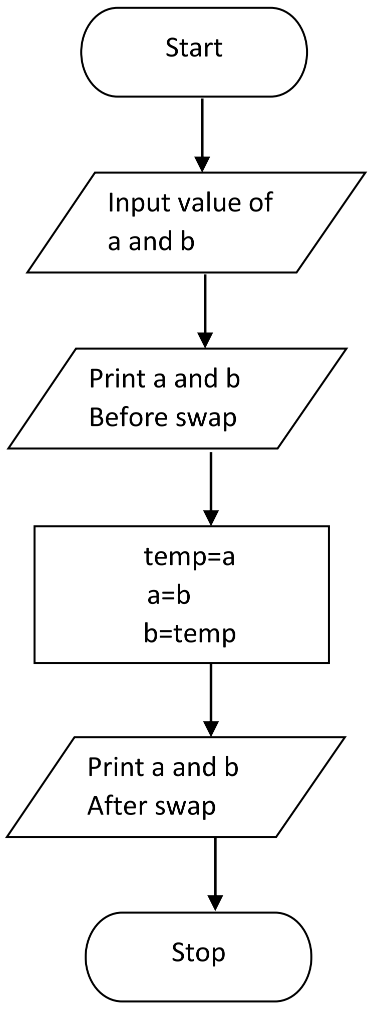
b) Swap two numbers:

(i) With temporary variable

Variable list:
a= one of the numbers to be swapped
b=one of the numbers to be swapped
temp= temporary variable

Algorithm:
```
Step 1: Start
Step 2: Input two numbers a, b
Step 3: Display Before Swap values a, b
Step 4: temp = a
Step 5: a = b
Step 6: b=temp
Step 7: Display After Swap values a, b
Step 8: Stop
```

Flowchart:



Code:

```
#include <stdio.h>
void main()
{      int a,b,temp;
      printf("Enter two numbers ");
      scanf("%d %d",&a,&b);
      printf("Before Swapping: \na=%d\nb=%d\n",a,b);
      temp=a;
      a=b;
      b=temp;
      printf("After Swapping: \na=%d\nb=%d\n",a,b);
}
```

Output:

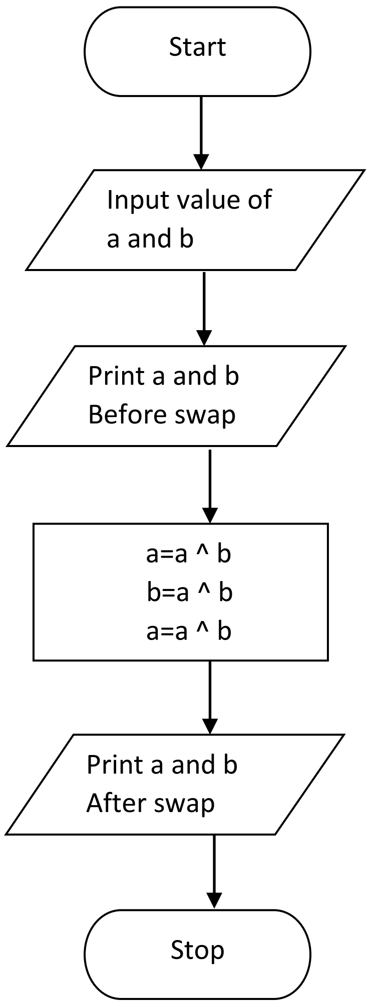
```
Enter two numbers 4 6
Before Swapping:
a=4
b=6
After Swapping:
a=6
b=4
```

(ii) Without temporary variable

Variable list:
a= one of the numbers to be swapped
b=one of the numbers to be swapped

Algorithm:
Step 1: Start
Step 2: Input two numbers a, b
Step 3: Display Before Swap values a, b
Step 4: a = a^b
Step 5: b= a^b
Step 6: a= a^b
Step 7: Display After Swap values a, b
Step 8: Stop

Flowchart:



Code:

```
#include <stdio.h>
void main()
{    int a,b,temp;
      printf("Enter two numbers ");
      scanf("%d %d",&a,&b);
      printf("Before Swapping: \na=%d\nb=%d\n",a,b);
      a=a^b;
      b=a^b;
      a=a^b;
      printf("After Swapping: \na=%d\nb=%d\n",a,b);
}
```

Output:

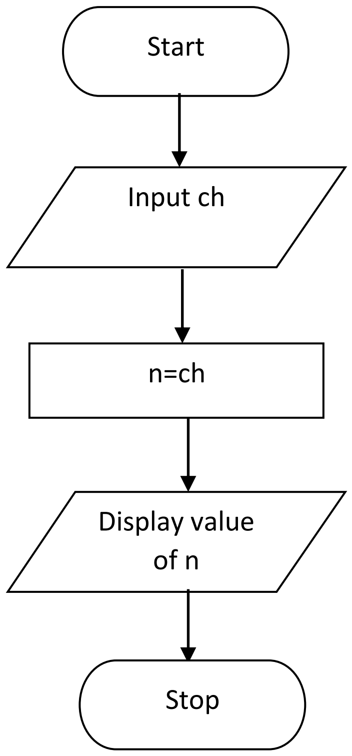
```
Enter two numbers 56 67
Before Swapping:
a=56
b=67
After Swapping:
a=67
b=56
```

c) Find ASCII value of a character

Variable List:
ch = character whose ASCII value is to be found
n= integer to store ASCII value of character

Algorithm:
Step 1: Start
Step 2: Input the character ch
Step 3: A = ch
Step 3: Display the value of A
Step 4: Stop

Flowchart:



Code:

```
#include <stdio.h>
void main()
{      char ch;
       int n;
       printf("Enter character whose ASCII value is to be determined ");
       scanf("%c",&ch);
       n = ch;
       printf("ASCII value of %c is %d",ch,n);
}
```

Output:

```
Enter character whose ASCII value is to be determined ~
ASCII value of ~ is 126
```