ESTR1002 2020 – 21 Term 1, Project Extension

Since we cannot have face-to-face final exam in this semester, we have arranged a project extension as discussed in the first lecture of this course.

- The project extension will contribute to **10%** of total course grade and
- you need to choose and complete **3 out of 6 extension items below** by modifying your code of the basic part of the project:

1. Step-by-Step Visualization

Currently, you are only required to display the game board after each swap operation. In this extension, you are required to display the game board in the intermediate steps, including the game board **after eliminating all numbers**, **after moving down the numbers** and **after generating the new numbers**, etc. You may use "0" to represent those empty cells on the game board.

2. Human player vs. AI player

Instead of having only one player to play the game, you are required to implement a competitive game that the human player and the AI player can **take turns** to perform the swap operations. The scores of one player is the total number of scores obtained by each player in the swap step of his/her turn. Your program should record the score of each player, and display their scores on top of the game board **after each swap**.

3. Random Generation of the "Stone Block"

In the original "Candy Crush" game, there is a special type of cell called "Stone Block", which **cannot** be swapped and eliminated. In this extension, you can include this type of block in your program and the block should satisfy the following requirements:

- First, before a game start, your program should be able to generate random number of such stones at random locations.
- Second, your program can detect that the swap operations that try to move a "Stone Block", and report an invalid step to the user.
- Also, having 3 "Stone" blocks in a row or column **will not** eliminate any blocks. This type of block should be displayed as **"@"** on the game board.

4. Random Generation of the "Choco Block"

Similar to the "Stone Block", there is another type of Candy in the original Candy Crush Game called the "Choco Block", which also cannot be swapped and eliminated. Furthermore, a "Choco Block" will lock its **4 neighbor numbers (i.e.,**

top, down, left, right), and does not allow them to be **swapped and eliminated**. In your implementation, you should treat each of the "Choco Block" and its four neighbors as a "Stone Block". However, instead of keeping all the "Stone Block" throughout the game, your program needs to randomly remove one existing "Choco Block" every 5 steps and also randomly grow (change a number to a "Choco Block") a "Choco Block" after the removal. This type of block should be printed as **"C"** in the game board.

5. Random Generation of an "Ice Block"

An "Ice Block" contains the same properties that the "Stone Block" has, but an "Ice Block" can "melt" (change to a number) if the numbers of its neighbors (i.e. top, down, left, right) are eliminated. This type of block should be printed as **"!"** on the game board.

6. Generation of Bomb

In the last extension, your program should be able to generate a special block, called "Bomb", when the user gets **more than 30 points in 1 step**. A "Bomb" can be swapped with any other neighbor numbers, and after swapping, **all the special blocks on the game board will be** eliminated. A "Bomb" should be printed as "B" on the game board. If you choose this item, you must also choose at least one of items 3-5 to ensure that you can test the features.

Grading:

Instead of integrating your extended functions into your basic program (see project specification), you need to prepare the project extension as a separate version of the program. **You have to submit the project extension to the blackboard by Dec 30** and besides, and need to do a demonstration of your program on the **project extension demo day** (Dec. 30).

On the demo day, The TA will evaluate your extension features one by one, by assessing the correctness of each extended feature, and asking your questions on your code. Note that to show the correct implementation of your feature, **you MUST design a scenario (i.e., a planned list of swap operations) by yourself**, and demo how your features work to the TA on the project demo day.