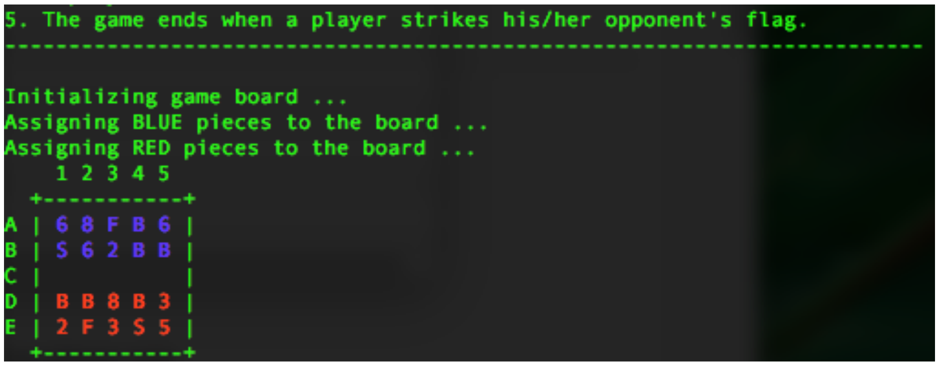
Riya Danait
CSCE 1030 Computer Science I – Section 002
Professor Helsing
November 20th, 2017

Algorithm for Homework 5

The calculations involved using the rand() and modulo functions to get the range of the numbers that are then mapped to the pieces or rows and columns. Note that if you want to get a random number in a range where the minimum is non-zero, you can do that by using rand() % (max – min + 1) + min, where [min, max] is the desired interval of numbers.



```
5. The game ends when a player strikes his/her opponent's flag.
------------------------------------------------------------------------

Initializing game board ...
Assigning BLUE pieces to the board ...
Assigning RED pieces to the board ...
    1 2 3 4 5
  +-----------+
A | 6 8 F B 6 |
B | S 6 2 B B |
C |           |
D | B B 8 B 3 |
E | 2 F 3 S 5 |
  +-----------+
```

Design Algorithm:

1. Include the iostream and stdio.h libraries.
2. Create a void function to print out name and information.
3. Create a void function to print out the rules of the game Stratego.
4. Create a constant global variable called BOARD_SIZE and set it equal to 5.
5. Define two enumerated type variables called Pieces and Color. In Pieces, define EMPTY = ' ', FLAG = 'F', BOMB = 'B', MARSHAL = '1', GENERAL = '2', COLONEL = '3', MAJOR = '4', CAPTAIN = '5', LIEUTENANT = '6', SERGEANT = '7', MINER = '8', and SPY = 'S' using the rules of the game. In Color, define RED = 'R', BLUE = 'B', and NONE = 'N'.
6. Declare three void functions: one to initialize the board, one to assign the pieces, and one to print out the board. Pass through 2 enumerated type arrays of Pieces and Color, as well as an integer variable representing the size of the array.
7. In the main function, call all of the functions, and define the 2 enumerated type arrays Pieces boardPieces and Color boardColor.
8. For the function that initializes the board, define two variables for the row and columns of the game board. Then create a for loop that is controlled by the int size that was passed through, and the int row that was created. Inside that for loop, make a nested for loop that is controlled by the ints size and column. Inside that loop, set the array boardPieces to EMPTY, and boardColor to NONE. The board is now initialized.
9. In the function assign pieces, declare and initialize the int col and row to be 0. Seed the random function.
    a. For the FLAG: We need to assign it to the back row (row 0) for the BLUE player. So, create a do-while loop that picks a random column between 0 and the size – 1 (meaning we mod the random number by size), and while the array boardPieces is not EMPTY, it sets the boardPieces to be FLAG, and boardColor to

be BLUE. Apply the same logic for RED, but instead of passing in 0 as the row, we pass in size – 1 (which would be 4 in a 5x5 array).

   b. For the rest of the pieces, there is no restriction on the location of the pieces (e.g. can be in either of the two back two rows for BLUE and RED).
   c. For the BOMBs: We need to randomly assign 3 BOMBs anywhere on the two rows for each side. We can create a for loop that runs three times, and inside that for loop we can create a do-while loop that picks a random row between 0 and 1 (meaning we mod the random number by 2), and a random column between 0 and size – 1 (meaning we mod the random number by size), and while the array boardPieces is not EMPTY, we set the boardPieces to be BOMB, and the boardColor to be BLUE. For the RED side, we apply the same logic, but for the row, we have to add size – 2 to the mod because we need to choose a random row between size – 2 and size – 1 (3 and 4 if size is 5).
   d. For the MARSHAL/GENERAL: We need to choose one of either of these pieces randomly, and assign them randomly to any of the two back rows. We can do the same steps as we did in (c) to assign the BLUE/RED pieces to any of the two back rows, but include an if-else statement to set a randomPiece to either MARSHAL or GENERAL.
   e. Assigning both the MINER and the SPY involve the exact same process used in (c).
   f. For the COLONEL/MAJOR/CAPTAIN/LIEUTENANT/SERGEANT: We can choose any 3 of these to assign to any place in the two back rows for both BLUE and RED. We can run a for loop 3 times, and each time, we can generate a random int called val. We can mod val by 5 and add 3 to get a random number in the range of 3 to 7 inclusive. Then we can pick a random row between 0 and 1 for the BLUE and mod it by 2. We can do the same for column, but mod it by size. Then while the boardPieces aren't EMPTY, we can add a switch statement that determines the piece that aligns with the random number val. We will also need to include a default statement that sets the randomPiece to EMPTY. Accordingly, we can set boardPieces to randomPiece, and boardColor to BLUE. We use the same approach for the RED pieces. When we mod the random number for row, we need to add size – 2, which will pick a random row between size – 2 and size – 1. Then we use the same switch statement for the RED as we did the BLUE.

10. For the last function, printing out the board, we need to pass in the two arrays and the integer size. Then we need to declare the row and column variables once more. We can start by printing the column header, which is just the numbers 1 through 5. We can do this with a simple for loop that is controlled by size and int i. We use printf to i each time the for loop increments i. Then we can print a new line. To print the top border, we can use the same for loop, but instead print out the dashes and plus signs. To print each row of the board, we print out the row name (A, B, C, D, E) which can be controlled by adding the ASCII value 65 ('A') to the row number to get the row name, and then print it as a character using static casting. We use the same code in printColor.cpp to actually print out the BLUE and RED pieces, else we print empty spots (e.g. entire row 2 in this case). Finally, we print out the bottom border in the same way we did the top border.