UNIVERSITY OF RUHUNA

FACULTY OF TECHNOLOGY

DEPARTMENT OF ICT

J.M.N.R.D. JAYAWEERA

TG/2020/700

PROGRAMMING PRACTICUM

TUTORIAL 02

1. Describe the steps in problem solving process?

- Analyze the problem
  - When we got a problem, we needed to identify clearly that problem to begin to find the solution for it. If we failed to identify problem clearly, we will end up with the solution that is not solved our problem. Because of it we need to read and analyze the problem statement carefully to list the principal components of the problem and decide the core functionalities that our solution should have. After analyzing the problem, we can decide what we should input and what outputs we can get.

- Developing an algorithm
  - We need to decide on a solution before writing a program. We can provide this solution by using natural language. It means we can use an algorithm to provide the solution. The algorithm describes the way to solve the problem step by step. First, we find a tentative solution and rearrange and develop it until we capture the best solution. We can use algorithms to find the best way to solve problems.

- Coding
  - After we got the best algorithm for the problem, we can convert the algorithm to a format which can be understood by the computer to generate the desired solution. We can use any high-level programming language to write those programs.

- Testing and Debugging
  - After coding the solution that program must be tested by various parameters. Our program should fulfil the requirements of the users. In this case, we need to check all inputs and all outputs should are correct or not. We need to check all syntactical errors and recorrect them. After done, we need to check for logical errors.

    In the software industry, we use unit testing, component testing, integration testing etc. to develop those solutions in complex applications. Testing and debugging are done again and again until all errors are removed from the program.

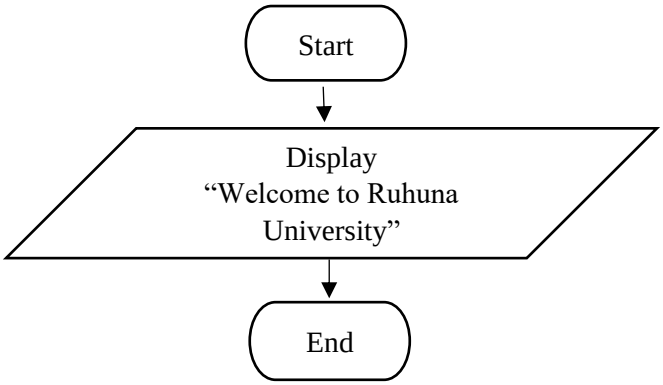
2. What is an algorithm?

- A set of exact steps which when followed, solve the problem, or accomplish the required task.
- Step-by-step problem-solving process to solution achieved in finite amount of time.

3. Draw flow-charts and pseudo codes for the following problems

i. Print Welcome to Ruhuna University

Flow chart

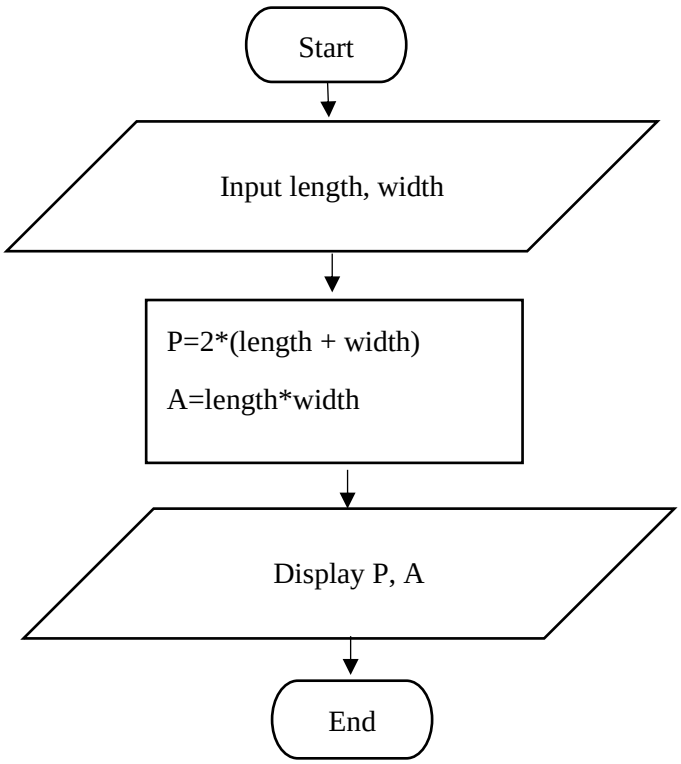


Pseudo Code

```
Begin
        Display "Welcome to
        Ruhuna University"
End.
```

ii. Calculate the perimeter and area of the rectangle given the length and the width

Flow chart



Pseudo Code

```
Begin
        Input length, width
        P=2*(length + width)
        A=length * width
        Display P, A
End.
```

4. Following program contains basic structure of C code. Explain meaning of each line?

- **Line1 // program to print the "Welcome to Ruhuna"**
  - This is a line comment, and those comments provide information to the people who read the program.
  - Those comments are removed by the preprocessor because of that compiler ignores them.

- **Line2 #include <stdio.h>**
  - This is a preprocessor directive and lines that are beginning with # are processed by the preprocessor before the program is compiled.
  - This specific line tells the preprocessor to include in the program the contents of the input/output stream header file<stdio.h>.
  - This file must use for programs that input data from keyboard and output data to screen.

- **Line3 int main()**
  - This is a part of a every C program.
  - Every C program has only one main function and other function when it's necessary.
  - The parentheses after "main" indicate that it is a program building block called a function and it is like other functions.
  - Main function is called by the OS when the user runs the program.
  - The keyword int indicates that main function returns an integer value.

- **Line4 {**
  - This indicates the begin of the main function.

- **Line5 printf("Welcome to Ruhuna\n");**
  - This displays the string of characters which are between the quotation marks.
  - And then by \n (new line control character) cursor moves to the next line.

- **Line6 return 0;**
  - This is used to return a value from a function.
  - If program returns 0, it means program terminated successfully.

- **Line7 }**
  - This indicates the end of the main function.

5. Explain advantages of using IDE (Integrated Development Environment) tools.

a. Maximize the programmer productivity
- The integrated development environment is designed to maximize programmer productivity by providing tight-weave components with the same user interface.
- Increased Efficiency – faster coding with less effort
- Collaboration – A group of programmers can easily work together within an IDE

b. Provide GUI environment
- We can easily code our solution and we can easily find out the errors and fix them.
- By using GUI, we can increase accuracy and efficiency

c. Faster
- When we use IDE for programming it is a well-designed software for the programming, and it provide us to faster way to code our programs and compile and run.

d. Providing instant feedback when syntax errors are found
- If our program has some syntax errors those errors will be shown when compilation process is done. It will show details of our program syntax errors.

e. Multiple programming languages within a ONE UI
- It means we can use one IDE to coding, compile and execute a program by different high-level languages.

6. What are the differences between high-level language and low-level language? Give some examples.

High-level language	**Low-level language**
• Programmer friendly. • Need a compiler or an interpreter for translation. • Program execution is slower than low-level languages. • Well structured. • Portable. • Memory efficiency is low. • Easy to debug. **Ex:** ▪ C++ ▪ Python ▪ Java ▪ C# ▪ JavaScript	• Machine friendly. • No need a translator but only for assembly language it's required an assembler to translation. • Program execution is faster than high-level language. • Not well structured. • Not portable. • Memory efficiency is high. • Difficult to debug. **Ex:** ▪ Machine Language ▪ Assembly Language

7. Develop an algorithm to input hour's work and hourly rate through the keyboard and print the salary.

Algorithm

Step 01: Start
Step 02: Input Hour's work and Hourly rate
Step 03: Find salary by multiplying Hour's work by Hourly rate (Salary=Hour's work*Hourly rate)
Step 04: Display salary
Step 05: End

Pseudo code

```
Begin
    Input HW, HR
    S=HW*HR
    Display S
End.
```

8. Write a pseudo code to calculate the hypotenuse (Length c) of a right-angled triangle given the other two lengths. Convert your pseudo code into the C program. (Hint: you can use sqrt() function defined in math.h header file, double sqrt(double x).->returns square root of x).

Pseudo code

```
Begin
    Input a, b
    c =√a²+b²
    Display c
End.
```

C Program

```
#include<stdio.h>
#include<math.h>
int main()
{
        float a,b,c;
        printf("Enter length of a:");
        scanf("%f",&a);
        printf("Enter length of c:");
        scanf("%f",&b);

        c=sqrt(pow(a,2)+pow(b,2));

        printf("Length of c(hypotenuse) is:%.2f",c);

        return 0;
}
```

9. Show the value of x after each statement is performed.

i. **x=7+3*6/2-1;**

= 7+18/2-1
= 7+9-1
= 16-1
= 15

ii. **x=2%2+2*2-2/ 2;**

= 0+2*2-2/2
= 2*2-2/2
= 4-1
= 3

iii. **x= (3*9*(3+(9*3/ (3)));**

= (3*9*(3+(9*3/3))
= (3*9*(3+(27/3))
= (3*9*(3+(9))
= 3*9*(3+9)
= 3*9*(12)
= 3*9*12
= 27*12
= 324

iv. **x = 10 % 3 * 4 + 5 * 2;**

= 1*4+5*2
= 4+10
= 14

v. **x = 7.0 / 4.0;**

= 1.75

vi. **x = 7.0 / 4;**

=1.75

10. Write a program that asks the user to enter two numbers obtains them from the user and prints their sum, product, difference, quotient, and remainder.

C program

```
#include<stdio.h>
#include<math.h>
int main()
{
		float a,b,tot,product,dif,rem; //a for first number and b for second number
		int div; //for division

		printf("Enter First Number:");
		scanf("%f",&a); //get value for first number
		printf("Enter Second Number:");
		scanf("%f",&b); //get value for second number

		tot=a+b; //sum
		product=a*b; //product

		if (a>b) //for find bigger number to find difference
		{
		    dif=a-b;
		}
		else
		{
		    dif=b-a;
		}

		div=a/b; //quotient
		rem=fmod(a,b); //remainder

		printf("Sum:%.2f\nProduct:%.2f\nDifference:%.2f\nQuotient:%d\nRemainder:%.2f\n",tot
,product,dif,div,rem); //to print answers

		return 0;

}
```