CCN2042 Computer Programming
Group Project – Battleship Game
(Due: 18:00, 21 April 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 **Battleship Game** that runs in the command line environment. Player plays against the PC by inputting commands to defeat the opponent's ships. You may find more information about the battleship game via this link: https://en.wikipedia.org/wiki/Battleship_(game).

This is a **group assignment**. You need to form a group with **5** to **6** students, and write a Win32 Console Application program called **battleship.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 settings (see **R2**). There are two stages in the game: (1) arrangement stage and (2) battle stage. In the arrangement stage, the player and PC secretly arrange their ships on their own "sea". After all the ships have been positioned, the game proceeds to the battle stage. Player and PC take turns to attack the opponent's sea, with the aim of destroying all the opponent's ships. The game ends when any one has all its ships being destroyed.

R1.1 Sea

Each player maintains a square sea with a *default* size 10 x 10. The sea is enclosed by a boundary with characters 'l' for vertical boundary, '-' for horizontal boundary, and '+' for the corner. Each location in the sea is identified by a letter and a number.

An example of sea with default size (10 x 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 Ship

Each player has a number of ships. There are four types of ship with different sizes. The default number of ships and their corresponding size are listed in the table below:

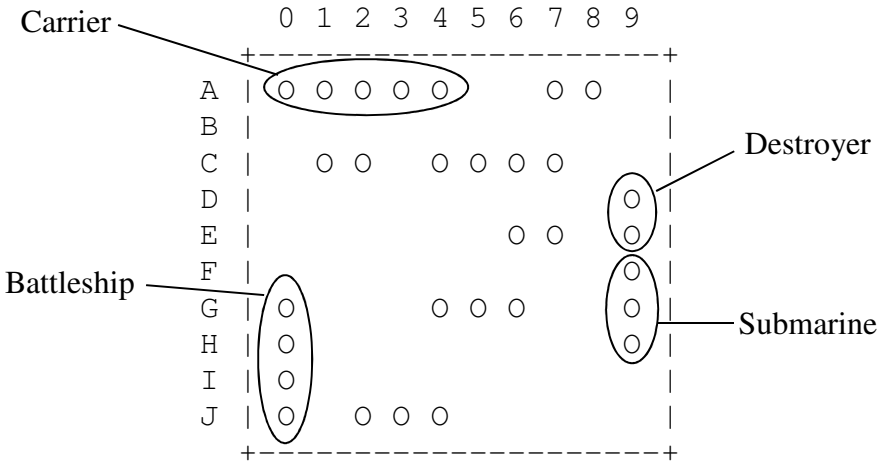
Ship	Size	Default number
Carrier	5	1
Battleship	4	2
Submarine	3	3
Destroyer	2	4

R1.3 Arrangement Stage

In the arrangement stage, each player arranges their ships on their own sea secretly. Each ship can be placed either horizontally or vertically on the sea, and occupies the required number of slots according to its size. Note that the ships cannot overlap with each other.

The game console should display appropriate instructions that assist player to arrange the ships one by one on the sea. Whenever a ship is placed on the sea, the occupied slots are marked with a capital letter 'O'. Then the player arranges the next ship until all the ships are placed on the sea.

Below is an example after all ships are arranged:



You may assume that PC arranges the ships randomly (but not overlapping) on its sea. When the player has arranged all the ships, the game proceeds to the battle stage.

R1.3 Battle stage

In the battle stage, the PC's sea is displayed with all ships are secretly hidden (*by default*), while the player's sea is displayed with ships on it. The number of remaining ships is also displayed with the corresponding sea.

Player and PC take turns (player first by default) to attack the opponent's sea by inputting the desired slot. The slot coordinates should be input together (e.g. A0). There are two possible cases:

(1) Attack on an empty slot. Then the slot is marked by an asterisk '*'.
(2) Attack on the ship body. Then the slot is marked by a capital letter 'H'.

When the entire ship has been marked by 'H', then that ship is destroyed. The number of remaining ships is reduced by one.

No matter which situation happens (i.e. a miss, a hit, or a ship is destroyed), appropriate message should be displayed to notify the player (e.g. A Submarine is destroyed).

R1.4 PC Intelligence (*with report, see Submission*)

Instead of randomly attacking the player's sea in every turn, PC should have a strategy to determine the locations of player's ships on the sea. You should design a strategic plan for PC, so that it can minimize the number of turns to win the game.

Note: PC does not know the locations of the player's ships.

R1.5 Command "Quit"

Player can input a special command "Quit" to quit the game at any time, in arrangement stage or battle stage. 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.

R1.6 Game Ends

The game continues until all the ships in a sea are destroyed. If all the PC's ships are destroyed, player wins. If all the player's ships are destroyed, PC wins. In any case, the game ends and the final result is displayed. Then it 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] Who starts first
[2] Display PC's ships
[3] Sea size
[4] Number of ships
[5] Return to Game Menu
**********************
Option (1 - 5):
```

R2.1 [1] Who starts first

When the user inputs 1 in the Settings Menu, the console displays the current setting of who starts first (*default: player first*), and prompts for the user input of the new setting.

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

R2.2 [2] Display PC's ships

When the user inputs 2 in the Settings Menu, the console displays the current setting of whether the PC's ships are displayed (*default: not displayed*), 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] Sea size

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

Note that the acceptable sea size is from 5 x 5 to 10 x 10 only, and the sea size should be large enough to hold all ships in it.

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

R2.4 [4] Number of ships

When the user inputs 4 in the Settings Menu, the console displays the current setting of the number of ships for each type (*default number given in R1.2*). The information should be displayed in the format below:

```
Type of ship         Number
Carrier    (size 5):   1
Battleship (size 4):   2
Submarine  (size 3):   3
Destroyer  (size 2):   4
```

Then, the player is prompted to input the new number of each type of ship. The acceptable number of each type of ship is from 1 to 5 only. Note that the total number of ships should not exceed the capacity of the sea.

After the setting is updated, it 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 battleship 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.

Submission

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

Report: Each group has to submit one report file that elaborates your **algorithm design** of the PC intelligence (in R1.4). The report should be **limited to 5 pages**, with the following 2 sections: (1) Brief Description of Algorithm, and (2) Flow Diagram.

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, 21 April 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 or 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%
User-friendliness (Clear guidelines to users, messages to users, etc.)	5%
PC Intelligence (Algorithm design and report writing)	10%
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:

Individual Mark = Group Mark x 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 -