Introduction à l'algorithmique **Winter 2024**

IFT2125-6001 Evaluation (7.5%)

Assignment 1 - Prime numbers, what child's play!

General instructions: To be done in teams of 2 people maximum. Submit a single pdf and the python and C++ code files. Indicate your name and registration number (matricule) on all files (pdf and code).

Your code must pass the tests provided with the assignment, otherwise a grade of 0 will be assigned to exercises whose basic tests are not passed. Other tests will be added during the correction. Only standard Python and C++ libraries are allowed, unless otherwise stated.

It goes without saying that the code must be clear, well indented and well commented.

1 Asymptotic Notation (25 points)

Question 1: (10 points) Using the definitions of O, Ω and Θ seen in class (without using limits), show or disprove the following statements:

1. $2^n \in \Theta(3^n)$
2. $2^{n+b} \in \Theta(2^n)$ where $b \in \mathbb{N}^{\geq 2}$

Question 2: (10 points) Using the limit rule, determine the relative order (O, Ω, or Θ) of the following functions:

1. $f(n) = 2^n$ and $g(n) = 3^n$
2. $f(n) = \frac{n}{\ln n}$ and $g(n) = \sqrt{n}$

Question 3: (5 points) Prove that the following function is smooth :

$$f(n) = 2n^2 - 3n - 4$$

2 Prime numbers - C++ Code (10 points)

Find the nth prime number in C++. Your algorithm should be efficient to get all the points.

Code
Call example :

```
Compilation :
g++ -o nth_prime.exe nth_prime.cpp PrimeCalculator.cpp

Execution :
.\nth_prime.exe 100
```

Submission
Complete the *PrimeCalculator.cpp* and *PrimeCalculator.h* files provided, and submit **only** these files. Do **not** submit the *nth_prime.cpp* file.

3 MST - Python Code (20 points)

Compute the weight of a Minimum Spanning Tree (MST) in Python.

Code
Your code must read a file specified as first argument that contains n problems. Each problem contains a number of coordinates m followed by these coordinates. The coordinates are all on a Euclidean plane. All edges between each pair of vertices are possible. The weight of an edge is the Euclidean distance between the two coordinates. The response returned should only be the weight of the MST. Write the answers to the n problems, line by line, in a file specified as the second argument.

Call example :

```
python3 acm.py input.txt output.txt
```

A test file *test_acm.py* and input files are provided to help you test your code.

Call example :

```
python3 test_acm.py
```

Submission
Complete the *acm.py* file provided, and submit **only** this file. Do **not** submit the *test_acm.py* file, or any other test file.

4 Dobble Game - Python Code (45 points)

You have to recreate the Dobble (or Spot It!) card game. It's a game of speed and observation.

The game is based on the concept of finite projective planes in mathematics. Each card contains a certain number of symbols. Each pair of cards always has one and only one symbol in common.

We say that the order of a game is n. The number of symbols per card is $n + 1$. The total number of symbols and also cards in the game is $n^2 + n + 1$. For this assignment, we will only be interested in orders that are prime numbers. Example :

- order = 2, symbols per card = 3, total symbols and cards in the deck = 7
- order = 3, symbols per card = 4, total symbols and cards in the deck = 13
- order = 5, symbols per card = 6, total symbols and cards in the deck = 31
- order = 7, symbols per card = 8, total symbols and cards in the deck = 57
- etc.

Explanations of the card generation algorithm

To correctly generate the cards for a game of order n, we can construct an array $n \times n$ of empty cards and a second array $(n+1) \times 1$ of empty cards which will represent " the horizon" of the projective plane. For each possible direction and for each start card, we will assign a symbol common to all the cards on this line and for the card on the horizon which represents this direction. Finally, we need to assign a symbol for all the cards on the horizon.

See the example illustrated in the appendix.
Wiki reference for those interested: Projective_plane#Finite_projective_planes

Code
Three classes will be used :

- Generator (dobble_generator.py): Receives the game order and constructs the cards symbolically. The result is written to the file "cartes.txt" where each line represents a card. Symbols are simply written as a list of numbers separated by spaces. A random mix of a card's symbols will be appreciated.
- Verificator (dobble_verificator.py): Receives a file containing the cards then verifies the validity and optimality of the game. With an order n, there should be $n+1$ symbols per card and n^2+n+1 cards and symbols in total for the game to be considered optimal. The number of symbols per card should be the same for all cards for the game to be valid. Each pair of cards always share one and only one symbol in common for the ju to be valid. The check returns 2 if the set is invalid, 1 if the set is valid but not optimal, and 0 if the set is valid and optimal.
- Creator (dobble_creator.py): Receives a "cards.txt" file containing the cards then creates the visual cards of the game. The images are taken from the "images" folder: "1.png2, "2.png" , "3.png", ... "⟨N⟩.png". The visual cards are saved in the "results" folder: "card1.jpg", "card2.jpg", "card3.jpg", ... "card⟨N⟩.jpg". On each card, images corresponding to the symbols on the card are placed. Image rotations will be appreciated. Test images will be provided.

The PIL library will be used to perform image manipulations:

```
from PIL import Image
```

Call example :

```
python3 cartes_dobble.py 5
```

Submission
Complete the *dobble_generator.py*, *dobble_verificator.py* and *dobble_creator.py* files provided, and submit **only** these files. Do **not** submit the *maps_dobble.py* file, or any other test/image file.

Bonus 10%
We will grant a maximum bonus of 10% (for the assignment) to teams who present in person a printed and personalized version of their deck of cards. The game must be produced by your own algorithm and the images must be different from the example images provided with the assignment and then different from the other teams. The game must be at least order 5 and must be optimal (6 symbols per card, 31 symbols in total, 31 cards in total). The cards must be cardboard, so no standard printer paper. You can then give it as a gift to your friends or family.

You can use the printing services of SIUM at the University of Montréal or of a company. Details for those interested will be provided later.

Appendix