Department of Networks

First Year

Problem Solving and Programming 1

Typical Programming (software development) Method

1. Determine the problem requirements
2. Analyze the problem to be solved
3. Design an algorithm or a flowchart to solve the problem
4. Implement the algorithm by writing your C++ code
5. Compile your program
   . If there are compile errors, go back to 4 and debug your code
6. Run your executable, test and verify
   . If the program does not run as expected go back to 2, 3 or 4 as appropriate
7. Maintain and update

Problem Requirements

Specifying the problem requirements: state the problem clearly and unambiguously and to gain a clear understanding of what is required for its solution.

Analyzing the problem: involves identifying:

- all the inputs - the data you have to work with,

- all the outputs - the desired results,
- any additional requirements or constraints.

Design

An algorithm is generally a high level abstraction that is not actually code. However, generally an algorithm expresses the ideas of a program.

Designing the algorithm: develop the actual list of steps (the algorithm) to solve the problem, and then to verify that the algorithm solves the problem as intended.

- Usually the hardest step in the development process.
- Best not to try to solve every detail at the beginning - use top-down design: list the major steps (subproblems) first then solve each subproblem.

Most algorithms consist of at least the following subproblems:

1. Get the (input) data
2. Perform the computations
3. Display the results

Implementation, Testing, and Maintenance

- **Implementing the algorithm:** involves writing your algorithm (pseudo-code) as a program in your chosen programming language.

- **Testing and verifying:** requires thorough testing (on various inputs) to verify that your program works as desired.
- **Maintenance and updating:** as things change over time (ie. TAX increase) a program may need to be updated and maintained

Some concepts used in programming:

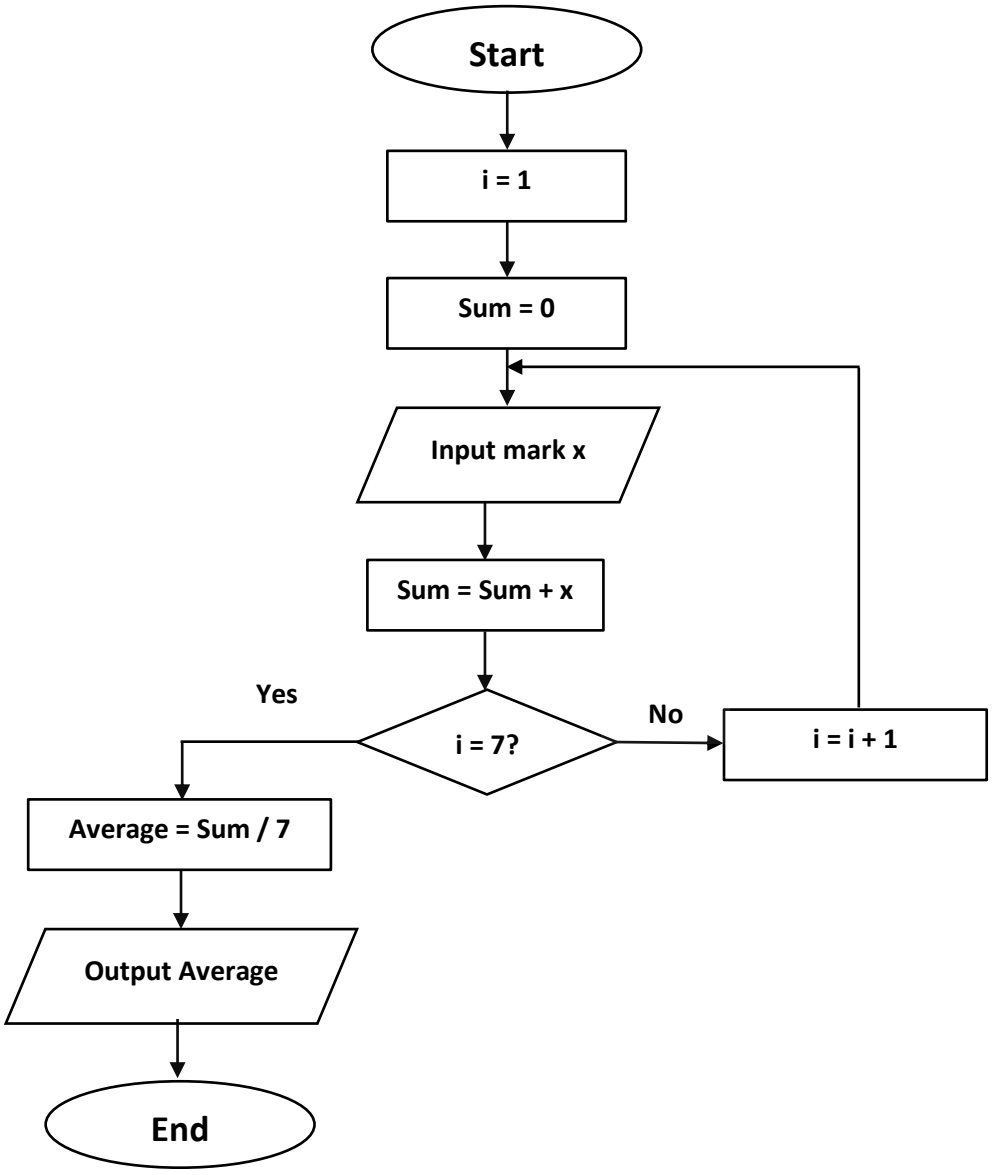
Loops: to repeat a statement or group of statements for number of times or as long as a certain condition is true.

Counters: to count things, usually counters need an initial value (could be zero or one or any other value). Some examples: number of students passed in an exam, number of positive integers in 100 random numbers, number of even numbers in 100 random numbers.

Summations: to find the summation (sum) of series of numbers or series of expressions. Some examples: summation of numbers 1-100, summation of $(x-1)^2$.

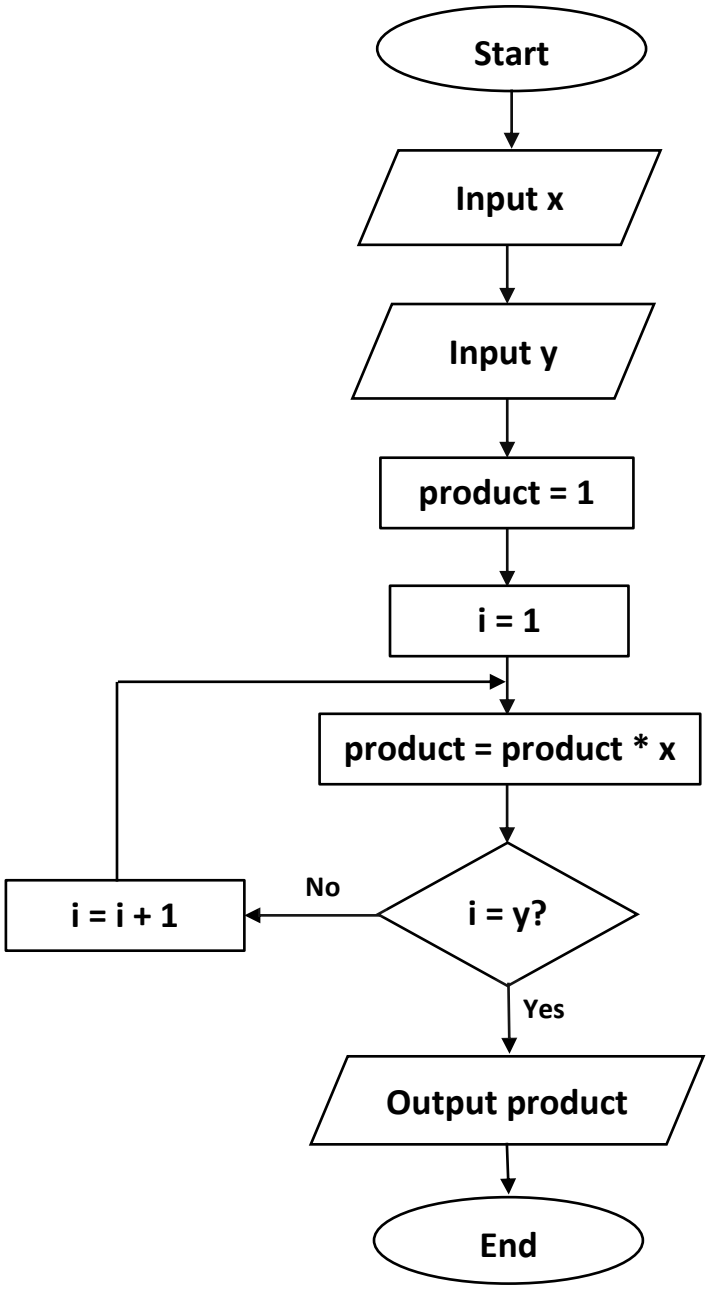
flowchart examples about repetition

Example 1: Finding the average of 7 marks



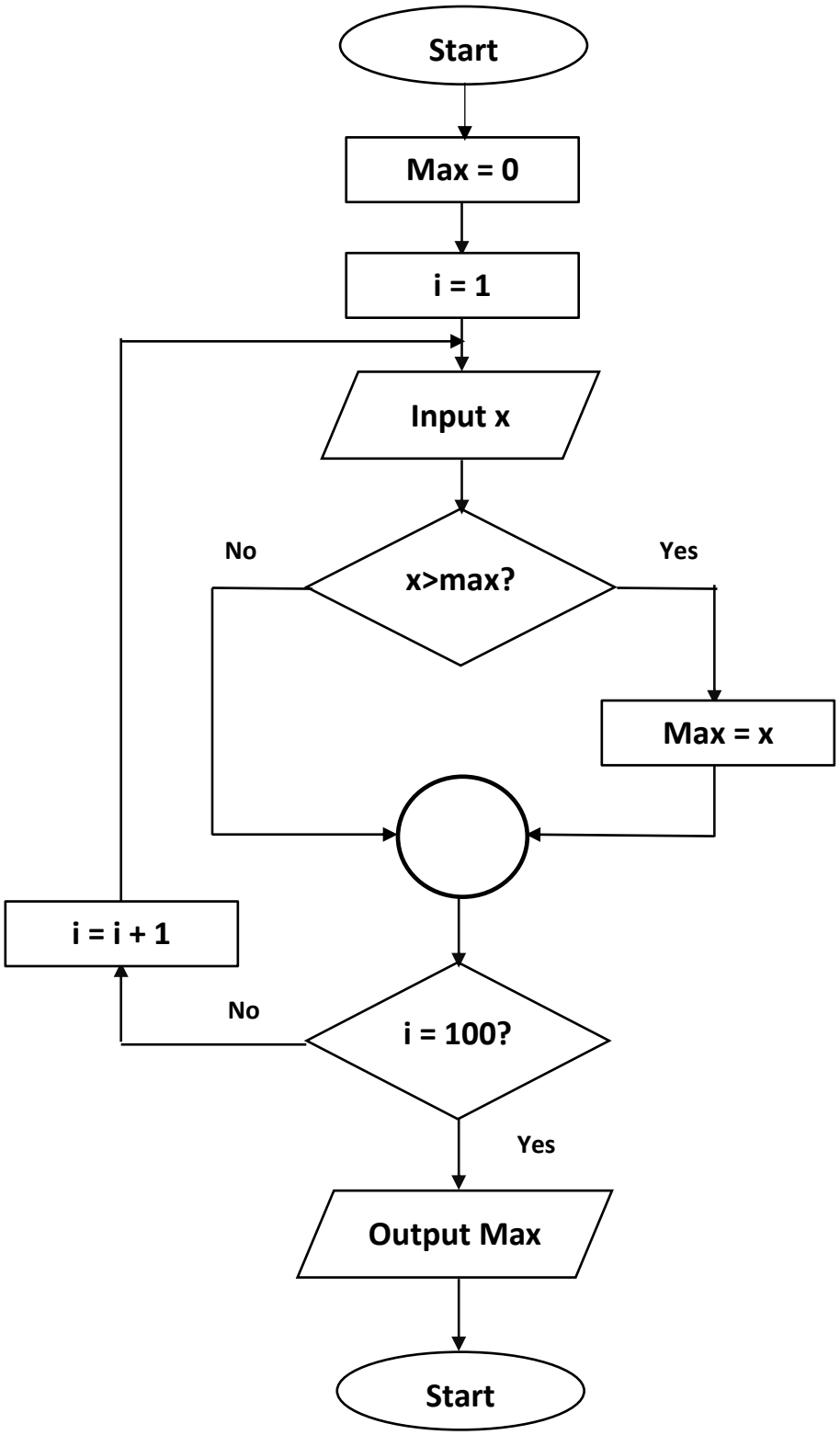
HW: modify the above example to work on n marks instead of 7 (n entered from keyboard).

Example 2: Finding x^y



HW: calculate the factorial of an integer number X, where factorial(x) = 1 * 2 * 3 …. X

Example 3: Finding the maximum value from 100 random values (positive integers) entered from keyboard.



Exercises:

1. Input 100 random numbers and count the odd and even numbers
2. Input 100 random integer numbers (positive and negative) and sum the positive and negative numbers
3. Check if a number is a prime or not
4. Output this series:

   1 2 4 8 16 ….. 1024
5. Output this series:

   2 4 6 8 10 …..100
6. Sum this series:

   3 5 7 ….99
7. Output this series:

   2 3 5 8……144
8. Calculate this series:

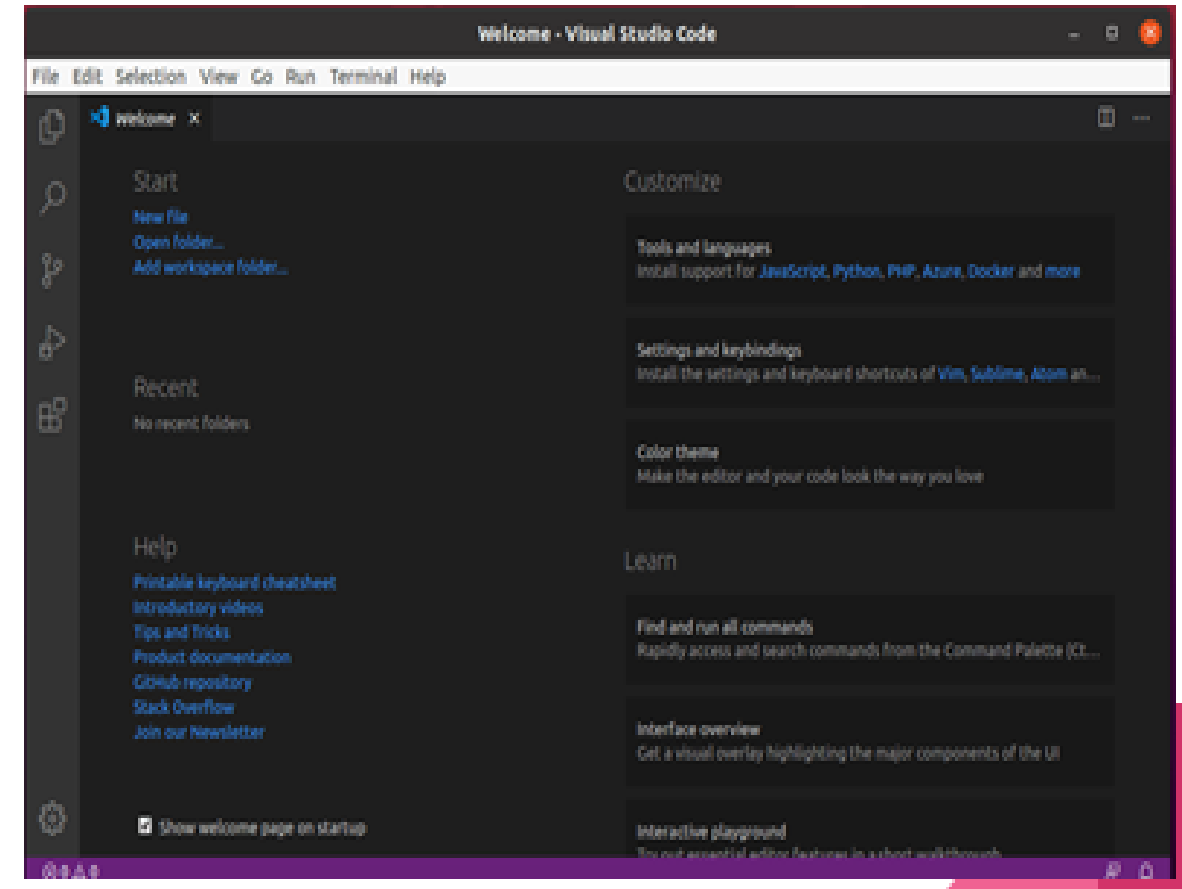
   $Y = \frac{1}{x^2} + \frac{2}{x^3} + \frac{3}{x^4} + \cdots . + \frac{n}{x^{n+1}}$
9. Calculate this series:

   $Y = \frac{1}{2!} + \frac{2}{3!} + \frac{3}{4!} + \cdots + \frac{n}{(n+1)!}$
10. Calculate this series:

    $Y = \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \cdots + \frac{x^n}{n!}$

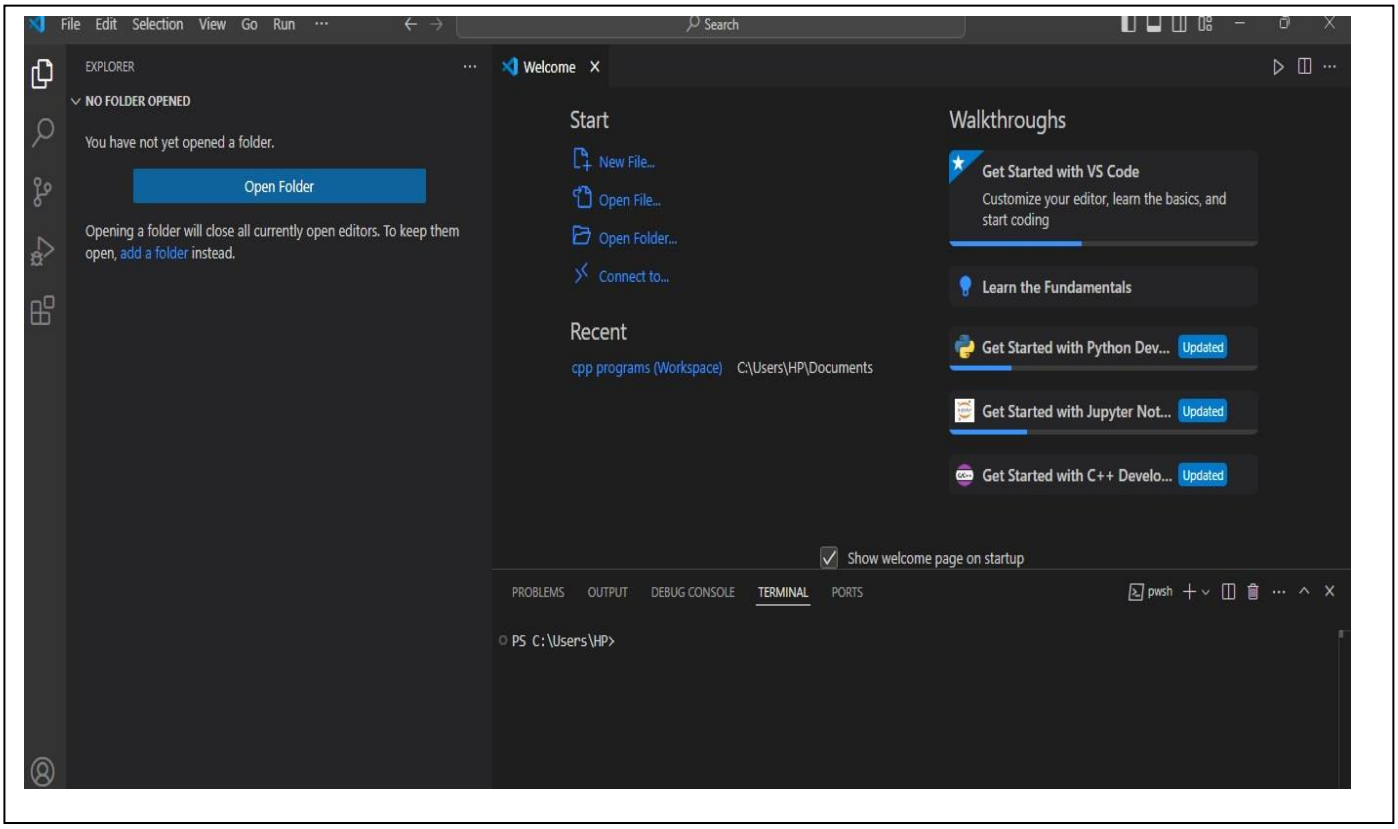
C++ IDE

- The easiest way to get started with C++, is to use an IDE.
- An IDE (Integrated Development Environment) is used to edit and compile code.
- In our lecture, we will use Visual Studio Code, which is free to download from
- https://code.visualstudio.com/sha/download?build=stable&os=linux-deb-x64
- Once the Visual Studio Code is downloaded and installed, open the IDE as shown in the figure
- After the installation is complete, click on the File menu then select New File.



Install the C/C++ extension by searching for 'c++' in the Extensions view

To create a new C++ program, click on New File and give a new to the file



The code first code to learn in C++:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World!";
    return 0;
}
```

How to compile and run C++ code:

- Don't worry if you don't understand the code above - we will discuss it in detail later. For now, focus on how to run the code.
- Run the program by clicking on Run Code. This will compile and execute your code. The result will look something to this:

```
Hello World!
```

Example explained

- **Line 1:** This line includes the <iostream> library, which allows the program to perform input and output operations.
- **Line 2:** This line allows the program to use all elements in the std (standard) namespace without needing to prefix them with std::. It simplifies the code, so instead of writing std::cout, you can just write cout.

- **Line 3:** This defines the main function, which is the entry point of any C++ program. The program execution begins with this function. It returns an integer (int), usually 0 to signify successful execution.
- **Line 4:** The curly braces {} marks the beginning and the end of a block of code.
- This is the opening curly brace {, which marks the beginning of the main function's code block.

```
cout << "Hello World!";
```

- This line uses cout (from <iostream>) to print "Hello World!" to the console. The << operator directs the text on its right to the output stream cout.
- This statement returns 0 to indicate that the program has ended successfully. The return value is optional in main for some C++ compilers, but returning 0 is a good practice to signify successful execution.
- This is the closing curly brace }, which indicates the end of the main function.

Examples

```
cout<<"Hello World! \n";

cout<<"I will print on a new line.\n";

cout<<"Hello World! ";
```

```
cout<<"I will print on the same line.";
```

Result:

Hello World!

I will print on a new line.

Hello World! I will print on the same line.