# COMP 472/6721 Project Winter 2018

| | |
|---|---|
| **Due dates:** | February 15, 2018 + March 15, 2018 + March 29, 2018 + April 1, 2018 |
| **Late submission:** | 20% per day on each deliverable. |
| **Teams:** | Students registered in COMP 472 can do the project individually or in teams of at most 4. Students registered in COMP 6721 must do the project individually or in teams of at most 2. Teams must submit only 1 copy of the project. |
| **Purpose**: | The purpose of this project is to make you develop heuristics and state space search. |

## Candy Crisis

In this project you will implement a heuristic search to play a puzzle called Candy Crisis. Candy Crisis is a type of sliding puzzle played with 14 tokens representing different types of candies in a box. The box has 15 predetermined positions arranged in a box of 3 rows and 5 columns.

Initially, the player is given a random arrangement of candies in it, and the goal of the game is to rearrange the candies into a symmetrical arrangement by sliding them one at a time into the empty space. Only a candy directly next to the empty space, not including diagonally, can be moved into it.

A symmetrical arrangement (a goal state) is one which is balanced around the middle row, and there are no constraints on the middle row. In other words, if the top row is identical to the bottom row, then the puzzle is solved.

**A sample game:**
To illustrate the game, let's take a configuration with: 6 Reese's Pieces, 4 Bazooka Bubble Gums, 2 Walnettos candies and 2 York Minis candies. Given the following initial random box arrangement:

| | | | | |
|---|---|---|---|---|
| | r | r | r | r |
| r | b | w | b | b |
| w | y | b | r | y |

where, the Reese's Pieces are represented by 'r', the Bazooka Bubble Gums by 'b', the Walnettos by 'w' ...

To represent the positions in the box, let us use letters as follows:

| | | | | |
|---|---|---|---|---|
| A | B | C | D | E |
| F | G | H | I | J |
| K | L | M | N | O |

Since a candy can only move into the empty cell, all that is required is to record the candy that moved. The following 23 moves:

```
B C H M N O J I N M L K F A B C D I H C D E J
```

will reconfigure the candy box to the goal state:

| | | | | |
|---|---|---|---|---|
| r | w | y | r | b |
| r | b | r | b | |
| r | w | y | r | b |

However, the following 16 moves is a shorter (and better) way to reach a goal state from the same initial configuration:

```
B G L K F G L M N I D E J O N I
```

leading to the goal state:

| | | | | |
|---|---|---|---|---|
| r | b | r | r | b |
| y | w | w | | y |
| r | b | r | r | b |

**Your task:**
You must develop a program to solve the Candy Crisis puzzle automatically in the shortest number of moves. You are free to develop the search algorithm and heuristic that you want, but your program must solve a series of puzzles within a specific time limit. Puzzles will be categorized in 4 levels of difficulty:

| **Level** | **Candies** | **Maximum allowed time**<br>**(real time using the desktops in the lab)** |
|---|---|---|
| Level 1 – Novice | 6 Reese's Pieces +<br>6 Bazooka Bubble Gums +<br>2 Walnettos | 10 seconds for 50 puzzles |
| Level 2 – Apprentice | 6 Reese's Pieces +<br>4 Bazooka Bubble Gums +<br>2 Walnettos +<br>2 York Minis | 12 seconds for 50 puzzles |
| Level 3 – Expert | 4 Reese's Pieces +<br>4 Bazooka Bubble Gums +<br>2 Walnettos +<br>2 York Minis +<br>2 Gobstoppers | 30 seconds for 30 puzzles |
| Level 4 - Master | 4 Reese's Pieces +<br>2 Bazooka Bubble Gums +<br>2 Walnettos +<br>2 York Minis +<br>2 Gobstoppers +<br>2 Pez | 20 seconds for 10 puzzles |

**Input:**
Your program will read a series of initial configurations from a text file, where each line will represent an initial configuration. Each candy type will be represented by a unique character ('r', 'b', 'w', 'y', 'g' or 'p') and the empty slot will be indicated by the character 'e'. For example, the line:

```
e r r r r r b w b b w y b r y
```

represents the first puzzle given on the previous page
(empty at position A, Reese's Pieces at positions B, C, D, E, F,
a Bazooka Bubble Gum at G, a Walnetto at H, ...)

**Play Modes**
Your program should be able to run in manual mode and in automatic mode. This means that you should be able to run your program with:

1. *Manual entry for the player*, i.e. a human indicating the moves by hand.
   Note that if a human enters an illegal move, your program should give a warning and allow the user to enter a new move.
2. *Automatic mode,* i.e. your "AI" solving and indicating the moves.
   Note that if your program generates an illegal move, the entire puzzle solution will be considered wrong.

After each move, your program must display the new configuration of the candy box.

**Output:**
Your program must output:

*1- A set of puzzle solutions:* Your program must produce an output file, which contains, for each puzzle of the input file:

a) the sequence of moves necessary to solve the puzzle,
b) the time (real time in milliseconds in the lab's desktops) required to solve the puzzle

c) In addition, at the end of the output file, you must indicate:
d) the total number of moves it took to solve **all** the puzzles of the input file

*2- A visual trace:* In addition to the puzzle solutions, for each puzzle, your program must also display on the screen (or in another output file) the configuration of the candy box after each move.

**Programming details:**
You can use Java, C, C#, C++ or Python. If you wish to use another language, please check with me first.
A simple command-line interface is sufficient. No extra points will be given for a graphical user interface.

**Challenge:**
To make the project more fun, we will organize a challenge at the end of the semester. For each level of difficulty, an input file containing a series of initial configurations will be made available, and you will be able to run your program on this dataset and submit your results for evaluation. If, at any time, your program takes more than the required time to solve a series of puzzles, you will automatically be eliminated from the challenge. Teams who fulfill the challenge within the time frame specified will then be ranked according the number of moves used to solve the puzzles. Up to one bonus point will be allocated to your result in this challenge. More details about the challenge will be provided later in the semester.

**Oral presentation:**
Your final deliverable must be accompanied by an oral presentation of about 5 to 10 minutes. Your presentation should:

1. Describe your program (how to run it, what the main functions and data structures are …).
2. Describe and justify your heuristic. In particular, describe how you came up with your evaluation function. For example, what features of the game state you considered and why you chose them, how you balanced speed of the evaluation function with performance, and generally why you settled on the evaluation function you did.
3. Describe and explain your results at the challenge (why you think your heuristic makes good or bad decisions).

**Deliverables:** The submission of the project will consist of 4 deliverables:

| **Deliverable** | **Functionality** | **Due Date** |
|---|---|---|
| Deliverable 1 | Manual Mode<br>(no heuristic search) | February 15, 2018 |
| Deliverable 2 | Deliverable 1 + Automatic Mode<br>(search + first version of heuristic) | March 15, 2018 |
| Deliverable 3<br>(Challenge) | Deliverable 2 + Your Final Heuristic | March 29, 2018 |
| Deliverable 4 | Deliverable 3 + Oral Presentation | April 1, 2018 |

**Evaluation criteria:**
Students will be given individual grades that will be a function of the team grade and the peer-evaluation.

*Team grade:*

| **Deliverable** | **Points** | **Focus of the Evaluation<br>(see Moodle for more details)** |
|---|---|---|
| Deliverable 1 | 20% | Manual entries + Functionality of the rules of the game<br>(e.g. visual trace after each move, detection of illegal moves, programming quality…) |
| Deliverable 2 | 30% | Automatic Mode<br>(e.g. implementation of the search, output file generation, programming quality…) |
| Deliverable 3 | 20% | Results of the Challenge<br>(i.e. time limit, length of the solution path, programming quality…) |
| Deliverable 4 | 30% | Oral Presentation + Quality of the Heuristic<br>(e.g. slides, oral explanation, discussion of the heuristic, question-answering…) |

*Peer-Evaluation:*
All team members will be asked to fill-in a peer-evaluation form to evaluate the contribution of each team member. The grade of a student will be a function of his/her team grade and the peer evaluation received.

**Submission:** Each deliverable must be handed-in electronically **by 5pm** on the due date.

1. Make sure that you have signed the expectation of originality form (available on Moodle) and given it to me.
2. Hand in each deliverable electronically:
   - Create one zip file containing all files necessary to run your project – **including an executable file that runs on the lab's desktops.**
     - Name your zip file: *project_studentID* (for individual work) or *project_studentID1_studentID* (for group work)
     - Upload your zip file at: https://fis.encs.concordia.ca/eas/ as project1, project2, project3 and project4.

**Have fun!**