**Testing and Reflections**

Based on the input we see the program run the following ways:

**Queue**

|  |
| --- |
| fred6 won against john7  hgo6 won against mary56  oemje10 won against oweng9  fred6 won against yu90  hgo6 won against oemje10  fred6 won against hgo6  The Tournament Champion is: fred6  Total Games Played: 6  The average number of Games Played: 0.86  fred6 played 3 games.  john7 played 1 games.  mary56 played 1 games.  hgo6 played 3 games.  oemje10 played 2 games.  oweng9 played 1 games.  yu90 played 1 games |

**Stack**

|  |
| --- |
| oweng9 won against yu90  oemje10 won against oweng9  hgo6 won against oemje10  hgo6 won against mary56  hgo6 won against john7  fred6 won against hgo6  The Tournament Champion is: fred6  Total Games Played: 6  The average number of Games Played: 0.86  fred6 played 1 games.  john7 played 1 games.  mary56 played 1 games.  hgo6 played 4 games.  oemje10 played 2 games.  oweng9 played 2 games.  yu90 played 1 games |

==================

What differences did you see?

The queue has a better distribution that is more players were involved in the tournament whereas the stack involved the same individual over and over until they lost.

How do these differences help you understand a Queue or a Stack better?

I suppose I didn’t see a striking difference between running the program as a Queue and as a Stack, but the program helped me gain a good understanding of the differences when I had to design them and implement them.

Is there a significant difference with different numbers of players?

   I think one of the issues with it is that we didn’t create the program to randomly generate a winner, so when I insert more players if one of the player’s name has an ASCII value above everyone else they will play all the games depending on if they’re at the top or end.

What about when there are only a few players?

   There’s a better distribution of games between the Queue and the Stack and they are closer to equal.

What about when there are a large number of players?

   One individual wins a majority of the matches and the Queue remains the better distribution of games compared to the Stack.

Persuasive Argument:

  Based on the instruction it seemed like I was supposed to conclude that the Queue was not as functional as the Stack for the assignment. But from what I can tell, the Queue is going to give an opportunity for more players to participate in the tournament.

It also should create better matches as the Queue/Stack starts with X people with 0 wins total. Then systematically the Queue saves all the players that win a match and eliminates all the people with 0 wins, before starting on “round 2” having all the 1 win players play until there are only 2 win players left and so on. Both of those seem beneficial for a tournament.

The Queue seems like the best choice in this case – though most tournaments are typically represented with a bracket system which might utilize two trees (or some combination of trees).

A stack might be a good system if the players are already ranked based on ELO (or some other rating) where you want the top players (at the bottom of the stack) to have to play less games.

**Errors (fixed)**

Error: Computer does not generate a correct queue when the last item in the queue is a hit

An error in computer.cpp's logic causes the error as Project 2 C and Project 2 B fail equally, with the task – we also see that STL Queue and the self created Queue are equally functional however, when creating the Queue and the removal of items from the Queue.

|  |
| --- |
| Based on this map:  0 1 2 3 4 5 6 7 8 9  A o o o o o o o o o o  B o o o o o o o o o o  C o o o o o o o o o o  D S S S S S o o o o o  E S S S S S o o o o o  F o S S S S o o o o o  G o o o S S o o o o o  H o o o o S o o o o o  I o o o o o o o o o o  J o o o o o o o o o o    ~~~~~~~~~~~~~~~~~~  //Random Guess Results in Hit  Random Guess: D 2  Computer Guessed: D 2  Computer player: Hit    //Queue is created with (Up, Right, Down, Left):  //C 2  //D 3  //E 2  //D 1  Queue Size: 4  Computer Guessed: C 2  Computer player: Miss    //Remove C2 and guess next  Queue Size: 3  Computer Guessed: D 3  Computer player: Hit    //Remove D3 and guess next  Queue Size: 2  Computer Guessed: E 2  Computer player: Hit    //Remove E2 and guess next  Queue Size: 1  Computer Guessed: D 1  Computer player: Hit  //D1 results in a hit on last item in queue  //Create new Queue (Up, Right, Down, Left):  Queue Size: 4  Computer Guessed: E 1  Computer player: Hit    Queue Size: 1  Computer Guessed: D 0  Computer player: Hit |

Error: Human player wins after seventeen turns (number of total ship sections)

A proper check for the human player and computer player is performed to make sure that when checkPosition results in a hit you subtract one from shipsectionsleft for the appropriate player.

The error exists when you keep attacking the same "H" section of map – so essentially if you keep hitting the same area of the map because the checkPosition was returning H for a section of map hit multiple times. Altered checkPosition function to return "Already" if area was already attacked.