Assignment 1

COMP9021, Trimester 2, 2023

1. General matters

1.1. **Aim.** The purpose of the assignment is to:

- develop your problem solving skills;
- design and implement the solutions to problems in the form of small sized Python programs;
- practice the use of arithmetic computations, tests, repetitions, lists, dictionaries, strings.

1.2. **Submission.** Your programs will be stored in files named

- `fish.py`,
- `pivoting_die.py` and
- `highest_scoring_words.py`.

After you have developed and tested your program, upload it using Ed (unless you worked directly in Ed). Assignments can be submitted more than once; the last version is marked. Your assignment is due by July 10, 10:00am.

1.3. **Assessment.** The assignment is worth 13 marks. It is going to be tested against a number of inputs. For each test, the automarking script will let your program run for 30 seconds.

Late assignments will be penalised: the mark for a late submission will be the minimum of the awarded mark and 13 minus the number of full and partial days that have elapsed from the due date.

The outputs of your programs should be **exactly** as indicated.

1.4. **Reminder on plagiarism policy.** You are permitted, indeed encouraged, to discuss ways to solve the assignment with other people. Such discussions must be in terms of algorithms, not code. But you must implement the solution on your own. Submissions are routinely scanned for similarities that occur when students copy and modify other people's work, or work very closely together on a single implementation. Severe penalties apply.

2. Fishing towns (4 marks)

Write a program, stored in a file named `fish.py`, that performs the following task.

- The program prompts the user to input a file name. If there is no file with that name in the working directory, then the program outputs an (arbitrary) error message and exits.
- The contents of the file consists of some number of lines, each line being a sequence of two strictly positive integers separated by at least one space, with possibly spaces before and after the first and second number, respectively, the first numbers listed from the first line to the last line forming a strictly increasing sequence. The first number represents the distance (say in kilometres) from a point on the coast to a fishing town further down the coast (so the towns are listed as if we were driving down the coast from some fixed point); the second number represents the quantity (say in kilos) of fish that has been caught during the early hours of the day by that town's fishermen. For instance, the contents of the file `coast_1.txt` can be displayed as

```
   5   70
  15  100
1200   20
```

  which corresponds to the case where we have 3 towns, one situated 5 km south the point, a second one situated 15 km south the point, and a third one situated 1200 km south the point, with 70, 100 and 20 kilos of fish being caught by those town's fishermen, respectively.
- The aim is to maximise the quantity of fish available in all towns (the same in all towns) by possibly transporting fish from one town to another one, but unfortunately losing 1 kilo of fish per kilometre. For instance, if one decides to send 20 kilos of fish from the second town to the first one, then the second town ends up having $100 - 20 = 80$ kilos of fish, whereas the first one ends up having $70 + 20 - (15 - 5) = 80$ kilos of fish too.
- The program outputs that maximum quantity of fish that all towns can have by possibly transporting fish in an optimal way.

Here is a possible interaction:

```
$ cat coast_1.txt
5 70
15 100
1200 20
$ python3 fish.py
Which data file do you want to use? coast_1.txt
The maximum quantity of fish that each town can have is 20.
$ cat coast_2.txt
20 300
40 400
340 700
360 600
$ python3 fish.py
Which data file do you want to use? coast_2.txt
The maximum quantity of fish that each town can have is 415.
```

You can assume that the contents of any test file is as expected, you do not have to check that it is as expected.

3. Pivoting die (5 marks)

Consider the board below, which can be imagined as being infinite, so only a small part is represented. A die is placed on cell number 1 of the board in the position indicated: 3 at the top, 2 at the front (towards cell number 4 of the board), and 1 on the right (towards cell number 2 of the board). Recall that 4 is opposite to 3, 5 is opposite to 2, and 6 is opposite to 1. The die can be moved from cell 1 to cell 2, then to cell 3, then to cell 4…, each time pivoting by 90 degrees in the appropriate direction (to the right, forwards, to the left, or backwards). For instance, after it has been moved from cell 1 to cell 2, 2 is still at the front but 6 is at the top and 3 is on the right.

43	44	45	46	47	48	49	50
42	21	22	23	24	25	26	
41	20	7	8	9	10	27	
40	19	6	1	2	11	28	
39	18	5	4	3	12	29	
38	17	16	15	14	13	30	
37	36	35	34	33	32	31	

Write a program, stored in a file named pivoting_die.py, that prompts the user for a strictly positive integer N, and outputs the numbers at the top, the front and the right after the die has been moved to cell N.

Here is a possible interaction.

```
$ python3 pivoting_die.py
Enter the desired goal cell number: A
Incorrect value, try again
Enter the desired goal cell number:
Incorrect value, try again
Enter the desired goal cell number: -1
Incorrect value, try again
Enter the desired goal cell number: 0
Incorrect value, try again
Enter the desired goal cell number: 1
On cell 1, 3 is at the top, 2 at the front, and 1 on the right.
$ python3 pivoting_die.py
Enter the desired goal cell number: 29
On cell 29, 3 is at the top, 2 at the front, and 1 on the right.
$ python3 pivoting_die.py
Enter the desired goal cell number: 2006
On cell 2006, 4 is at the top, 1 at the front, and 2 on the right.
```

4. A word game (4 marks)

Write a program, stored in a file named highest_scoring_words.py, that performs the following task.

- Prompts the user to input between 3 and 10 lowercase letters (with possibly whitespace inserted anywhere); if the input contains too few or too many lowercase letters or any character which is neither a lowercase letter nor whitespace, then the program outputs an error message and exits.
- Finds in the file dictionary.txt, assumed to be stored in the working directory, the words built from the letters input by the user (with the exclusion of any other character) with highest score, if any; the score of a word is defined as the sum of the values of the letters that make up that word, the value of each letter being defined as follows:

a	2	b	5	c	4	d	4	e	1	f	6
g	5	h	5	i	1	j	7	k	6	l	3
m	5	n	2	o	3	p	5	q	7	r	2
s	1	t	2	u	4	v	6	w	6	x	7
y	5	z	7								

- Outputs a specific message if there is no such word; otherwise, outputs the highest score and all words with that score, one word per line, with a different introductory message depending on whether there is a unique such word (in which case the introductory message is on the same line as the word) or at least two such words (in which case the introductory message is on a line of its own and all words are preceded with 4 spaces).

Here is a possible interaction.

```
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: abc2ef
Incorrect input, giving up...
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: ab
Incorrect input, giving up...
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: abcdefghijk
Incorrect input, giving up...
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: zz zz zz
No word is built from some of those letters.
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: a  a   a
The highest score is 2.
The highest scoring word is a
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: a e i o u
The highest score is 8.
The highest scoring words are, in alphabetical order:
    iou
    oui
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: prmgroa
The highest score is 24.
The highest scoring word is program
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: a b  e    o    r   a t
The highest score is 16.
```

```
The highest scoring word is abator
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: r a m m o x y
The highest score is 17.
The highest scoring words are, in alphabetical order:
    mayor
    moray
    moxa
    oryx
$ python3 highest_scoring_words.py
Enter between 3 and 10 lowercase letters: eaeo rtsmn
The highest score is 17.
The highest scoring words are, in alphabetical order:
    matrons
    transom
```