SEHH2042 Computer Programming

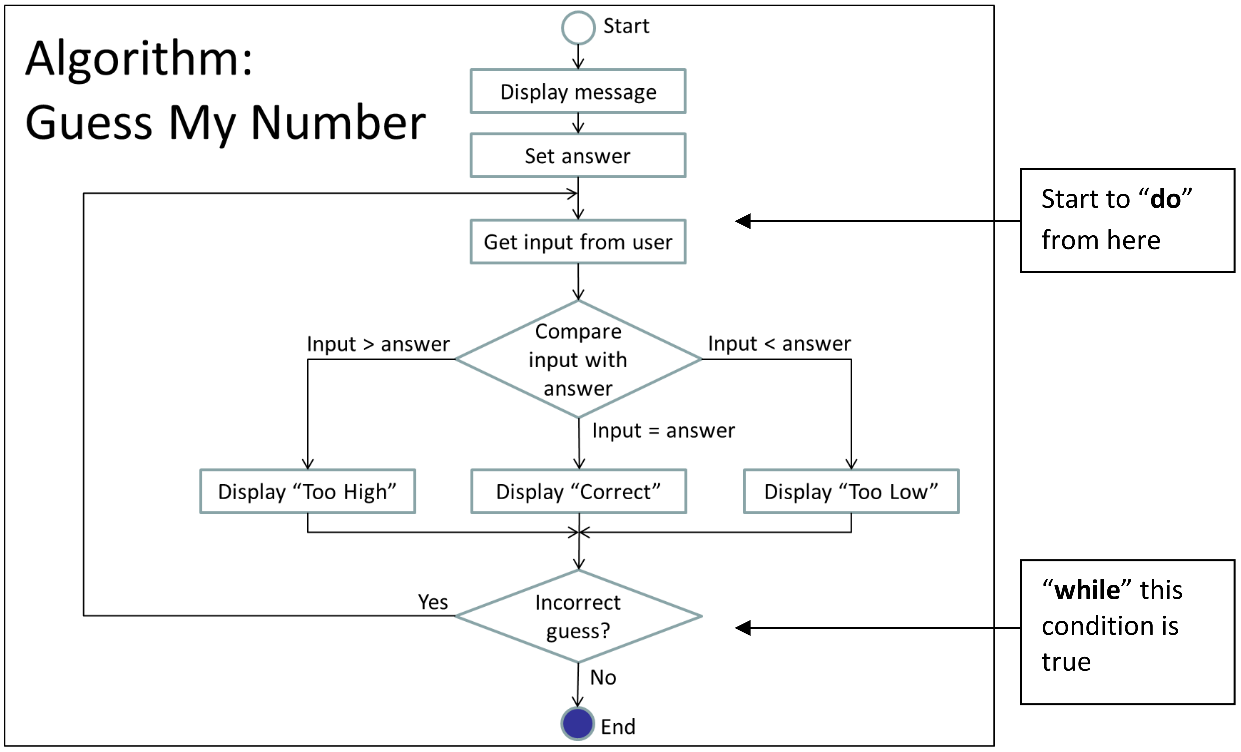
Tutorial 3 – Repetition statements (1)

Q1. Write a program **GuessNum.cpp** that asks user to guess an integer in 1 – 100. If the guess is smaller than the answer, it shows "Too low. Try again." If the guess is larger than the answer, it shows "Too high. Try again." If the guess is correct, it shows "Excellent! Correct guess." You can use any value (1 – 100) for the answer. Use **do-while loop** in your program.

Sample result (e.g. answer is 31):

```
I have a number between 1 and 100.
Can you guess my number?
Guess: 14
Too low. Try again.
Guess: 55
Too high. Try again.
Guess: 31
Excellent! Correct guess.
```

Program analysis and algorithm:



Follow the steps below to write the program:

1. Include necessary header files/library and write the main block.

```
#include <iostream>
using namespace std;

int main()
{
    // code of subsequent steps

    return 0;
}
```

2. Declare necessary variables.

```
    int input, ans = 31;
```

Initialize *ans* with the answer value.

3. Display welcome message before the game.

```
    cout << "I have a number between 1 and 100.\n";
    cout << "Can you guess my number?\n";
```

4. Start the **do-while loop** with condition of “incorrect guess”.

```
    do {

        // code to implement one round,
        // to be repeated by loop.

    } while (input != ans);
```

Continuation condition controls the loop body should repeat again or not. If input is incorrect, let the player try again.

5. Prompt for user input of a guess value, store the value to variable *input*.

```
    cout << "Guess: ";
    cin >> input;
```

This is one round of the game. Repeat it for every incorrect guess.

6. Display the guess result by comparing *input* and *ans*, similar to T2 Q1.

```
    if (input > ans)
        cout << "Too high. Try again.\n";
    else if (input < ans)
        cout << "Too low. Try again.\n";
    else
        cout << "Excellent! Correct guess.\n";
```

Q2. Write a program **Factorial.cpp** that accepts an integer *n* from the user, and displays the value of *n*!. Use **for loop** in your program. Assume the user must input non-negative numbers.

The factorial of a non-negative integer *n* (*n*!) is defined as:

$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$ (for *n* > 1)
$n! = 1$ (for *n* = 0 or *n* = 1)

Think: what should do in the nth round?
- n x (n – 1) ?
- (previous result) x n ?

Sample result:
```
n: 5
5! = 120
```

Q3. Write a program **SumOdd.cpp** that accepts two integers, *valueA* and *valueB* from the user. It then adds all odd integers from the smaller of *valueA* and *valueB*, to the larger of *valueA* and *valueB* (use a **for loop** to do so). Finally, it prints *valueA*, *valueB* and the summed result on the screen as shown below.

Sample result 1:
```
Please input value A: 11
Please input value B: 7
Sum of odd integers from 7 to 11 is 27
```
Sample result 2:
```
Please input value A: 5
Please input value B: 8
Sum of odd integers from 5 to 8 is 12
```

Need to check from A to B or from B to A

Q4. Write a program **Power.cpp** that accepts user's input of the values of *base* (double type) and *exponent* (integer type), then calculates and displays the result of $base^{exponent}$. Assume the exponent is a positive, non-zero integer. What kind of loop should be used?

Sample result 1:
```
Enter the base value: 2
Enter the exponent value: 3
2 to the power 3 is 8
```
Sample result 2:
```
Enter the base value: 1.5
Enter the exponent value: 4
1.5 to the power 4 is 5.0625
```

Enhanced version: Update your program so that it also allows zero and negative integer exponent.

Q5. Write a program **Prime.cpp** that prompts for user input of a positive integer and displays whether the input number is a prime number of not. A number is a prime number if it is divisible by 1 and itself only. Note that 1 is not a prime number.

Sample result 1:
```
Input a positive integer: 13
13 is a prime number
```
Sample result 2:
```
Input a positive integer: 123
123 is not a prime number
```

Hint: All numbers, no matter prime or not, are divisible by 1 and itself. Except 1 and itself, a prime number does not has other factor.