IFT2125-6001 Évaluation (7.5%)

Homework 4 - The Last Mile

General instructions : To be done in teams of 1 to 2 people. Submit only the python code files. Indicate your name and registration number (matricule) on all files.

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 libraries are allowed, unless otherwise stated.

It goes without saying that the code must be clear, well indented and well commented. **The submission date is Tuesday April 23rd at 10:30 p.m.** The structure of the submission in studium must be as follows:

```
StudiUM
├── well_placement.py
├── school_group_separation.py
└── prime_friend_groups.py
```

1 Technological progress ?=social progress - Discussion (0 points)

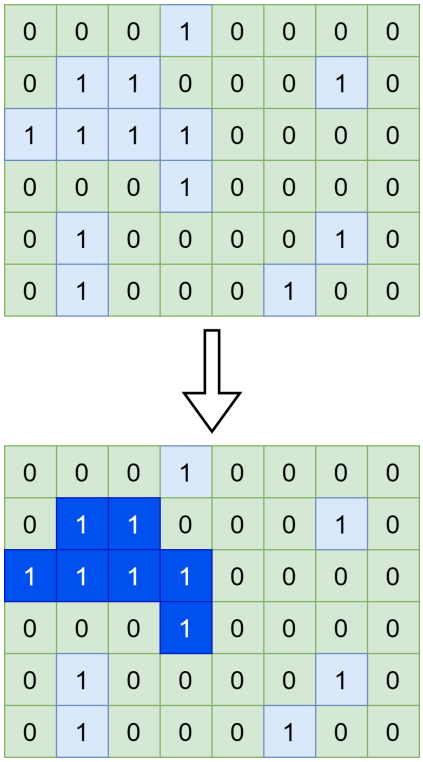
We're very proud when we discover efficient algorithms to solve problems, or when our implementation is widely used. But does technological progress always mean social progress?

Talk to a classmate for a few minutes about cases where algorithms (and the software that includes them) have had negative effects on some aspect of our society:

- Economy
- Environment
- Equality
- Justice
- Physical Health

- Mental Health
- Etc.

2 Well, where can I put my well? - Python Code (30 points)



Your mandate is to find the best location for a water well on a plot of land. You have geographical data indicating the presence of aquifers (water reserves) under the ground. This data divides the land into a grid, with each cell having a value of 1 if an aquifer lies beneath the ground and 0 if there is no trace of an aquifer in that cell.

We want to locate the well where the water reserve will be the greatest, so we try to identify the largest aquifer. In this problem, we assume that an aquifer is a set of cells that are connected vertically (top and bottom) and horizontally (left and right) on the grid. Diagonal boxes are not connected. You are asked to return only the size of the largest aquifer. If there is no trace of water, then 0 is returned.

Data :

```
<height> <width>
<1st row 0/1>
<2nd row 0/1>
...
<last row 0/1>
```

Examples :

```
Example 1 :
3 4
0001
1100
1011
--> should return 3

Example 2 :
3 4
0000
0000
0000
--> should return 0

Example 3 :
10 20
00000000000000000000
00110000001000000000
00110000000000000000
00000000000001000000
00000000000000000000
00000001000000000010
00000000000000000001
00000000000000000111
00000000000000010011
00000000000000001111
--> should return 10
```

Code

Your code must read a file specified as the first argument that contains a problem and write the answer to a file specified as the second argument.

Example call:

```
python3 well_placement.py input.txt output.txt
```

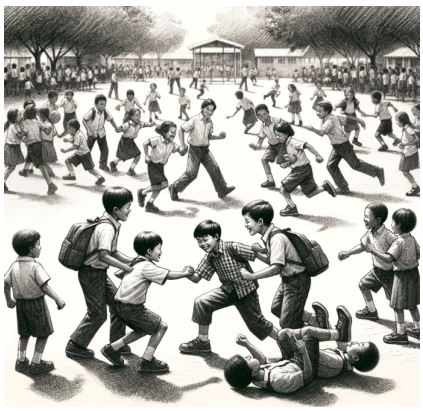
An *test_well_placement.py* file and input files are provided to help you test your code.

Example call:

```
python3 test_well_placement.py
```

Submission
Complete the *well_placement.py* file provided, and **only** submit this file. Do **not** resubmit the test file *test_well_placement.py*, or any other test file.

3 Unruly schoolchildren - Python Code (30 points)



After your success with youngsters creating card games and mazes, a Montreal elementary school consults you about a problem with unruly schoolchildren.

Teachers have noted that certain pairs of pupils are dangerous: when you put two children from a dangerous pair in the same group for an activity, you can almost certainly expect behavioral problems to arise at some point.

With a list of schoolchildren and a list of dangerous pairs, determine whether the children can be separated into two groups such that each dangerous pair is separated between the two groups. If possible, return a combination of children in two groups that works, otherwise return "impossible".

Data :

```
<n number of kids>
<name 1>
<name 2>
```

```
...
<name n>
<m number of dangerous pair>
<name A> <name B>
<name C> <name D>
...
```

Examples :

```
Example 1 :
5
Isabella
Jacob
Kaitlyn
Liam
Mohammed
2
Kaitlyn Mohammed
Liam Isabella

--> must return (among other valid answers)
Isabella Jacob Kaitlyn
Liam Mohammed


Example 2 :
3
Noah
Olivia
Parker
3
Noah Olivia
Olivia Parker
Parker Noah

--> must return
impossible
```

Code

Your code must read a file specified as the first argument that contains a problem and write the answer to a file specified as the second argument.

Example call:

```
python3 school_group_separation.py input.txt output.txt
```

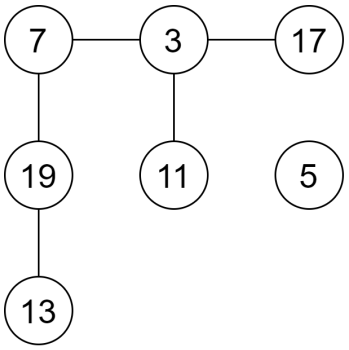
An *test_school_group_separation.py* file and input files are provided to help you test your code.

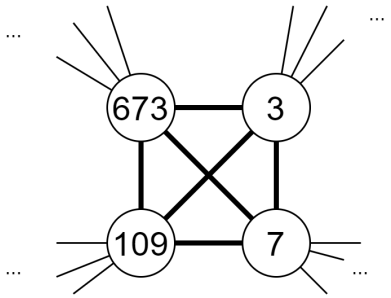
Example call:

```
python3 test_school_group_separation.py
```

Submission
Complete the *school_group_separation.py* file provided, and **only** submit this file. Do **not** resubmit the test file *test_school_group_separation.py*, or any other test file.

4 Prime numbers special sets - Python Code (40 points)





Let p and q be two prime numbers. This pair of numbers is said to be special when the concatenation of the two numbers in any order also gives a prime number. For example, 3 and 7 form a special pair because 37 and 73 are prime. On the other hand, 2 and 3

do not form a special pair because 23 is prime but 32 is not. Note that 307 cannot be considered a concatenation of 3 and 7 because of the extra 0.

A set of primes is special when all pairs of numbers in the set form special pairs. In graph theory, this is called a clique.

To characterize such a set, we can look at the sum of all its elements. For example, the special set of $(67, 3, 37)$ has a sum of 107. This characterization is not necessarily unique.

Find the set of 4 primes that is special and whose sum is the nth among all the special sets of 4 primes. $1 \leq n \leq 7$. You must return this sum of the nth smallest special set in the file given as an argument. Your code must find and calculate these values and not keep the pre-calculated answers in the code.

For this exercise, you'll be asked to implement Miller-Rabin's primality test function, which will greatly speed up certain calculations (primality tests will undoubtedly take up most of the computation time). This algorithm for checking the primality of a number is a probabilistic Monte Carlo algorithm. On the other hand, for limited values ($< 10^{24}$), it has already been determined that all you need to do is test with a specific set of numbers to obtain a deterministic algorithm!

Don't be surprised if the code takes a few tens of seconds to run.

For example, here are the first 3 special sets of 4 prime numbers in order of increasing total sum :

```
(3, 7, 109, 673)
(3, 37, 67, 2377)
(7, 19, 97, 3727)
...
```

And their corresponding sums :

```
792
2484
3850
...
```

Data : none (only 2 parameters passed by arguments)

Code
Your code must read a file specified as the first argument that contains a problem and write the answer to a file specified as the second argument.

Example call:

```
python3 prime_special_groups.py 2 output.txt
```

Expected answer in the output file :

```
2484
```

Un fichier *test_prime_special_groups.py* et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code.

Example call:

```
python3 test_prime_special_groups.py
```

Important : These tests are for $n = 1, 2, 3, 6$. We ask you to please not publicly share your answers for $n = 4, 5, 7$. Thank you!

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

Bonus 10%
We will grant a maximum bonus of 10% (for the assignment). This bonus will be announced only after the assignment has been officially handed in. The bonus will be conditional on the successful completion of the targeted exercise(s) of this assignment and will be announced only after the assignment has been handed in. The bonus will also be conditional on the assignment being handed in on time.