05OGDLP - Algorithms & Programming

Cristiano **Pegoraro Chenet**, Riccardo **Smeriglio**

Department of Control and Computer Engineering (DAUIN),
Politecnico di Torino, Italy

September 26, 2024

Laboratory 1

This assignment is designed to jog your programming and problem-solving skills.

Learning objectives

- **Problem solving**: Translate real-world scenarios into coded solutions, developing analytical and computational thinking.
- **Mastering control structures**: Use selection and iteration mechanisms effectively to drive your program's logic.
- **Adapting between languages**: Implement solutions in Python and then transition to C, understanding the core differences and similarities between the languages.
- **Analytical reflection**: Post-implementation, analyze the challenges encountered and the differences in coding between Python and C.

General guidelines

- Approach each exercise methodically. Consider the objectives and aim for efficient, clear solutions.
- Ensure that each program provides clear instructions for the user.
- Implement error handling to manage potential invalid inputs.
- First, develop your solution using Python. Once you have tested its functionality, try implementing it in C.
- Consistent effort is paramount!
  - Attend and engage in lectures and lab sessions.

– Dedicate time outside the classroom for practice, self-study, and reviewing materials.
– If challenges arise, communicate with us promptly! Remember, you have tutoring hours for this purpose.

1 Lab Exercises

1.1 Number guessing game

Code a game where the computer selects a random number between 1 and 100, and the player has to guess it. After each guess, the computer must inform the player if their guess was too high, too low, or correct. Keep track of how many attempts the player makes. On a successful guess, show the player the number of tries they took.

```
Number Guessing Game
--------------------
I have chosen a secret number between 1 and 100.

Your guess: 50
Hint: Too low!

Your guess: 75
Hint: Too high!

Your guess: 63
Hint: Just right!

Congratulations! You guessed the number in 3 tries.
```

1.2 Simple ATM Interface Simulation

You're tasked with building a basic ATM interface for virtual banking. Start with a predefined balance. Users should have the option to:

- Deposit funds
- Withdraw funds, keeping in mind that you can't draw more than available
- View their current balance.

Note: Exploit the `switch` construct to create the menu.

```
A&P Bank Virtual ATM
--------------------

Current balance: 500 EUR
```

```
Please choose an option:

1. Deposit funds
2. Withdraw funds
3. Check balance
4. Exit

Your choice: 1

Enter the amount you wish to deposit: 150

The transaction is successful! New balance: 650

Please choose an option:
1. Deposit funds
2. Withdraw funds
3. Check balance
4. Exit

Your choice: ... [continues]
```

1.3 The Nim game

Nim is a well-known game with various versions. The version described here has a particularly fascinating winning strategy. Two players alternatively pick marbles from a pile. In each turn, a player decides how many marbles to take: at least one but no more than half of the available marbles. Then it's the other player's turn. The player who takes the last marble loses the game.

Write a program that allows the user to play against the computer. Generate an integer between 10 and 100 to use as the initial pile size of marbles. Generate another integer, either 0 or 1, to determine if the user or computer will play first.

Also, generate an integer, 0 or 1, to decide if the computer will play smart or randomly:

- In random mode, during each of its turns, the computer simply takes a random (but valid) number of marbles (between 1 and n/2 if n marbles are left).
- In smart mode, it takes a number of marbles so that the remaining ones in the pile are a power of two minus one - that is, 3, 7, 15, 31, or 63. This move is always valid unless the pile size is precisely a power of two minus one; in that case, the computer chooses randomly among valid moves.

```
Welcome to the Nim Game!
----------------------------

The initial pile has 57 marbles.
you will play first.

Your turn! How many marbles will you take (1-28)? 7

Marbles left: 50

Computer took 15 marbles.

Marbles left: 35

Your turn! How many marbles will you take (1-17)? 7

Marbles left: 28

Computer took 1 marble.

Marbles left: 27

... [game continues]

Oops! You took the last marble. You lose. Better luck
next time!
```

As you can experimentally verify, the computer can't be beaten when it plays smartly and goes first, unless the initial pile size is 15, 31, or 63. Similarly, a human player familiar with this strategy and going first can beat the computer.

2 Homework

2.1 Hamming distance in DNA strands

DNA strands consist of sequences using the letters C, A, G, and T. When cells divide, DNA replicates, occasionally introducing errors in the sequences.

The Hamming Distance quantifies these errors by counting the number of differences between two DNA strands. Put differently, it measures the minimum number of substitutions required to transform one string into another.

Hamming Distance is applicable only for strings of equal length. Your program should not compute it for strings of different lengths and should handle such cases gracefully. Assume all strings manipulated by your program are 100 characters long at most.

Listing 1: Example of program execution when correct inputs are provided.

```
Hamming Distance Calculator for DNA Strands
-------------------------------------------

 Please enter the first DNA strand: GAGCCTACTAACGGGAT
 Please enter the second DNA strand: CATCGTAATGACGGCCT

 Calculating Hamming Distance...

DNA Strand 1: GAGCCTACTAACGGGAT
DNA Strand 2: CATCGTAATGACGGCCT

 Differences:  ^ ^ ^  ^ ^    ^^

 Result: The Hamming Distance between the two DNAs is 7.
```

Listing 2: Example of program execution when a wrong input is provided.

```
Hamming Distance Calculator for DNA Strands
-------------------------------------------

 Please enter the first DNA strand: XYZ

 Error: DNA strands consist of the letters C, A, G, and T.

 Please enter the first DNA strand: ... [continue]
```

2.2 Binary to decimal conversion

Convert a binary number, represented as a string (e.g. 101010), to its decimal equivalent using first principles.

Given a binary input string, your program should produce a decimal output. The program should handle invalid inputs.

Decimal is a base-10 system. A number, e.g. 23, in base 10 notation can be understood as a linear combination of powers of 10:

- The rightmost digit gets multiplied by $10^0 = 1$
- The next number gets multiplied by $10^1 = 10$
- ...
- The nth number gets multiplied by 10^{n-1}
- All these values are summed.

So:

$$23 \Rightarrow 2 \cdot 10^1 + 3 \cdot 10^0 \Rightarrow 2 \cdot 10 + 3 \cdot 1 = 23 base 10$$

Binary is similar, but uses powers of 2 rather than powers of 10:

$$101 \Rightarrow 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \Rightarrow 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 4 + 1 = 5 base 10$$

Note: Implement the conversion yourself. Do not use something else to perform the conversion for you.

Acknowledgment

The teachers would like to thank Leonardo Giannantoni for writing the first version of these exercises for last year's edition of this course.