Sumer 2016

Board Game Application

Final Project – Milestone 1

Problem Description

Your job for this project is to write a board game program called “Checkout”.

Game Rules:

- Up to 8 players compete to be the first to obtain $1000
- Game takes place on a square board with 5 tile types
    - Empty Tile – No effect
    - Win Tile – Win a random prize
    - Lose Tile – Lose a random prize
    - Grand Prize Tile – Win a grand prize
    - Checkout Tile – Sells all your prizes for cash
- Players can roll 1 - 3 dice to determine how many tiles they move each turn
- If a player lands on the same tile as another player, that player steals a prize and moves 1 tile back

Project Development Process

Your development work on this project has four milestones and therefore is divided into four deliverables. The approximate schedule for deliverables is as follows

- Milestone 1 Due: Friday July 8th at 11:59PM
- Milestone 2 Due: Friday July 15th at 11:59PM
- Milestone 3 Due: Friday July 22th at 11:59PM
- Milestone 4 Due: Friday July 29st at 11:59PM

Design Requirements

1. You must use **`const`** or **`#define directives`** to declare constants. Here is a sample list of constants:

```
MAX_INVENTORY_SIZE 4
TAX .13
```

2. You are allowed to code additional functions.
3. You must document (i.e. comments and indentation) your code properly.
4. You must put the following statement (as a comment) at the beginning of your source code:

```
I declare that the attached assignment is wholly my own work in accordance with
Seneca Academic Policy.  No part of this assignment has been copied manually or
electronically from any other source (including web sites) or distributed to
other students.

Name  _____________________   Student ID  _________________
```

5. You must use blank lines to separate logical units in the source code.
6. You are not allowed to declare global variables.

Milestone #1

Learning Outcomes

Upon successful completion of this milestone, you will be able to:

- Use basic mathematical operations to determine patterns
- Write functions to complete specific tasks
- Use loops and I/O functions to display structured output to the user

In this milestone you will be coding the functions required to output the board to the screen. You will be required to write two functions:

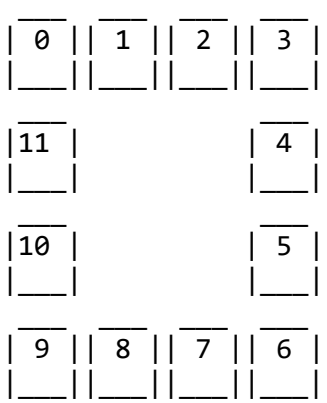
```
char getTileType(unsigned int index);
```

- This function returns the tile type for the tile indicated by the parameter "index"
- The tile to return is determined by the following rules:
    - 'C' – Checkout tile
        - The 0th tile
    - 'W' – Win prize tile
        - Every 3rd tile
    - 'L' – Lose prize tile
        - Every 5th tile
    - 'G' – Grand prize tile
        - Every 7th tile
- Any tile that does not fall within the above rules is an empty tile (space ' ')
- Any tile that would end up with multiple types will adopt the less common type
    - Example: The 15th tile would have both W and L, since L is will appear on the board less frequently it will be chosen in place of W

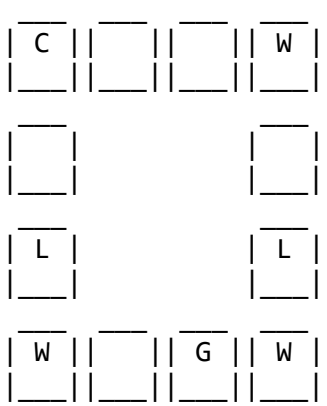
```
void displayBoard(unsigned int size);
```

- This function outputs the square game board
- The number of tiles along one side is indicated by the parameter "size". (all sides are the same number of tiles)
- Each tile will be composed of 3 lines:
    - Top: " ___ " (3 underscores surrounded by 1 space on each side)
    - Middle: "| ? |" (The tile symbol surrounded by 1 space on each side and a 1 pipe on each side)
    - Bottom: "|___|" (3 underscores surrounded by 1 pipe on each side)
- Keep in mind that the board must be printed one line at a time, so special consideration on output formatting is required!
- This function should use the getTileType function to determine what symbol to print on each tile
- A board of **any size** should be able to be outputted correctly
- Read below for more details on board configuration

The board tiles are counted in a clockwise order starting from the top left, shown below:



The board above has a side length of 4 tiles and a total of 12 tiles. This board should end up being outputted to the screen with all the correct tile symbols, shown below:



Once both functions have been successfully coded, test your code with the following main function:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main() {
	int s = 1;
	if (getTileType(153) != 'W') printf("153 should return W");
	else if (getTileType(65) != 'L') printf("65 should return L");
	else if (getTileType(49) != 'G') printf("49 should return G");
	else if (getTileType(0) != 'C') printf("0 should return C");
	else if (getTileType(105) != 'G') printf("105 should be G");
	else if (getTileType(79) != ' ') printf("79 should be space");
	else {
		printf("getTileType test passed!\nEnter board size: ");
		scanf("%d", &s);
		displayBoard(s);
	}
}
```

See the following page for program completion and submission:

Program completion

Your program is complete if your output matches the following output. Green numbers show the user's input.

```
--Program Start—

getTileType test passed!
Enter board size: 4
 ___  ___  ___  ___
| C ||   ||   || W |
|___||___||___||___|
 ___              ___
|   |            |   |
|___|            |___|
 ___              ___
| L |            | L |
|___|            |___|
 ___  ___  ___  ___
| W ||   || G || W |
|___||___||___||___|

--Program End--

--Program Start—

getTileType test passed!
Enter board size: 1
 ___
| C |
|___|

--Program End--
```

```
--Program Start—

getTileType test passed!
Enter board size: 6
 ___  ___  ___  ___  ___  ___
| C ||   ||   || W ||   || L |
|___||___||___||___||___||___|

 ___                      ___
|   |                    | W |
|___|                    |___|

 ___                      ___
| W |                    | G |
|___|                    |___|

 ___                      ___
|   |                    |   |
|___|                    |___|

 ___                      ___
|   |                    | W |
|___|                    |___|

 ___  ___  ___  ___  ___  ___
| L || G ||   || W ||   || L |
|___||___||___||___||___||___|

--Program End--
```