Algorithms

1. Objectives

- Introduce you to the world of programming
- Programming constructs
- A good design for your program will save you an incredible amount of time testing at the computer.

2. Design You Solution

- The following steps are used in designing a large program or a simple function or procedure:

  1. **Define the problem.**

     - Describe the input data available (when the program starts),
     - Describe the output (result) desired,

  2. **Plan a solution** (design the program). This stage often uses one of two ways to describe the solution:
     1. **pseudocode** - An English-like structured language, or

2. A **flow chart** - a pictorial representation of the step-by-step solution to the problem.

3. **Code the program**: Translate the logic from the flow chart/pseudocode into a programming language.

4. **Test the program**: Run a number of test cases through the program to demonstrate that it works.

5. **Document the program**: This is the time for a formal document called the user manual to be produced. Most of the program code should already contain adequate comments for documentation in the coding phase.

3. Structure of an algorithm:

- Your algorithm consists of the following:
    - A name of your program

- A set of inputs
- A mark that indicates the beginning of the set of executable statements
    - A set of executable statements
- A mark that indicates the end of your program.

4. Pseudocode:

- It consists of short, English phrases used to explain specific tasks within a program's algorithm.

- It is a "text-based" detail (algorithmic) design tool.

- One programmer should be able to take another programmer's pseudocode and generate a program based on that pseudocode.

- **Why is pseudocode necessary?**

  The programming process is a complicated one. You must first understand the problem specifications, of course, then you need to organize your thoughts before creating your program

- Writing pseudocode WILL save you time later during the construction & testing phase of a program's development.

5. Example

- Write a program that obtains three numbers from the user. It will print out the sum of those numbers.
    - **Pseudocode:**
        - Ask the user to enter the first integer.
        - Obtain user's first integer input.
        - Ask the user to enter a second integer.
        - Obtain user's second integer input
        - Ask the user to enter a second integer.
        - Obtain user's third integer input.
        - Compute the sum of the three user inputs.

            - Display an output prompt that explains the answer as the sum Display the result.

- **Sequences**: Straight-Line Algorithms
    - A sequence refers to a set of statements separated by a delimiter, i.e., semicolon.

- The program enters the sequence, execute from the top to the bottom each statement, until the end of the sequence

- Example:
    - Ask a user to enter the price of an item being purchased at a store and calculate and display the final price including sales tax. Let us assume the sales tax is 4.5%.
        - Pseudocode:
            - Ask the user for the price.
            - Read the price into the variable x.
            - Compute the tax value y as follows:

              $y = 0.045 * x;$

            - Compute the final price:

total = x + y

- Display total.

6. Selection or Conditional Execution

- Selection structures are used to check conditions and specify actions based upon those conditions or to choose an action from multiple options.
- Examples of selections:
    - One-way selection:

```
if (condition(s)) the
    Sequence
end if;
```

    - To-way selection (or binary selection):

```
if (condition(s)) the
    Sequence
else
    Sequence
end if;
```

- Example: Compute the grade letter of a given course according the following chart:

Grade	Grade Letter
90-100	A
80-89	B
70-79	C
65-69	D
64-below	F

7. Looping or Iteration

- A common requirement in computing is to repetitively perform the same, or very similar, sequence many times.

- Repetition is achieved using a loop:
    - A statement that causes the computer to repeat the same sequence of code many times.

- Looping statements almost invariably include a condition, since it pays to stop repeating yourself eventually.

- **Infinite loop:**
    - It is a loop that never stops
    - It is usually a bug. However there are always exceptions.

- **Two types of loops:**
    - Defined loop: The number of iterations is defined ahead of time.
    - Undefined loop: The number of iterations is not known in advanced.

- **Examples of looping:**
  - For loop,
  - While loop,
  - Nested loops,
  - Etc.

- Examples:

  - Print a number n of Hellos where n is a user input values.

Algorithm 1:	Algorithm 2:
- Comments: this pseudocode uses the following variable: Variable **n** is used to store the user input. - Ask the user to input a value - Read the user input in n - Repeat n times Print ("hello"); - end repeat;	- Comments: this pseudocode uses the following variables: ✓ Variable **n** is used to store the user input. ✓ Variable **i** is used to control the number - Ask the user to input a value - Read the user input in n - For i=1 to n do Print ("hello"); - end for;

  - Compute the average number of a set of input values entered by the user. The set of inputs is ended by –1.
    - **Algorithm**:

- Comments: We are going to use the following variables:
  - ✓ variable **total** to store the sum of all user input values. Initially it is set to zero.
  - ✓ variable **count** to store the total number of user inputs. Initially it is set to zero.
  - ✓ variable **average** to store the average of user input values. Initially it is set to zero.

- **Start the program**:

```
- Ask the user to input a value;
- Read the user input in x;
- while x is different from –1 do
      add the value of x to total:
      total=total+x;
      count=count+1;
      ask the user to input a value;
      read the user input in x;
- end of while
- Compute the average as follows:
      - average=total/count;
      - print average on the screen;
      - End of program
```

8. Questions/Practice:

- Write the pseudocode of a program that computes the sum of all multiples of 10 between 1 and 1000 (10, 20, …). Your pseudocode should also print each multiple of 10.

- Write the pseudocode of a program that computes the sum of all even numbers between 1 and 1000 (0, 2, 4, …).

- Write the pseudocode of a program that adds only negative numbers within a set of user input numbers. The processing stops when the user enters –999.

- Write the pseudocode of a program that computes the maximum number of a set of user input numbers. The processing stops when the user enters –1.