CCN2042 Computer Programming
Group Project – Block Puzzle Game
(Due: 18:00, 25 November 2017)

Expected Learning Outcomes

- familiarise themselves with at least one high level language programming environment.
- develop a structured and documented computer program.
- understand the fundamentals of object-oriented programming and apply it in computer program development.
- apply the computer programming techniques to solve practical problems.

Introduction

In this assignment, you are going to develop a **Block Puzzle Game** that runs in the command line environment. Player scores by putting blocks of different shapes onto a square game board, and by producing complete horizontal or vertical lines of blocks and clear those lines.

This is a **group assignment**. You need to form a group with **5** to **6** students, and write a Win32 Console Application program called **blockPuzzle.cpp**. The requirements are listed below.

System Requirements

R0 When the program starts, the console should **display a welcome message**, and then the Game Menu of the program. User can enter the options of the corresponding actions (see **R1** to **R5** below).

```
Welcome Message designed by your group

*** Game Menu ***
[1] Start Game
[2] Settings
[3] Instructions
[4] Credits
[5] Exit
*****************
Option (1 - 5):
```

R1 [1] Start Game

When the user inputs 1 in the Game Menu, the game starts with the current game mode and current game settings (see **R2**). There are two game modes: (1) the original mode and (2) the timer bomb mode. In both game modes, the player needs to put blocks onto a square game board to score, with the aim of producing complete horizontal or vertical lines of blocks in order to clear those lines. The game ends when the player fails to put any blocks onto the game board, or in addition when a bomb explodes (i.e., bomb timer expires) if it is under game mode (2).

You are free to design a good way to show the game status information and prompt messages. But the console should display at least the followings: (1) the total marks scored by the player, (2) the updated game board, and (3) a choose list with **THREE** randomly-generated blocks for player to choose.

R1.1 Game board

The player plays on a game board with a *default* size 10 × 10. The game board is enclosed by a boundary with characters 'l' for vertical boundary, '-' for horizontal boundary, and '+' for the corner. Each location in the board is identified by a letter and a number.

An example of board with default size (10 × 10) is given below, with the top-left location identified by A0:

```
   0 1 2 3 4 5 6 7 8 9
  +--------------------+
A |                    |
B |                    |
C |                    |
D |                    |
E |                    |
F |                    |
G |                    |
H |                    |
I |                    |
J |                    |
  +--------------------+
```

R1.2 Blocks

There are **19** types of blocks available in the game. The table below shows the blocks and their corresponding scores which the player can get after putting that block onto the game board.

Type	Horizontal blocks:				
Shape	X	X X	X X X	X X X X	X X X X X
Scores	10	20	30	40	50
Type	Vertical blocks:				
Shape	X X	X X X	X X X X	X X X X X	
Scores	20	30	40	50	
Type	L-shape blocks (1):				
Shape	X X X	X X X	X X X	X X X	
Scores	30	30	30	30	
Type	L-shape blocks (2):				
Shape	X X X X X	X X X X X	X X X X X	X X X X X	
Scores	50	50	50	50	
Type	Square blocks:				
Shape	X X X X	X X X X X X X X X			
Scores	40	90			

R1.3 Game Play – Putting blocks

For each move, there is a choose list with **THREE** randomly-generated blocks (with choice index 0 to 2) which the player can choose to put. The console should then display appropriate instructions that accept user's input of block choice and block position.

The input location should take the form with a row letter and a column number together (e.g., A0). It indicates where the **top-left 'X'** of a block is placed. For the "reversed L-shape" block (i.e., `X X` or `X X X`), the input location indicates where the **leftmost 'X'** of the block is placed. Note that blocks should be placed within the board and should not overlap with other blocks, and should not overlap with the bombs under game mode (2).

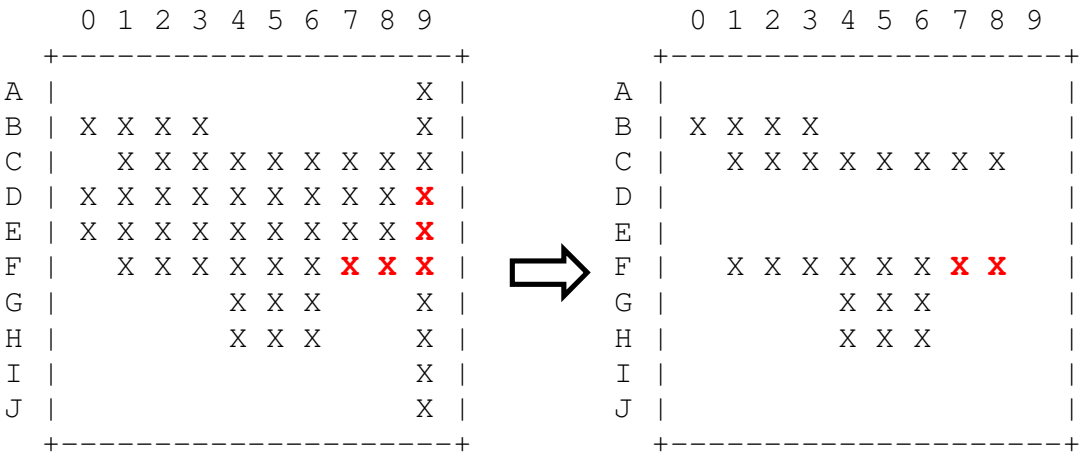
The game status information should be updated after getting the player's input: (1) the game board should be updated to show the block placement, (2) total score should be updated after one block placement, and (3) one new block should be randomly generated to replace the just-chosen one in the choose list. The player then proceeds to the next move.

R1.4 Game Play – Clearing lines

If the player puts a block that can successfully produce complete horizontal or vertical line(s), the line(s) will be removed. If this happens, the console should display how many lines are removed after the player's move, and update the game board accordingly. Extra scores as a result of forming complete lines (in addition to the score because of block placement) should be included according to the table below:

Number of complete lines formed in one move	Marks
1	100
2	300
3	600
4	1000
5	1500

The example on next page shows a scenario after a "reversed L-shape" block (`X X X`) is put at location F7, and three complete lines of blocks (row D, row E, column 9) can be removed.

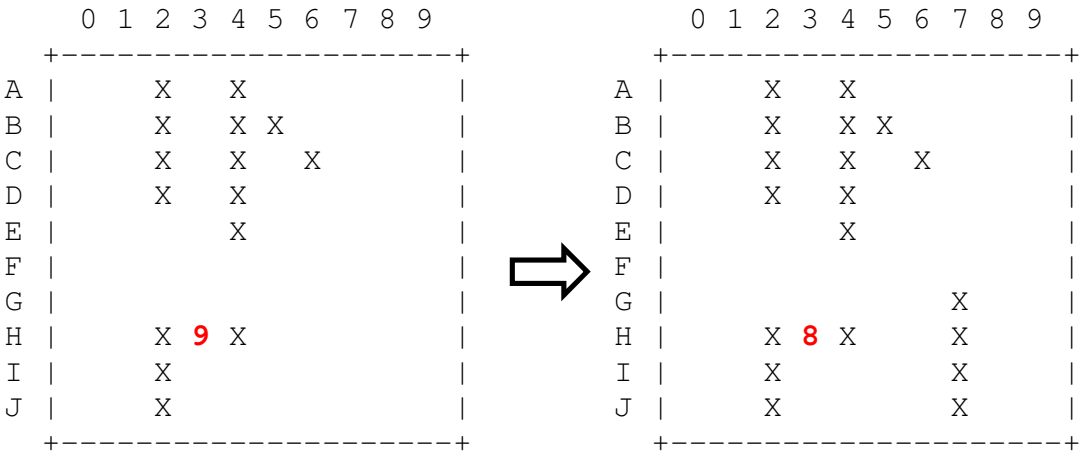


In the above example, the total score obtained after putting the "reversed L-shape" block should be 50 (scores of putting the block) + 600 (3 complete lines are removed) = 650.

R1.5 Game Play – Timer bomb mode

In timer bomb mode, for every **FIVE** player's moves, a bomb will be placed randomly on any board location that has been occupied by a block (i.e., 'X'). A bomb is represented by a digit (i.e., the bomb timer) on the game board, which starts from the value according to the current game setting (see **R2**) and decreases by 1 on each player's move. The player needs to put blocks and form a complete line across the bomb to clear it before the bomb timer decreases to 0. Otherwise, the player loses and the game finishes.

Below shows an example scenario that a bomb (with starting bomb timer 9) was placed at location H3 after the 5th move, and after the 6th move the bomb timer decreases to 8:



R1.6 Game play – Command "Quit"

Player can input a special command "Quit" to quit the game at any time during the game. If this happens, the system should prompts for player's confirmation. If the player inputs 'y' or 'Y', the game ends and the system returns to the Game Menu. If the player inputs 'n' or 'N', the game continues at its current status. Other input is not acceptable and the system should ask the player to confirm again.

R1.7 Game Play – Game ends

The game finishes when the player fails to put any blocks onto the game board. If the game starts in the timer bomb mode, in addition, the game finishes when a bomb explodes. If the game finishes, the final score should be displayed and the program returns back to the Game Menu.

R2 [2] Settings

When the user inputs 2 in the Game Menu, the console displays the Settings Menu. User can enter the options of the corresponding actions (see **R2.1** to **R2.5** below).

```
*** Settings Menu ***
[1] PC Game Demo
[2] Timer Bomb Mode
[3] Change Board Size
[4] Change Bomb Timer
[5] Return to Game Menu
**********************
Option (1 - 5):
```

R2.1 [1] PC Game Demo

When the user inputs 1 in the Settings Menu, the console displays the current setting of whether the PC plays a demo game, or the player plays the game (*default: player plays the game*), and prompts for the user input of the new setting.

After the setting is updated, it returns back to the Settings Menu.

If it has been set to having the PC playing a demo game, the demo game starts when the user enters [1] in the Game Menu. During the game, the user does not enter any block choice or block position. Instead, on each PC's move, the console should show clearly

the block choice and block position decided by the PC, and show all game information as described in **R1** above.

Move by PC can be randomly generated, but should follow the game rule.

R2.2 [2] Timer Bomb Mode

When the user inputs 2 in the Settings Menu, the console displays the current setting of whether the game starts in (1) the original mode, or (2) the timer bomb mode (*default: original mode*), and prompts for the user input of the new setting.

After the setting is updated, it returns back to the Settings Menu.

R2.3 [3] Change Board Size

When the user inputs 3 in the Settings Menu, the console displays the current setting of board size (*default: 10 × 10*), and prompts for the user input of the new setting. Since the board is in square shape, user is required to input one integer value only. For example, user inputs the value 8 to update the board size to 8 × 8.

Note that the acceptable board size is from 7 × 7 to 10 × 10 only. After the setting is updated, it returns back to the Settings Menu.

R2.4 [4] Change Bomb Timer

When the user inputs 4 in the Settings Menu, the console displays the current setting of the starting bomb timer value (*default value: 9*), and prompts for the user input of the new setting. Note that the acceptable starting bomb timer value is from 6 to 9.

After the setting is updated, the system returns back to the Settings Menu.

R2.5 [5] Return to Game Menu

When the user inputs 5 in the Settings Menu, the system returns back to the Game Menu.

R3 [3] Instructions

The system displays the instructions for playing the block puzzle game. After displaying the instruction, the system returns back to the Game Menu.

R4 **[4] Credits**

The system displays the personal particulars (e.g. student name, student ID, class, tutorial group, etc.) of the group members. After displaying the information, the system returns back to the Game Menu.

R5 **[5] Exit**

When the user inputs this option, the system prompts for user's confirmation. If the user inputs 'y' or 'Y', the program terminates. If the user inputs 'n' or 'N', the system returns to the Game Menu. Other input is not acceptable and the system should ask the user to confirm again.

Other General Requirements

R6 Meaningful guidelines should be printed to assist with user's input. Whenever an option is selected, meaningful messages should be displayed.

R7 Suitable checking on user's input is expected. Appropriate error messages should be printed whenever unexpected situation happens, e.g., invalid input, input out-of-range, etc.

R8 The use of functions (in addition to main function) and classes are expected in your program. Appropriate comments should be added in your source code file.

R9 Creativity and Critical Thinking: other features that you find useful or can enhance the user experience can also be implemented.

Submission

Source File: Each group has to submit one source code file (i.e., **blockPuzzle.cpp**) that implements this group project.

Peer-to-peer Evaluation: Each student has to fill in and submit the peer-to-peer evaluation form (i.e., CCN2042_P2P_Evaluation.docx) for your group.

All submission should be done **through Moodle by 18:00, 25 November 2017**. Late submission is subject to 20% deduction in your final marks for each day (including public holidays and Sundays). No late submission is allowed **4** days after the due date.

Grading criteria

Aspects	Percentage
Program correctness (Follow ALL instructions, marks deduction on errors found)	70%
Program design (Appropriate use of functions, use of class, modularity, etc.)	10%
Program standard (Use of variable names, indentation, line spacing, clarity, comments, etc.)	5%
Algorithm design (Use of reasonable algorithms and data structures)	5%
User-friendliness (Clear user interface, clear guidelines / messages to users, etc.)	5%
Creativity and critical thinking (Additional useful features)	5%
Total (Group Mark)	**100% (max)**

Note: the length of your program does not affect the grading of the assignment. However, appropriate use of loops and functions are expected to avoid too many repeated codes in your program, which contributes to the program design score of this assignment.

Individual mark is determined by both group mark and percentage of individual contribution, where the percentage of individual contribution is directly proportion to the average marks given by group members in the peer-to-peer evaluation form.

Individual mark is calculated by the formula below:

$$\text{Individual Mark} = \text{Group Mark} \times \text{Percentage of Individual Contribution}$$

Note: You may score 0 mark if you have 0% contribution.

Marks deduction

Marks will be deducted if the program fails to be compiled. The deduction is from **5 to 20** marks, depending on how serious the compilation error is. Note that if the program contains unacceptably too many serious compilation errors, your program will score **0** marks.

Compilation errors also lead to failure in the program correctness if the function cannot be tested. So please make sure that your program can be compiled successfully before your submission.

Tips

To refresh the console screen, you may use the following **windows** platform command appropriately in your program by adding the header file <stdlib.h>:

```
system("cls");   // Clear everything on the screen.
system("pause"); // Wait until user pressing any key to continue.
```

To handle unexpected input error (e.g. input a character to an integer variable), you may use the following code appropriately in your program:

```
cin.ignore();    // Discard the content in the input sequence.
cin.clear();     // Reset the input error status to no error.
```

***** Ensure the originality of your work. Plagiarism in any form is highly prohibited. *****

- End -