COMPETITION BOOKLET

McGill Engineering Competition 2023

PROGRAMMING

McGill Engineering Competition 2023

Sponsored by:

ST-JEAN & fils
CONSTRUCTION

ABB

DRW

McGill
McGill Engineering Student Center

SSMU

Tables of contents

1. Context

Welcome to the Canadian Space Agency (CSA) " pee-wee " recruitment program . As a potential recruit, you need to stand out from your colleagues. To do this, you are tasked with solving a series of challenges as a team. These challenges will allow you to demonstrate your abilities to solve new challenges, but also to showcase your qualities as a teammate. As individualism is the number one enemy of the space program, you will be evaluated as a whole; the important thing is that your entire team performs well.

2. Goals

In a team of 4, you will have to solve the four given challenges. Each challenge is worth exactly 25% of the final grade, so they are all as important as each other.

At the end of the design period, you will have to present your solutions in front of judges. This will involve explaining your algorithm clearly and concisely.

3. Challenge progress

3.1 Maze

To ensure your ability to orient yourself in the coordinate system chosen by the CSA, you will have to produce a maze solving algorithm projected onto a map of hexagonal boxes.

Hexagonal boxes are represented according to the hexagonal effective coordinate system. In this system, the position of each box is specified using 3 variables; a, r and c. For more information, refer to the Wikipedia page :

- https://en.wikipedia.org/wiki/Hexagonal_Efficient_Coordinate_System

Each maze has a start square, an end square, walls and movable surfaces. The goal is simple, start from the beginning to get to the end.

Mazes are stored in json format according to the following convention:

- Each object has a "_jsontype_" attribute which specifies the type of the serialized object. The 3 types of objects are "Labyrinth", "HexTile" and "TileType".

- **Labyrinth** :

    - “_jsontype_”: “Labyrinth”

    - “tiles”: List of pairs of (“HexTile”, “TileType”)

- **HexTile** :

    - “_jsontype_”: “HexTile”

    - “a”: Integer between 0 and 1

    - “r”: Unbounded integer

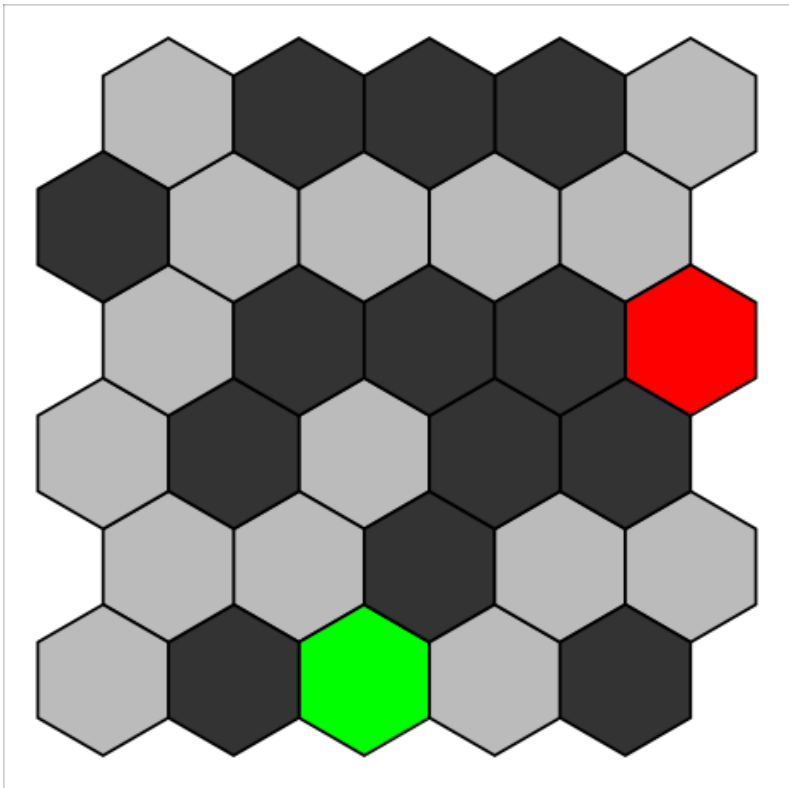
    - “c”: Unbounded integer

- **TileType** :

    - “_jsontype_”: “TileType”

    - “type”: { TileType.NORMAL, TileType.WALL, TileType.START, TileType.END }

To make sure you have a good representation of the maze, here is a visual example of the maze “easy/labyrinth_00.json”. Green is the start square and red is the end square.



3.1.1 Output format

Write the coordinates of the route in a “solutions” folder following the same structure as the “labyrinths” folder. The solution must be the complete list of walked boxes (including the start and end) specifying only their coordinates “(a, r, c)”. There should only be one coordinate per line. The solution file name must be the same as the maze, for example, the solution to the challenge “./labyrinths/easy/labyrinth_00.json” will be found in the file “./solutions/easy/labyrinth_00.txt”.

For example:

(0, 5, 6)

(0, 5, 7)

…

The score for this challenge will be calculated as follows:

40% of points for the result: this is the percentage of mazes completed. Each maze is worth exactly 1 point, there is no weighting according to difficulty level.

40% for the test: you will be graded as much on the readability of your code as on the quality and ingenuity of your algorithm.

20% for presentation: you will be required to present your solution in front of judges. A visualization of some solutions is strongly recommended.

3.2 Monopoly

In space, you can never be too careful. CSA evaluators want to test your ability to estimate complex data, while reducing the risk of an incorrect estimate causing a catastrophic accident to zero. Welcome to the world of admissible heuristics!
We give you the status of a Monopoly game where 4 players have already completed 5 turns each.
The information is given to you in a text file (in.txt) in the following format:

28 lines of the form:

Box number	Box name	Number of houses (5 for a hotel)	Is mortgaged (0 if no, 1 if yes)	Owner (missing if not yet purchased)

Followed by 4 lines of the form:

Player name (A, B, C or D)	Remaining budget	Number of square occupied by the player

The file contains 32,000 lines, or information relating to 1000 different games.
Standard rules from the original Monopoly apply, and the Chance[1] and Common Chest[2] cards[3] are also the same as in the original game.

The only differences are:

- Players cannot trade or bid (Land will always be purchased if the player has the necessary cash budget. Otherwise, the land will remain neutral).
- Players start the game with the same amount of money, different for each game and chosen randomly between $500 and $2500.

For each game, you will need to estimate the amount of money the players started the game with. **BE CAREFUL**. **Under NO circumstances** should you overestimate the amount. So you must have the exact value (almost impossible) or underestimate the value.

Place your estimates in a monopoly/out.txt file, writing one integer per line for each part.

The score for this challenge will be calculated as follows:
- 40% of points for the result: The sum of the differences between the true values and your estimates for each part, normalized by dividing it by the score obtained by the participant with the best result.
- 40% for the test: you will be graded as much on the readability of your code as on the quality and ingenuity of your algorithm.
- 20% for presentation: you will be required to present your solution in front of judges.

3.3 Hamming

The challenge consists of decoding a series of messages coded in the following way:

- Conversion of the message into binary using the ASCII standard (8 bits per character)

- The full binary message is permuted using a permutation matrix of random size between 2 and 10 (inclusive). This matrix is provided at the beginning of the message in its “flat” form; each of the rows is put one behind the other to form a vector of size n^2. IMPORTANT: The given matrix is the one used to encode the message.

[1] https://www.hasbro.com/common/instruct/00009.pdf
[2] https://monopolyguide.com/traditional/monopoly-list-of-community-chest-cards-main-version/
[3] https://monopolyguide.com/traditional/monopoly-list-of-chance-cards-main-version/

- Finally, encoding of the binary in Hex-CQI, which is a base 17 system. The characters making up the base are the 16 characters of hexadecimal (0-9A-F) and CQI, which must be considered as the 17th symbol from the base.

Each message to be decoded is on a line in a hamming/in.txt file, separated by a “\n”. Each line is composed of a permutation matrix and the encoded message. These two elements are separated by a “:” .

For example, if you see the following line:
“0100000100101000:11d48ed9dCQIc6ab6c6147d845e586da03b9”

The resulting permutation matrix should be:

0	1	0	0
0	0	0	1
0	0	1	0
1	0	0	0

and the output message should be “Hello world!”

The output format is a file consisting of all decoded messages, each on its own line and in the order they are found in the encoded file. To skip a message that your algorithm is unable to decode, simply leave the line blank. Please make sure to leave a blank line, otherwise the correction script will get lost and give a wrong response for each message that follows. Save this file as hamming/out.txt.

IMPORTANT : Make sure that reading the encoded message file and writing responses to the output file are all done in utf-8 .

The score for this challenge will be calculated as follows:

- 40% of points for the result: your decoded message file will be scored out of 808 messages then converted into a percentage. For example, a file of 404 valid messages would give a partial score of 50%, therefore a weighted score of 20%.
- 40% for the test: you will be graded as much on the readability of your code as on the quality and ingenuity of your algorithm.
- 20% for presentation: You will be required to present your solution in front of judges.

3.4 Shakespeare

Here's a fun logical sequence: -7; 1; -2; 3; -1; 2; 3; 11; 38; 423…

CSA evaluators want to validate your ability to write reports with eloquence and style. You will therefore have to write a script in SPL (Shakespeare Programming Language) which produces the following 5 elements of the sequence above when given the first two numbers as input (two numbers are sufficient to calculate the rest of the sequence). For example, with inputs -7 and 1, the script should produce -7 on different lines; 1; -2; 3; -1; 2; 3. Your script must be submitted in a shakespeare/script.spl file.

Note: This suite is just an example. In reality, due to interpreter limitations, no negative numbers will be given as input.

The score for this event will be calculated as follows:

- 40% of points for the result: your script will be tested to ensure that it accomplishes the requested task.
- 40% for the essay: This is about the quality of the prose in your script. You should feel like you're reading a play and not an algorithm.
- 20% for presentation: You will be required to present your solution in front of judges.

Relevant resources:

- https://en.wikipedia.org/wiki/Shakespeare_Programming_Language
- https://esolangs.org/wiki/Shakespeare
- https://esolangpark.vercel.app/ide/shakespeare (online interpreter)

4. Score sheet

Evaluation criteria	Scoring
Prototype (quality of the code, ingenuity of the solution)	**40**
Maze	10
Monopoly	10
Hamming	10
Shakespeare	10
Test	**40**
Labyrinth	10
Monopoly	10
Hamming	10
Shakespeare	10
Presentation	**20**
Eloquence and clarity of speech	10
Clarity and aesthetics of the visual support	10
Penalties	
Plagiarism	Elimination
Insufficient citation	-50
Documents not received on time	-50
Team member absent	-25
Failure to comply with the specifics of the submitted documents	-10
Entry into the presentation room before the permitted time (after the first offense)	-10
Total	**100**