Summer 2016

Board Game Application

Final Project – Milestone 3

Problem Description

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

Full 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 #3

Learning Outcomes

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

- Use structs to construct new types to store complex data sets
- Use pointers to structs to allow data manipulation from multiple scopes

In this milestone you will be coding a player struct to contain the player's information and altering your current functions to work with it. This version will allow up to a total of 8 players to play.

Your program will require the following struct:

`struct Player`

This struct will contain all the necessary data for a single player. A player is composed of a single character to act as the player's name, whole numbers for the player's current amount of money (score), number of inventory items, position on the board and lastly an array of 10 whole numbers to store their prizes.

int main()
The main function will need to be altered to meet the requirements of this milestone. Instead of storing a player's individual variables, the main will now allocate memory to store an array of 8 player instances. When the user chooses 'p' to play the game, this function should prompt the user for the number of players then initialize each of those players one at a time using the initPlayer function. Then, the main() function prompts the user for the size of the board then calls the playGame function. Many function calls will require minor alteration to adhere to the function signature changes listed below.

In addition to main() your milestone must alter functions with the following prototypes:

`void initPlayer(struct Player* player)`

This function will operate in the same as the previous milestone, but will now use a player struct pointer instead of individual variables pointers.

`void playGame(unsigned int boardSize, struct Player players[], unsigned int playerCount);`

Alter this function so that it follows all the same rules as the previous milestone but now works with many players. The players will take turns in the order they are stored in the 'players' array. The number of players who will be playing is indicated by the 'playerCount' variable. Once the final player finishes their turn, the first player will start their second turn, going through the list of players again until someone wins.

Now that multiple players are able to play at once, you must code in one additional rule to the game. When a player lands on a tile that is already occupied by another player, the moving player will steal and

item from the player on that tile and then moves back one tile. If the player being stolen from doesn't have any items, nothing happens and the player simply moves back one tile. If the player being stolen from has an item but the stealing player does not have room for the item, the item is lost. When the stealing player moves back one tile, the player will respond to whatever tile they land on. For example, if a player were to steal an item and move back one tile causing them to land on a 'W' tile, they would additionally win a prize that turn. If the player would land on the tile they originally came from, their turn ends without anything else happening. (HINT: group your tile resolution code together and simply run it every time a player moves backwards or forwards!)

`char getTileType(unsigned int index, struct Player players[], unsigned int playerCount)`

Alter this function to match the above signature. This function works as it did in the previous milestone but will first check all the player's positions to see if any match. If a player's position matches, it returns the name of the player as a character, otherwise, it returns the tiles based on the board rules.

`void displayBoard(unsigned int size, struct Player players[], unsigned int playerCount);`

Alter this function to match the above signature. This function works the same as it did in the previous milestone but will now additionally pass the 'players' array and 'playerCount' variable to the getTileType function (you no longer need the getDisplayTile function).

`void winPrize(struct Player* player);`

Alter this function to match the above signature. This function works the same as it did in the previous milestone, but now uses the player struct pointer instead of individual variables.

`void winGrandPrize(struct Player* player);`

Alter this function to match the above signature. This function works the same as it did in the previous milestone, but now uses the player struct pointer instead of individual variables.

`int loseItem(struct Player* player);`

Alter this function to match the above signature. This function works the same as it did in the previous milestone, but now uses the player struct pointer instead of individual variables.

`int checkout(struct Player* player);`

Alter this function to match the above signature. This function works the same as it did in the previous milestone, but now uses the player struct pointer instead of individual variables.

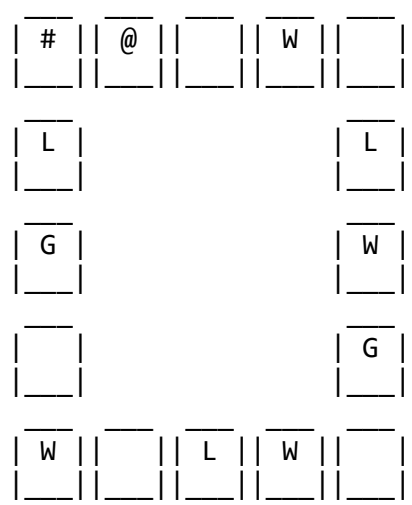
See the following page for program completion and submission:

Program completion

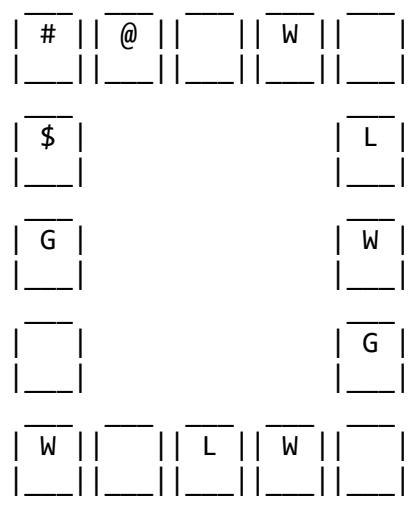
Your program is complete if your output matches the following output when the srand function is not used or commented out. Green numbers show the user's input. Your program does not have to match this output exactly to be considered complete.
HINT: If your dice rolls do not match up with this output, double check your random number functions!
HINT: Check the end of this document for debugging inputs!

```
Welcome to CHECKOUT
Main Menu
p-(p)lay q-(q)uit r-inst(r)uctions s-HI(s)core:
p
Enter number of players: 4
Enter player ID: @
Enter player ID: #
Enter player ID: $
Enter player ID: %
Enter board size: 5
 ___  ___  ___  ___  ___
| @ ||   ||   || W ||   |
|___||___||___||___||___|

 ___                 ___
| L |               | L |
|___|               |___|

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

 ___                 ___
|   |               | G |
|___|               |___|

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

@'s turn, how many dice will you roll: 3
You rolled: 6 6 5
You do nothing.


 ___  ___  ___  ___  ___
| # || @ ||   || W ||   |
|___||___||___||___||___|

 ___                 ___
| L |               | L |
|___|               |___|

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

 ___                 ___
```

#'s turn, how many dice will you roll: 3
You rolled: 5 6 5
You do nothing.

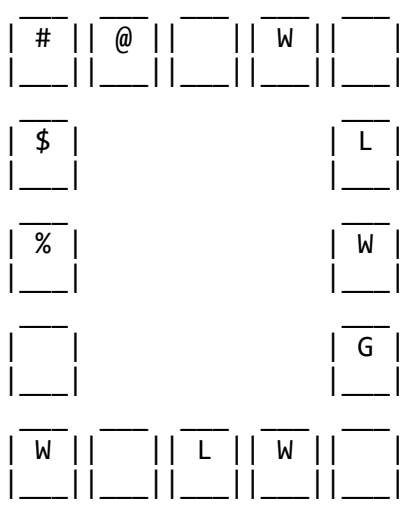


$'s turn, how many dice will you roll: 1
You rolled: 1
You were unable to steal from @!
You were unable to steal from #!
You have nothing to lose!

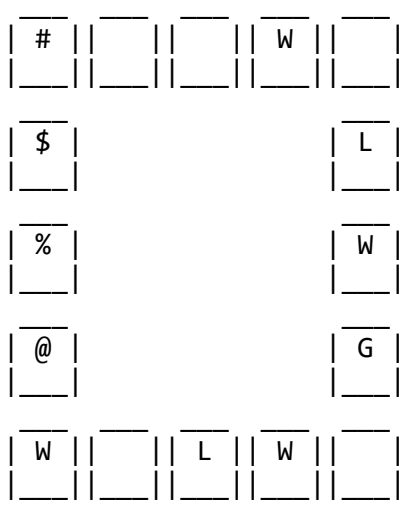


%'s turn, how many dice will you roll: 1
You rolled: 1

```
You were unable to steal from @!
You were unable to steal from #!
You were unable to steal from $!
You won a grand prize valued at $196
```




```
@'s turn, how many dice will you roll: 3
You rolled: 3 6 6
You were unable to steal from #!
You were unable to steal from $!
You stole a prize from %!
You do nothing.
```




```
#'s turn, how many dice will you roll: 1
You rolled: 4
You do nothing.
```

```
|___||___||___||___||___|

 ___                 ___
| $ |               | L |
|___|               |___|

 ___                 ___
| % |               | W |
|___|               |___|

 ___                 ___
| @ |               | G |
|___|               |___|

 ___  ___  ___  ___  ___
| W ||   || L || W ||   |
|___||___||___||___||___|
```

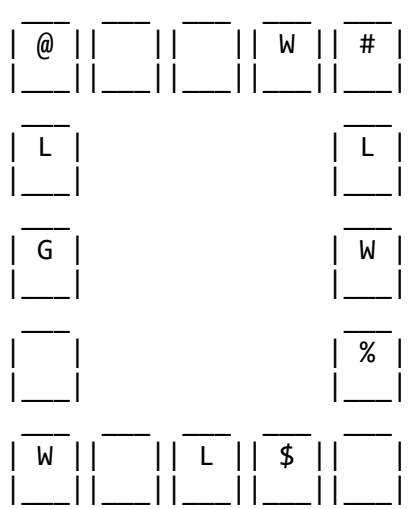
$'s turn, how many dice will you roll: 3
You rolled: 2 6 2
You won a prize valued at $31

```
 ___  ___  ___  ___  ___
| C ||   ||   || W || # |
|___||___||___||___||___|

 ___                 ___
| L |               | L |
|___|               |___|

 ___                 ___
| % |               | W |
|___|               |___|

 ___                 ___
| @ |               | G |
|___|               |___|

 ___  ___  ___  ___  ___
| W ||   || L || $ ||   |
|___||___||___||___||___|
```

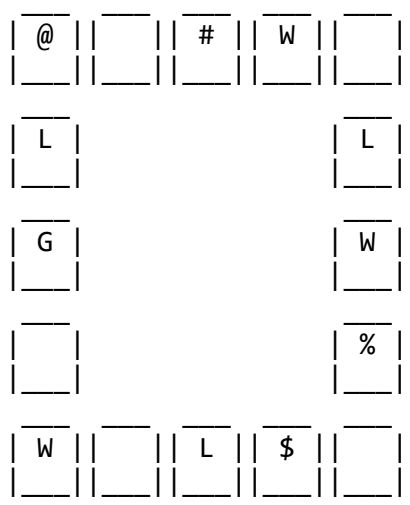
%'s turn, how many dice will you roll: 3
You rolled: 4 1 4
You won a grand prize valued at $160

```
 ___  ___  ___  ___  ___
| C ||   ||   || W || # |
|___||___||___||___||___|

 ___                 ___
| L |               | L |
|___|               |___|

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

 ___                 ___
| @ |               | % |
|___|               |___|
```

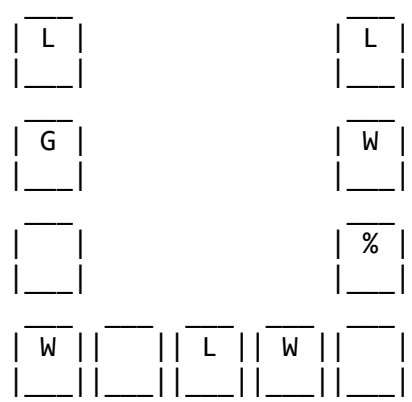
```
@'s turn, how many dice will you roll: 1
You rolled: 3
You checkedout for $196.00
```



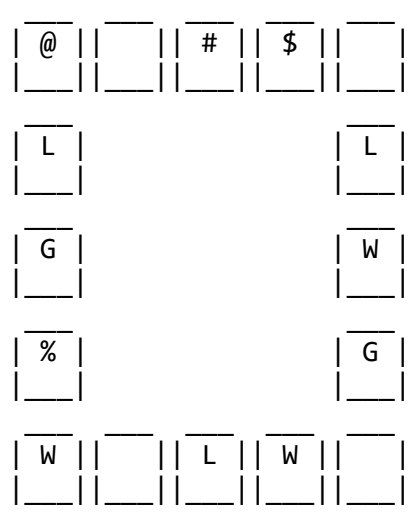
```
#'s turn, how many dice will you roll: 3
You rolled: 4 5 5
You do nothing.
```



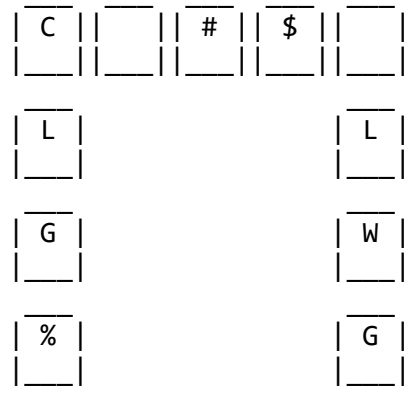
```
$'s turn, how many dice will you roll: 3
You rolled: 4 3 3
You won a prize valued at $67
```



%'s turn, how many dice will you roll: 1
You rolled: 6
You do nothing.



@'s turn, how many dice will you roll: 3
You rolled: 1 6 1
You do nothing.



```
| W ||   || L || W || @ |
|___||___||___||___||___|

#'s turn, how many dice will you roll: 3
You rolled: 4 5 6
You do nothing.


 ___  ___  ___  ___  ___
| C || # ||   || $ ||   |
|___||___||___||___||___|

 ___                 ___
| L |               | L |
|___|               |___|

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

 ___                 ___
| % |               | G |
|___|               |___|

 ___  ___  ___  ___  ___
| W ||   || L || W || @ |
|___||___||___||___||___|

$'s turn, how many dice will you roll: 1
You rolled: 2
You lost a prize valued at $67


 ___  ___  ___  ___  ___
| C || # ||   || W ||   |
|___||___||___||___||___|

 ___                 ___
| L |               | $ |
|___|               |___|

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

 ___                 ___
| % |               | G |
|___|               |___|

 ___  ___  ___  ___  ___
| W ||   || L || W || @ |
|___||___||___||___||___|

%'s turn, how many dice will you roll: 3
You rolled: 1 6 4
You were unable to steal from @!
You won a grand prize valued at $112


 ___  ___  ___  ___  ___
| C || # ||   || W ||   |
```

```
|___||___||___||___||___|
 ___                 ___
| L |               | $ |
|___|               |___|
 ___                 ___
| G |               | W |
|___|               |___|
 ___                 ___
|   |               | % |
|___|               |___|
 ___  ___  ___  ___  ___
| W ||   || L || W || @ |
|___||___||___||___||___|
```

@'s turn, how many dice will you roll: 3
You rolled: 4 4 3
You won a prize valued at $45

```
 ___  ___  ___  ___  ___
| C || # ||   || @ ||   |
|___||___||___||___||___|
 ___                 ___
| L |               | $ |
|___|               |___|
 ___                 ___
| G |               | W |
|___|               |___|
 ___                 ___
|   |               | % |
|___|               |___|
 ___  ___  ___  ___  ___
| W ||   || L || W ||   |
|___||___||___||___||___|
```

#'s turn, how many dice will you roll: 1
You rolled: 2
You stole a prize from @!
You do nothing.

```
 ___  ___  ___  ___  ___
| C ||   || # || @ ||   |
|___||___||___||___||___|
 ___                 ___
| L |               | $ |
|___|               |___|
 ___                 ___
| G |               | W |
|___|               |___|
 ___                 ___
|   |               | % |
```

```
|___|               |___|

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

$'s turn, how many dice will you roll: 3
You rolled: 5 6 3
You were unable to steal from @!
You stole a prize from #!
You do nothing.


 ___  ___  ___  ___  ___
| C || $ || # || @ ||   |
|___||___||___||___||___|

 ___                 ___
| L |               | L |
|___|               |___|

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

 ___                 ___
|   |               | % |
|___|               |___|

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

%'s turn, how many dice will you roll: 3
You rolled: 4 4 2
You stole a prize from $!
You checkedout for $317.00

You won the game!
Main Menu
HI SCORE: 460 Player Name: @
Main Menu
p-(p)lay q-(q)uit r-inst(r)uctions s-HI(s)core:
q
This game is much more fun than bash...
```

To make debugging easier, simply copy and paste the following inputs in your program to test it.

```
4
@
#
$
%
5
3

3

1

1

3

1

3

3

1

3

3

1

3

3

1

3

3

1

3

3

1
```

3

3

1