Jared Parkinson

CS162 Assignment 1

Assignment 1 Reflections

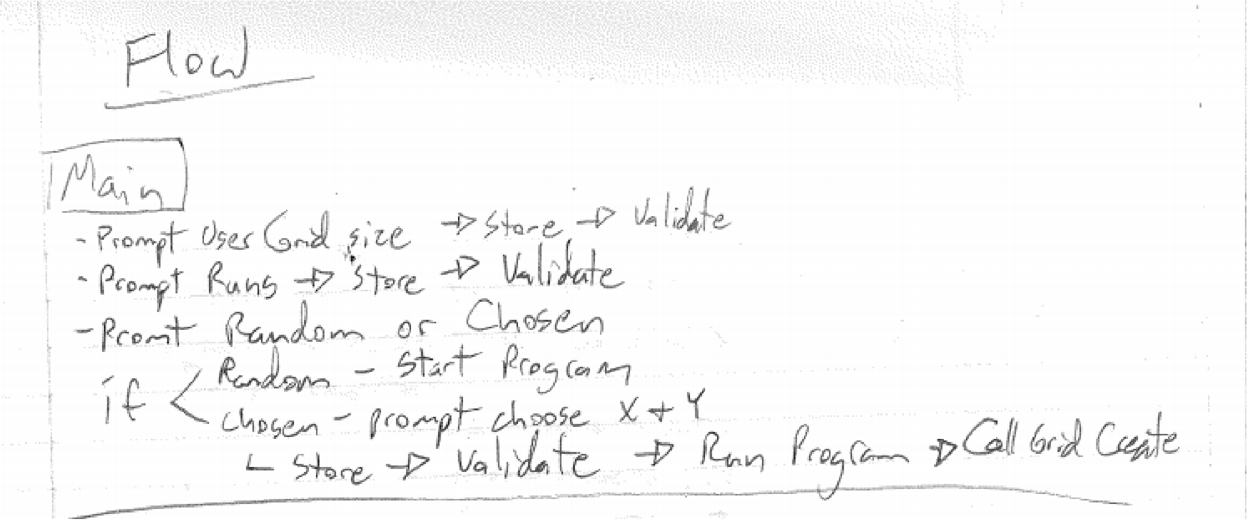
Requirements:

This assignment code is supposed to:

1. Prompt user for input. (Grid X and Y size, Desired runs, start X and Y if applicable.)
2. Take that input and pass it to Grid Creation and moves function.
3. Moves function will then move Ant 1 space per run based on 2 rules listed below.
4. Ant will continue to move until moves has been met. If moves are high enough, Ant will create a continual pattern.

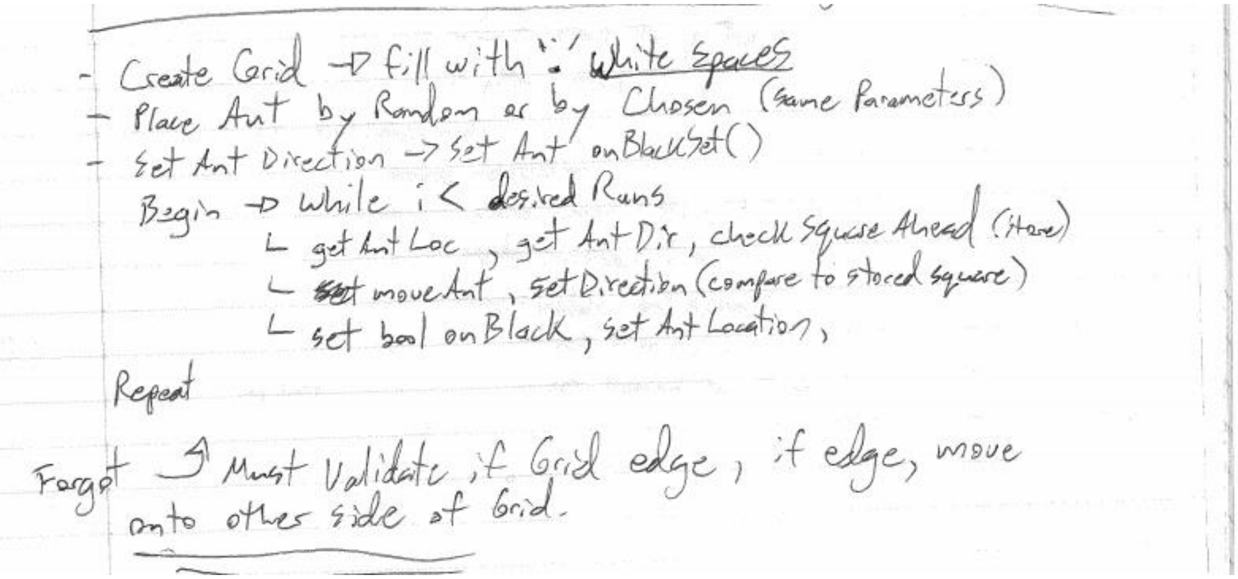
The way I understood this assignment explanation is as follows:

There is 1 Ant character in a 2D array of characters. This Ant will move according to two simple rules. If this Ant is on a set square(Black), it turns right, then moves one space forward. If it is on an unset square(White), it turns Left and moves forward one space. User Input: User will input 1 or greater desired runs. User will input X and Y cords if chosen start location is specified, or nothing if Random is specified.



Flow

```
Main
- Prompt Users Grid size → Store → Validate
- Prompt Runs → Store → Validate
- Promt Random or Chosen
if < Random - Start Program
     Chosen - prompt choose X + Y
        └ Store → Validate → Run Program → Call Grid Create
```



- Create Grid → fill with ' ' White Spaces
- Place Ant by Random or by Chosen (same Parameters)
- Set Ant Direction → set Ant onBlackSet()

Begin → while i < desired Runs
  └ get Ant Loc, get Ant Dir, check Square Ahead (store)
  └ ~~set~~ moveAnt, setDirection (compare to stored square)
  └ set bool onBlack, set Ant Location,

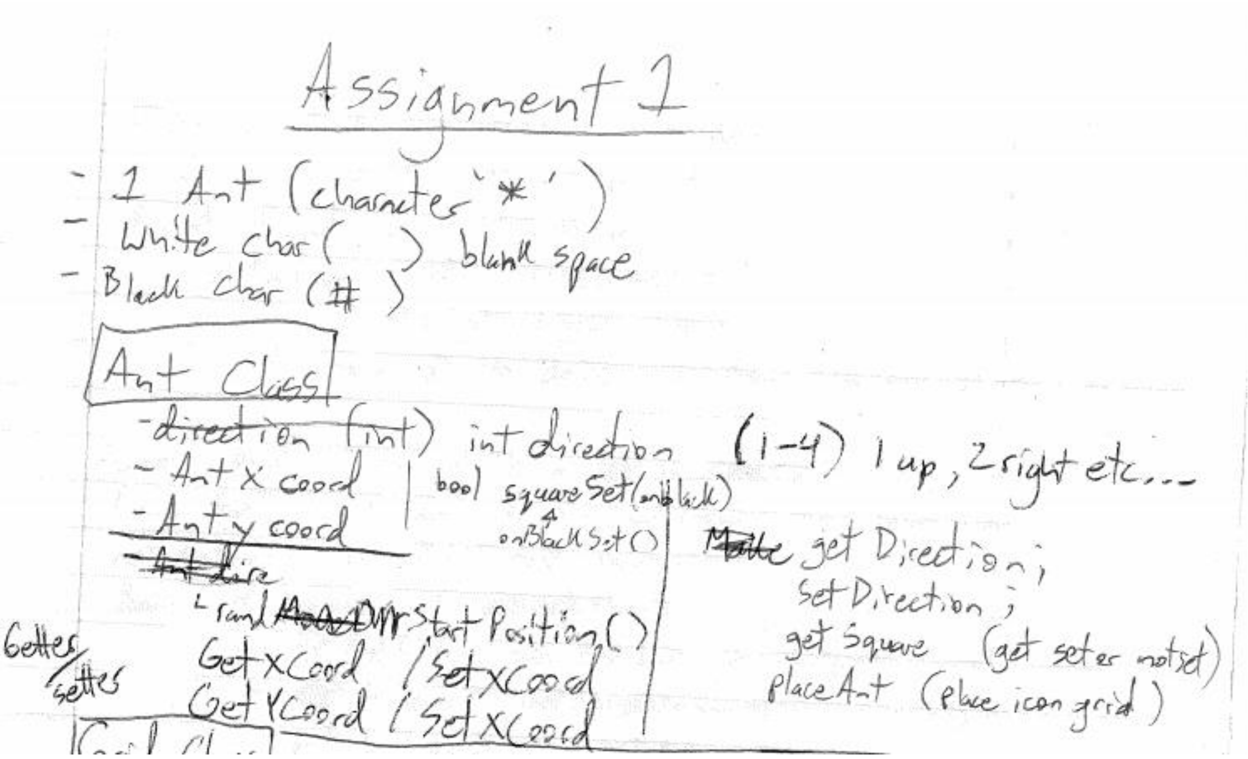
Repeat

Forgot ↑ Must Validate if Grid edge, if edge, move onto other side of Grid.

Class Design:

In the future documents, I will use a table. For assignment 1, I will list my hand written work.

There were quite a few design changes that needed to be made from this initial design. I created an Ant class and a Grid class. After reading over the forum I realized I needed a Menu Function. This Menu function I created as a Menu.cpp and hpp and created its own class. The Menu did not have all the functions I ended up using for validation as the ones I had written were not enough to work with the user input.



Assignment 1

- 1 Ant (character '*')
- White char ( ) blank space
- Black char (#)

Ant Class

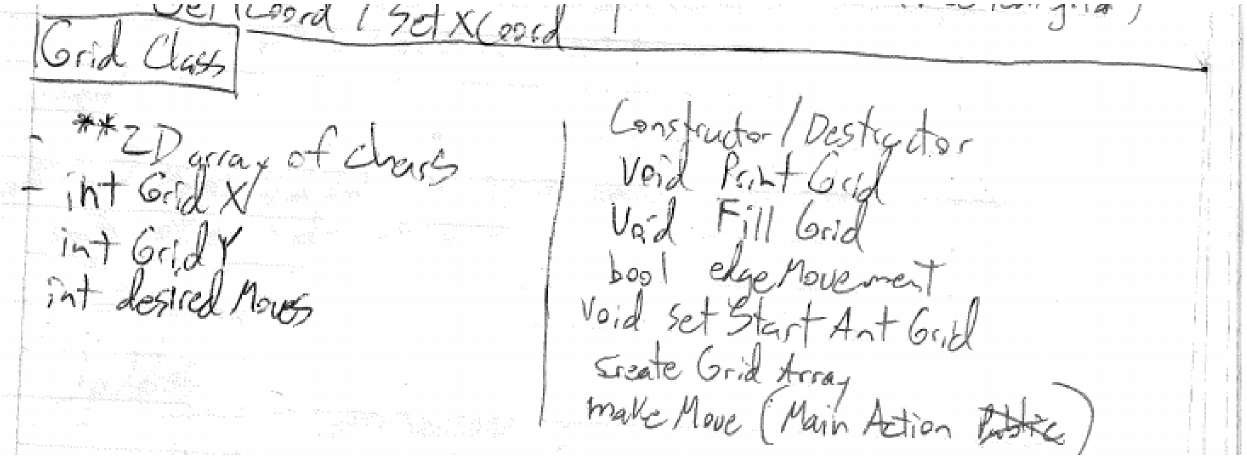
- ~~direction~~ (int) int direction (1-4) 1 up, 2 right etc...
- Ant X coord | bool squareSet(onBlack)
- Ant y coord onBlackSet()

~~Ant Dire~~
└ rand ~~AntDir~~ Start Position()

Getters/setters
GetXCoord / SetXCoord
GetYCoord / SetXCoord

~~Make~~ get Direction;
SetDirection;
get Square (get set or notset)
PlaceAnt (place icon grid)

Grid Cl

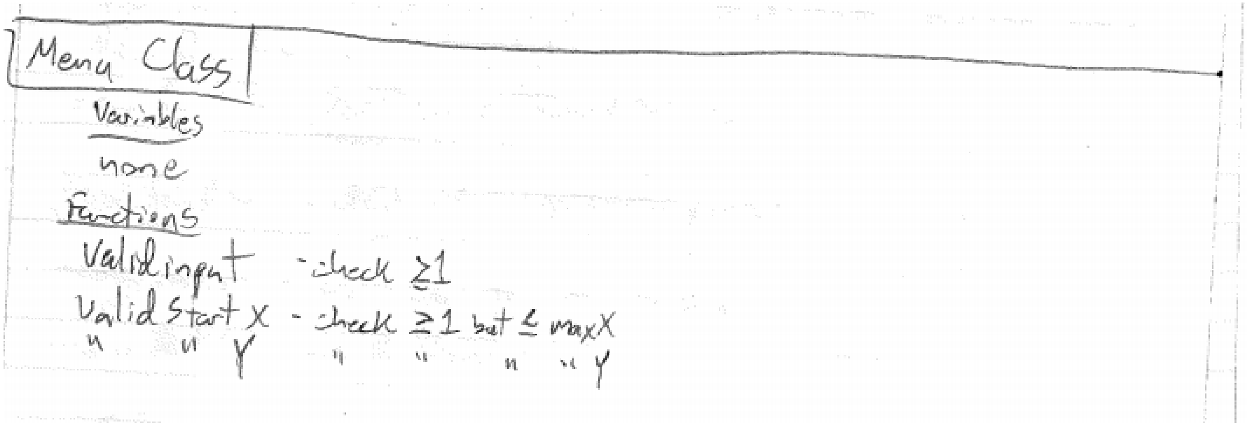


Get YCoord / Set XCoord

Grid Class

- **2D array of chars
- int Grid X
int Grid Y
int desired Moves

Constructor / Destructor
Void Print Grid
Void Fill Grid
bool edgeMovement
Void set Start Ant Grid
Create Grid Array
makeMove (Main Action ~~Public~~)



Menu Class

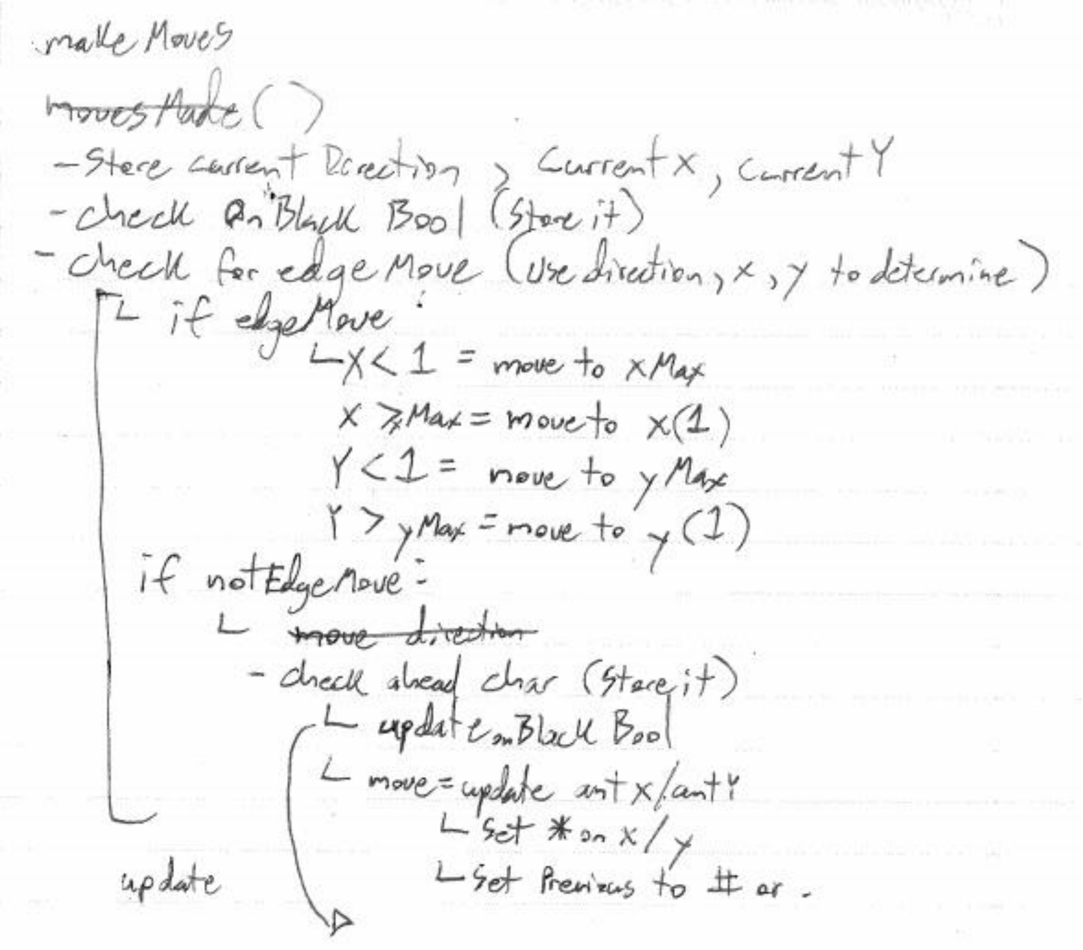
Variables

none

Functions

ValidInput - check ≥1
Valid Start X - check ≥ 1 but ≤ maxX
" " Y " " " " Y

The below makeMoves function was vetted to design how the initial move function would work. edgeMove was then removed because it was not needed.



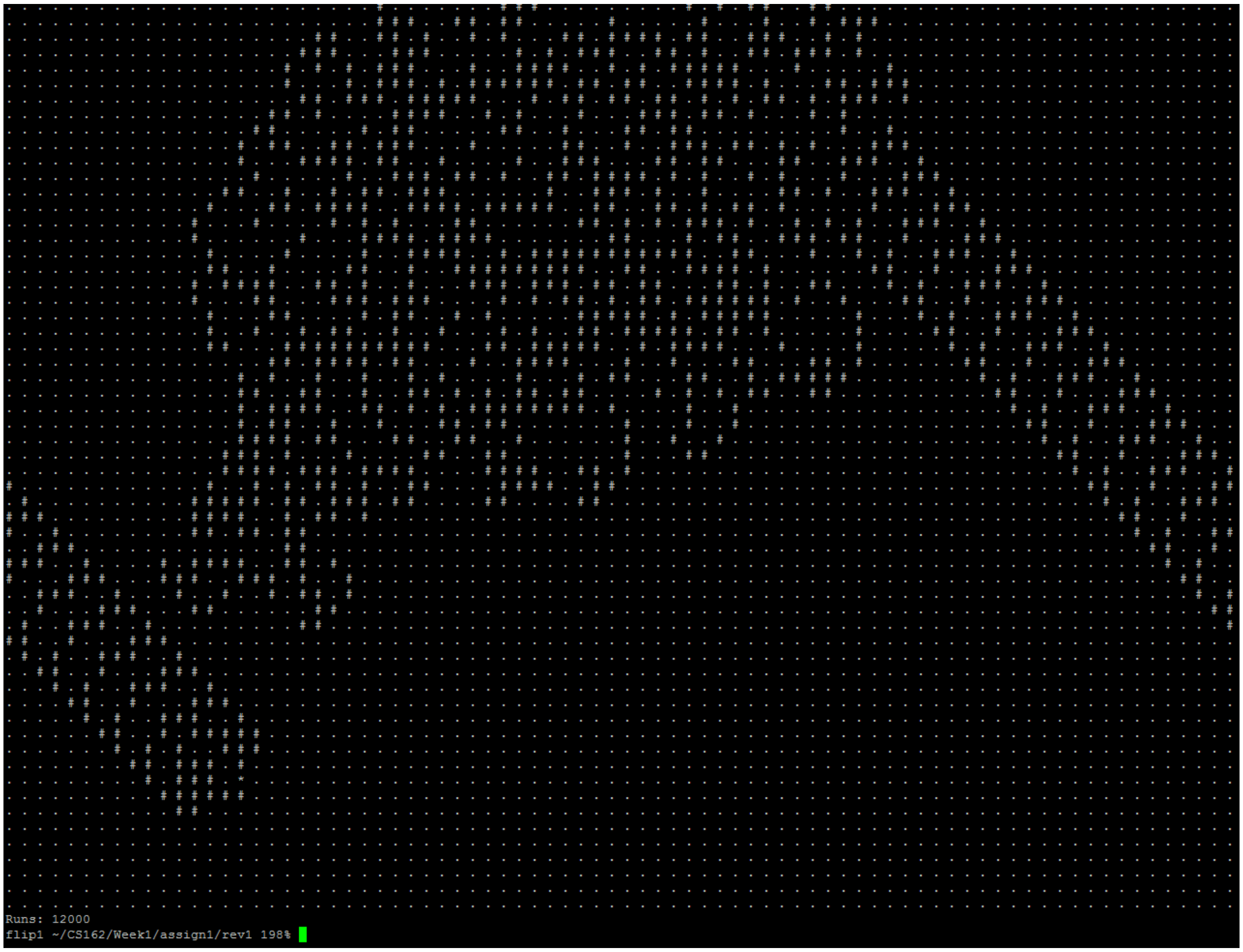
```
make Moves

~~movesMade~~( )
 - Store current Direction, Current X, Current Y
 - check OnBlack Bool (Store it)
 - check for edgeMove (use direction, x, y to determine)
   └ if edgeMove :
              └ X < 1 = move to xMax
                x > xMax = move to x(1)
                Y < 1 = move to yMax
                Y > yMax = move to y(1)
     if notEdgeMove :
         └ ~~move direction~~
           - check ahead char (Store it)
              └ update onBlack Bool
              └ move = update antX/antY
                   └ Set * on x/y
   update          └ Set Previous to # or .
```

Test Plan: A table or outline of test cases that shows what is being tested and what the expected outcome is, then the result of the test.

Test Case	Expected Outcome	Actual Outcome
User Input < 1: Grid, desiredRuns, startX etc User cannot put too high	Fail w/retry	Initial: Fail due to incorrect bool usage in Menu function 2nd: Pass
Enter offset grid size e.g. 10x50	No fails on any access point of the Grid	Initial: Failed at first per referencing cols instead of rows 2nd: Failed again due to accessing invalid memory spot After Fixing both: Passed
Enter 1x1 grid size	No fails, run for X entered times even though only 1x1	Initial: Fail. Failed bc it did not show Ant on the 1x1

Enter 1x1 starting location on a large Grid	No fails, go over edge of Grid	Initial: Fail. It kept trying to access invalid memory in the updatePrevSquare function. 2nd: Pass. I updated the function to have boundaries
Make sure edgeMovement function worked by stepping through actual values	No fails, find edge and report as an edge movement	Initial: Failed.
Check move count	Show correct move count as per user input	Initial: Failed. I was 1 move off. This is because I was counting the iteration and not the iteration +1 2nd: passed: Add ++ int to the count section and pass at 500 moves on a 500 run.

Below is an example of what the program did at 12000 runs on Flip. 80x80 Grid. User chosen start at 40x40.



```
Runs: 12000
flip1 ~/CS162/Week1/assign1/rev1 198%
```

Reflections:

For this assignment, the first thing I did was sit down with a pen and paper and try to write out what I would need to do. I layed out 3 classes in the beginning. Menu, Ant and Grid. I then began to write what I thought might be good Attributes and Functions. After I wrote out the initial classes and variables, I went ahead and wrote a small outline of how the program would Flow. This design was rudimentary and eventually changed quite a bit. The reason it changed, was because there needed to be quite a bit more thought into how the program would do the steps and what functions it would use to make those steps.

I added a section below that called Updates. This updates section had more functions written out that I had missed/needed and a direction wheel to look at. As I fleshed out the design more and more, I went ahead and started to code the initial program. When writing the program, it was readily apparent that I needed more functions to deal with things or I would have a monster 2 page function impossible to troubleshoot on my hands. Below are some issues I ran into.

Some Issues:

1. Random Start location – Random start location was constantly picking outside locations that didn't exist in memory. Reviewing the function with stops for testing revealed the

offending functions and I rectified by updating the functions to properly find the grid over edges.

2. Moving Ant – I kept seeing the Ant move 1 space to the right infinitely. This was because I had accidently used an X in a Ycoord location and it would consistently choose Black.
3. Incorrect makefile – The makefile I was using ended up failing because it was missing the new Menu items I had created. Adding them resolved the issue.
4. Not enough planning – If I had to do this all over again I would plan more. The planning was rudimentary and lacking in its foresight. I wish I had thought of the issues more clearly ahead of time.
5. Constant changing Ant location without color – At first my Ant would change location infinitely but it would not change color behind it. This was because the onBlack bool was always set to true! Fixing this resolved the issue.
6. edgeMovement - Completely unneeded and wasted programming. I created a function that took up 40 lines and took a while to program. This function ended up not being needed and I completely took care of it in 4 lines of code in another function. This was a slap in the face to me and made me angry at how much time I had wasted on it!
7. cin user input character fail – I abandoned using the below cin.fail because I could not get it to work for additional user validation as intended. This was not a requirement per the assignment, but was a suggestion on the forum. Some validation (which I do with number size) was required. I fully plan on using the below when I figure out how to properly implement it into a function validation rather than using it straight on Main.

```
while (std::cin.fail()) {
                std::cout << "You must choose a digit." << std::endl;
                std::cin.clear();
                std::cin >> desiredRuns;
        }
```