ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

Department of Computer Science and Engineering (CSE)

CSE 4108: Structured Programming I Lab
Lab 6, Section 2A

Objectives

- Introduction to `switch-case`
- Loops with `break` and `continue` statement
- Introduction to nested loops

Guidelines

- All source codes (.c files) in the naming format `ID_Lab6_TaskN.c`, eg. `230041101_Lab6_Task4.c`.
- Ensure that your program produces the correct output for all sample cases.
- Source codes must be properly indented
- Screenshots are not required.
- No *Google Classroom* submissions are required for the `vjudge` contest, submit solutions directly in the `vjudge` platform.
- Participate in the `vjudge` contest after you are done with the other tasks, and solve as many problems as you can throughout the week. Do not copy any of the solutions from others.
- Throughout the rest of the semester, try to regularly solve problems and participate in contests on online judges such as *Codeforces*, *Codechef*, *Atcoder*.

Task 1 — Fibonacci Numbers

In mathematics, the **Fibonacci numbers**, commonly denoted as F_n, form a sequence called the **Fibonacci sequence**, where each number is the sum of the two preceding ones, starting from 0 and 1. The sequence is defined as follows:

$$F_0 = 0,\ F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2},\ \text{for } n > 1$$

The sequence starts:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$$

Write a C program that takes an integer n ($n \geq 1$), representing the sequence length as input, and then prints the Fibonacci sequence up to the n^{th} value.

Input

The input consists of a single integer n ($1 \leq n \leq 65$), representing the length of the Fibonacci sequence.

Output

The output consists of the Fibonacci sequence up to the n^{th} value, printed in a comma-separated format.

Sample Execution(s)

Input 1

```
Enter sequence length: 5
```

Output 1

```
Sequence: 0, 1, 1, 2, 3
```

Input 2

```
Enter sequence length: 1
```

Output 2

```
Sequence: 0
```

Hint: Use two variables to keep track of the last two numbers of the series. After getting the next number F_n, move F_{n-1} to F_{n-2} and F_n to F_{n-1}. In this way, continue building up the series.

Task 2 — Grade Yourself

Write a C program that would take marks of a student as input and print out his/her grade and grade point based on table 1.
You can only use `switch` statements.
`if...else` statements are **not** allowed.

Grade	Marks	GP
A	80 and Above	4.00
B	70 to 79	3.50
C	60 to 69	3.00
D	50 to 59	2.50
F	Below 50	0.00

Table 1: Reference Grade Sheet

Input

The input consists of one integer $marks$, $0 \leq marks \leq 100$

Output

The output consists of two lines. The first line will comprise of the student's grades, and the second line will comprise of his grade point.

Sample Program(s)

Input

```
Enter marks: 75
```

Output

```
Grade: B
Grade Point: 3.50
```

Hint: Use some arithmetic calculations to cut down the number of `switch cases` required drastically.

Task 3 — Let's Play A Game

Write a C program that selects a random number between 1 and 100, inclusive. The program should then allow the user to continuously guess the number, providing feedback after each guess to indicate whether the guess is higher, lower, or equal to the selected number. The game ends when the user correctly guesses the number.

Input

The input consists of multiple guesses, where each guess is an integer between 1 and 100.

Output

The output consists of feedback after each guess:
- Print `"Too high!"` if the guess is greater than the selected number.
- Print `"Too low!"` if the guess is less than the selected number.
- Print `"Correct! You win!"` if the guess is equal to the selected number, and terminate the game.

Sample Execution(s)

Sample 1

```
Enter your guess: 50
Too high!
Enter your guess: 25
Too low!
Enter your guess: 37
Correct! You win!
```

Sample 2

```
Enter your guess: 70
Too high!
Enter your guess: 30
Too low!
Enter your guess: 50
Correct! You win!
```

Hint: Use the `rand()` function to generate the random number. A sample code to generate a random number is given below

```c
#include <stdlib.h>
#include <time.h>

srand(time(0));  // Seed the random number generator
int randomNumber = rand() % 10;  // Generate random number between 0 and 9
```

A seed in the context of random number generation is an initial value used by a random number generator (RNG) algorithm to start the process of creating a sequence of ”random” numbers.

In this code `srand(time(0))` seeds the random number generator with the current time, so the sequence of random numbers is different every time the program runs.

`rand()` generates a random number, and taking it modulo 10 ensures that the output is between 0 and 9.