**Project 2**

Title

<Fighter Fighter>

Class:

CIS-5

Due Date:

May 4, 2016

By:

Bassam Elsaleh

Table of contents

Summary………………………………………………………………………………….Pg.3

Samples…….……………………………………………………………………………..Pg.5

Pseudo Code…………………..………………………………………………………….Pg.7

Flow Chart…….………………………………………………………………………….Pg.10

Constructs Used.………………………………………………………………………….Pg.16

Sources……………………………………………………………………………………Pg.17

Summary

Fighting games are always fun to play both alone and with friends, from the more famous ones like mortal combat to the lesser known games like skull girls. These games deal with a lot of skill and timing, but there were and still exist games that are not as complex and challenging. This game requires little to no skill, it’s a simple rock paper scissors match. Where the user can have fun with friends or play agents the computer.

This program is not complex yet required over 900 lines due to the fact that I added a special attacks that has restrictions, which will be covered in this section. Players will start in the menu and chose to either play one player, two players, look at the rules or quit which will end the game.

Now,  as for rules, the player can use one of 4 moves, the first being a simple punch that will deal 2 damage to the opponent, now that doesn’t mean there is no way to prevent the attack, the players are allowed to block the damage by using the next ability Blocking. Blocking stops all the damage coming from the punch, now some players are strange and like to joke around by doing noting and just keep blocking so I created the special attack. The special attack will deal damage to a character that is blocking and 2 damage to a blocking character. Though, because this attack is so strong I had to restrict the special attack so that it could only be used 4 times, by doing so I had to re-write the code (copy and paste) for every instance that one or both players might run out of special attacks. I then added one more move just for fun that would allow the character to be able to counter only the special attack, dealing all 4 damage back to the player who is using the special attack, again this will only block the special attack not the standard punch.

This code took me about a 2 day to write up both the single player and multiplayer, but it wasn’t perfected with all the damg and outputs that were needed. Within a week I had the single player up and running and it was a simple copy and paste with a few modifications to make it multiplayer. The hardest part of the code was that because of the intensity of the code and going over 900 lines of code it made it difficult to relocate specific areas, if I were to re-write it, I might add in more voids that would execute when the players us up their special moves. Doing this might add more lines, but it will also make it cleaner and easier to find specific areas of the code.

Also if I wanted to shorten the code it might be better to lose one of the moves, because as of right now there are 4 moves a player can pick and the second player will then pick theirs. For each move the single player picks there are 4 options that need to be foreseen that makes 16 possible outcomes not including whether or not the player is out of special attacks.

Version number 2

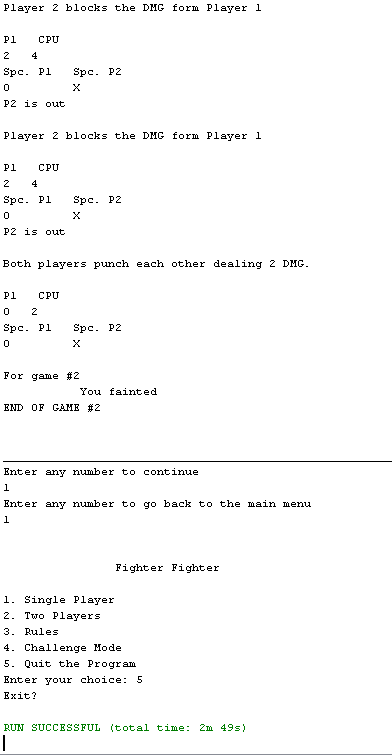
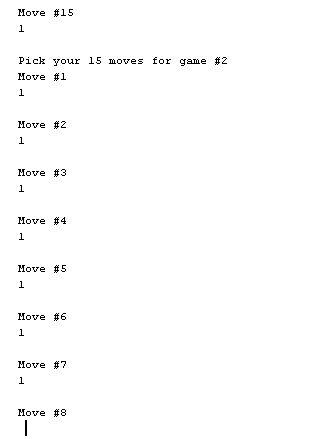
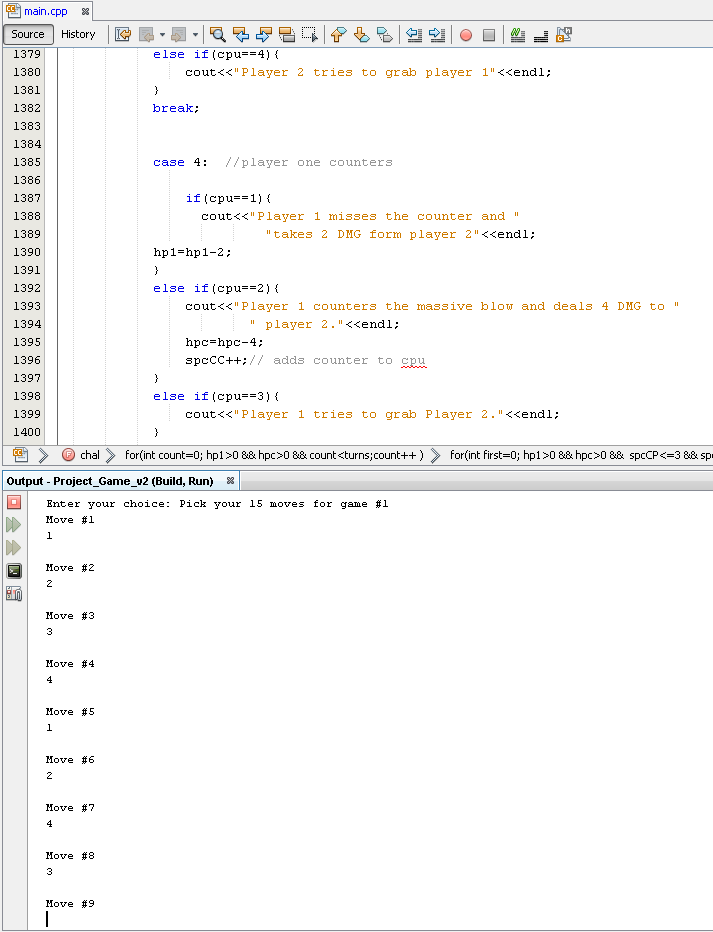
When wringing the second version of the project, I decided to take a part of the code that simulated the fighting and inputted it into a new programing file. I did this in order to play around with the code so that I was able to implement the new concepts and not mess with the old code.

I realized before I did anything that to use the new concept of arrays i was going to have to make a new game play, thus I created the “Challenge Mode”. In essence, Challenge Mode allows the player to pick any number up to 10 games and pick 15 moves for the possible fights.  This was perfect for me because I was able to demonstrate my ability to use not only arrays, but double arrays.

The next concept I added was the search. What I basically did was after the player finishes game #x, the program should search to see if they won that game. If so a message should be revealed that they won and saved.

A problem I noticed with changing the program to only having 15 moves was that, one player might still be alive after the 15th turn, so I had to add extra lines of code that would still represent a win or loss based on the who had the most health, and I decided on my own to remove the surrender button just in case someone were to press it by accident and the fact that they picked the number of games which run really fast, so there is no need to surrender.

After finishing the project, I feel like I should of used more functions in order to lower the amount of lines for the different attaches, for instance all 3 game modes use the same attacks and outcomes where if the specific moves are called then it goes to that specific function instead of staying in the same function, I might lose speed, but it would really clean up the code.



Pseudo Code

//function for the menu

//function for the rules

//function for one player

//function for two players

//declare variables

//do while choice=4

//call menu function

//players pick

//go to the function player picked

//exit stage right

//menu function

//show menu

//rules function

//declare variables

//show the rules until player exits

//one player function

//declare variables

//computer random

//player picks move

//show health

//1 = punch

//2 = spc.

//3 = block

//4 = counter

//keep looping till one player has no health or quits

//player and computer both pick their movies as described in the rules

//describe to the player what happens and heath is removed where needed.

// show heath and specials

//same fighting code but both players are out of specials

//same fighting code only player one is out of specials

//same fighting code only cpu is out of specials

//if both players die

//if player one dies

//if cpu dies

//if player one gives up

//go to file and show that player one wins.

//two player function

//declare variables

//player picks move

//player 2 picks

//show health

//1 = punch

//2 = spc.

//3 = block

//4 = counter

//keep looping till one player has no health or quits

//player and computer both pick their movies as described in the rules

//describe to the player what happens and heath is removed where needed.

// show heath and specials

//same fighting code but both players are out of specials

//same fighting code only player one is out of specials

//same fighting code only cpu is out of specials

//if both players die

//if player one dies

//if player 2 dies

//if both players give up

//if player one gives up

//if player two gives up

//Version number 2

//Challenge mode function

//declare variables

//player picks move

//player 2 picks

//show health

//1 = punch

//2 = spc.

//3 = block

//4 = counter

//keep looping till one player has no health or quits

//player and computer both pick their movies as described in the rules

//describe to the player what happens and heath is removed where needed.

// show heath and specials

//same fighting code but both players are out of specials

//same fighting code only player one is out of specials

//same fighting code only cpu is out of specials

//if both players die

//if player one dies

//if player 2 dies

//if both players give up

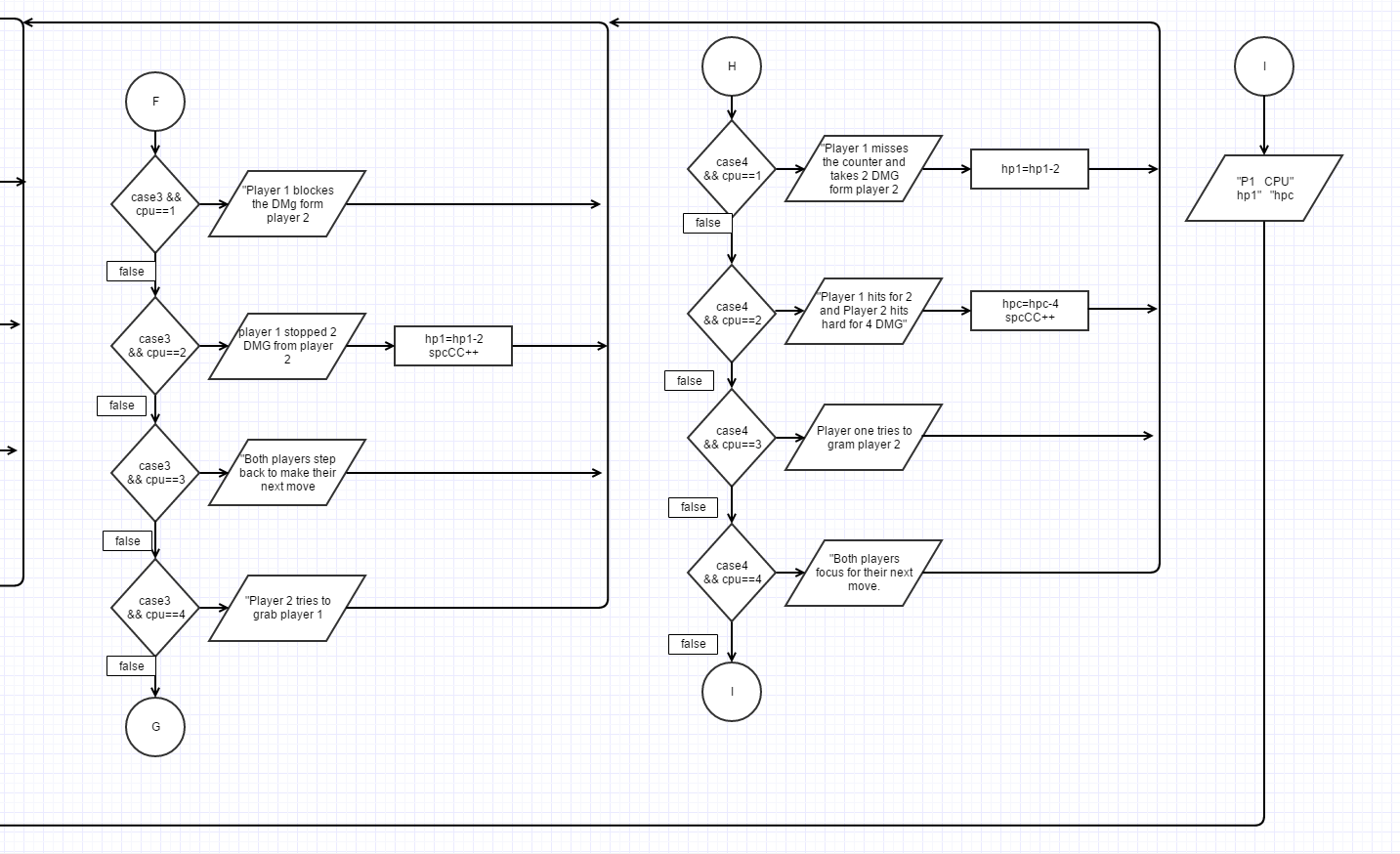
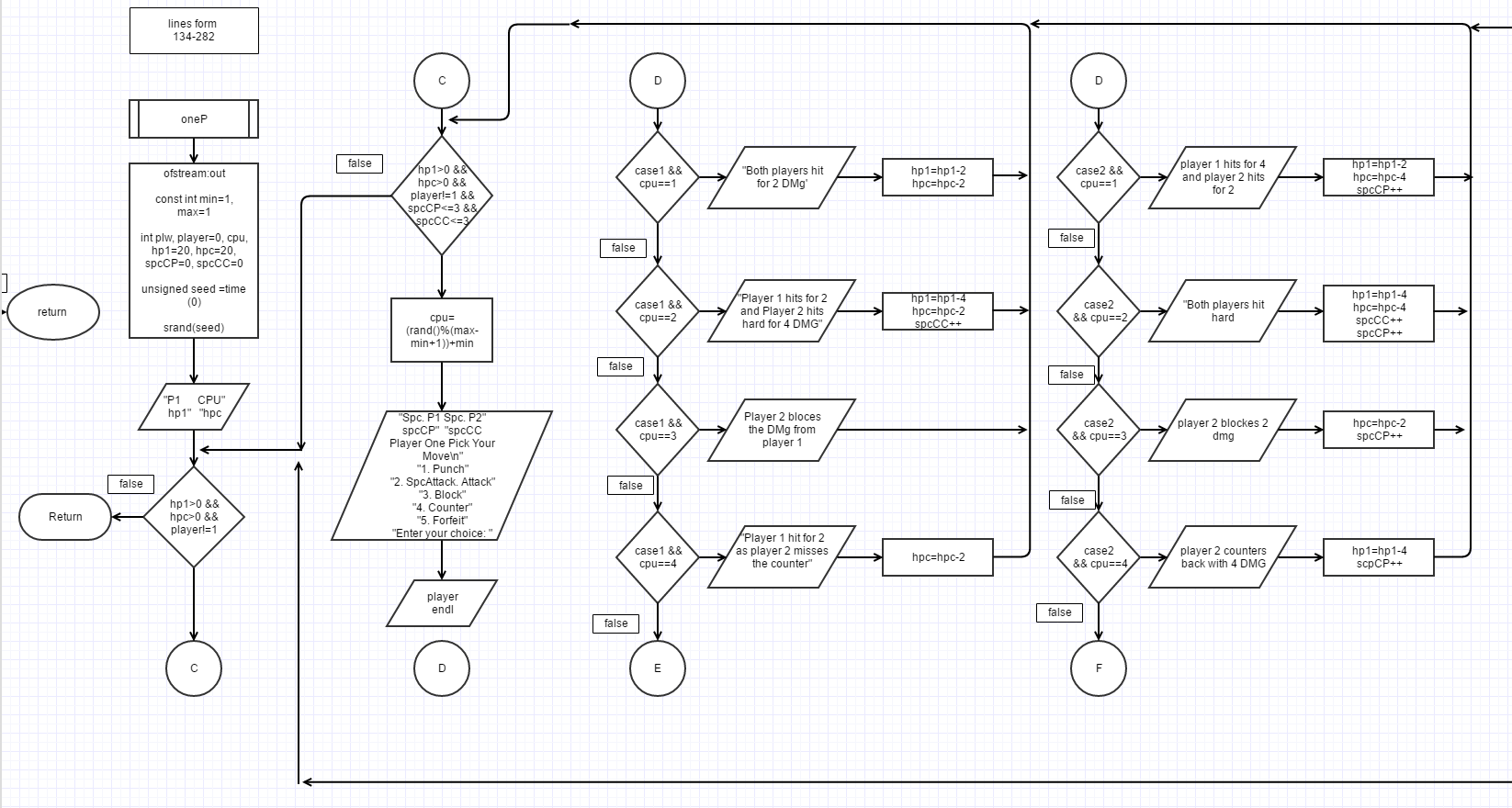
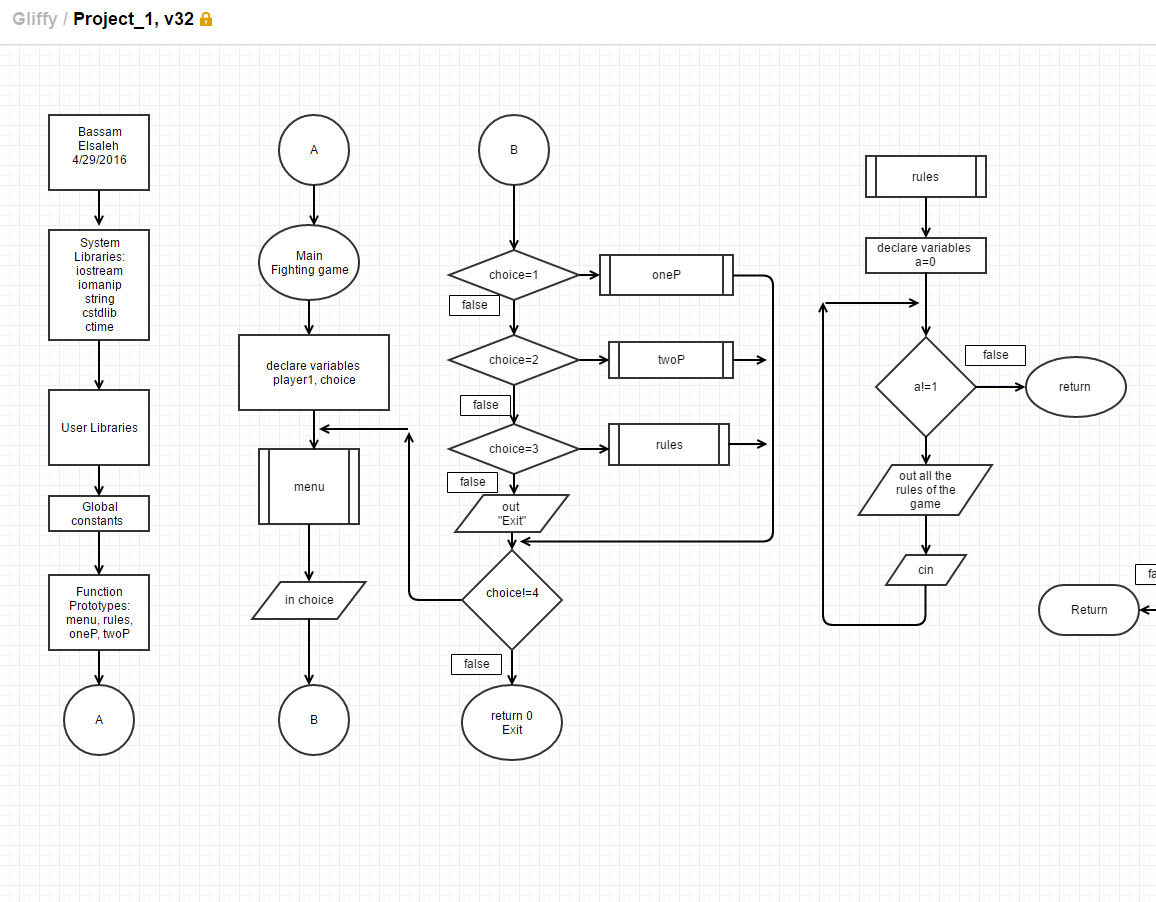
//if player one gives up

//if player two gives up

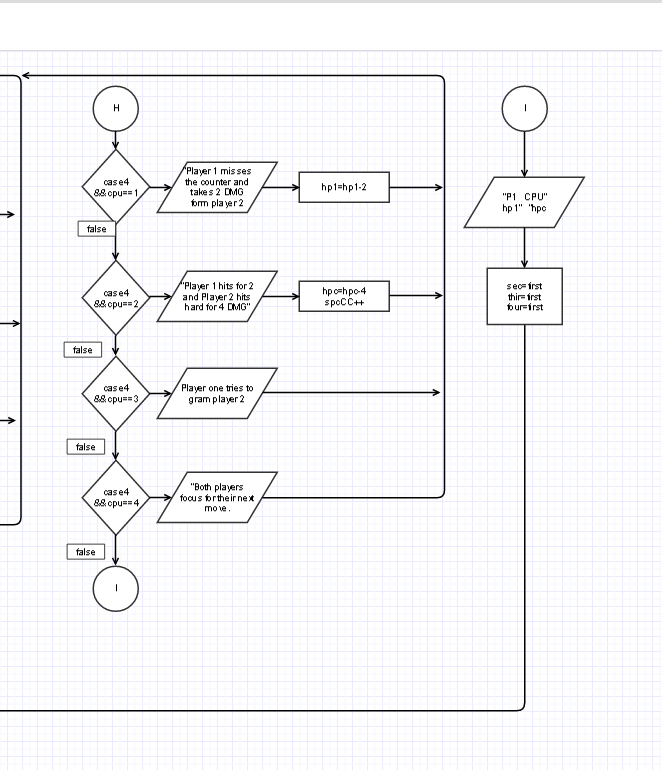
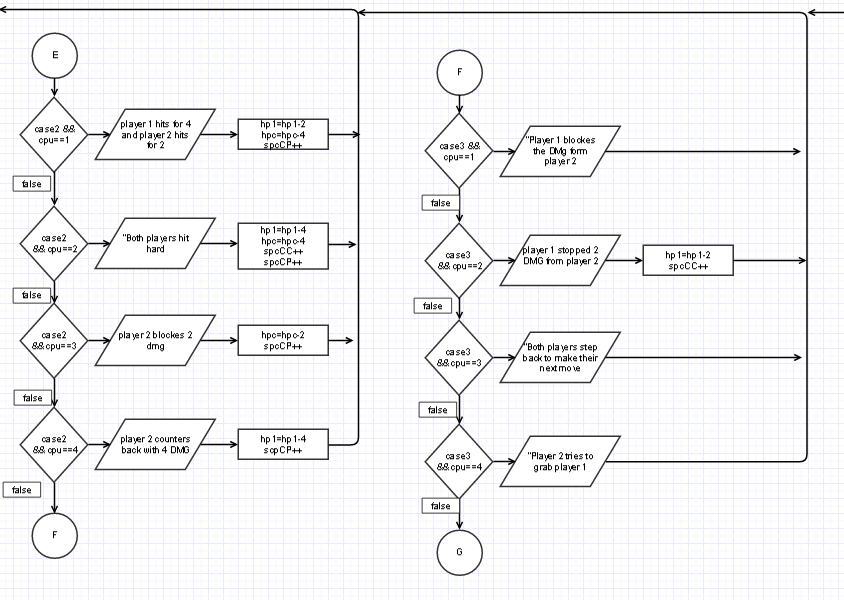
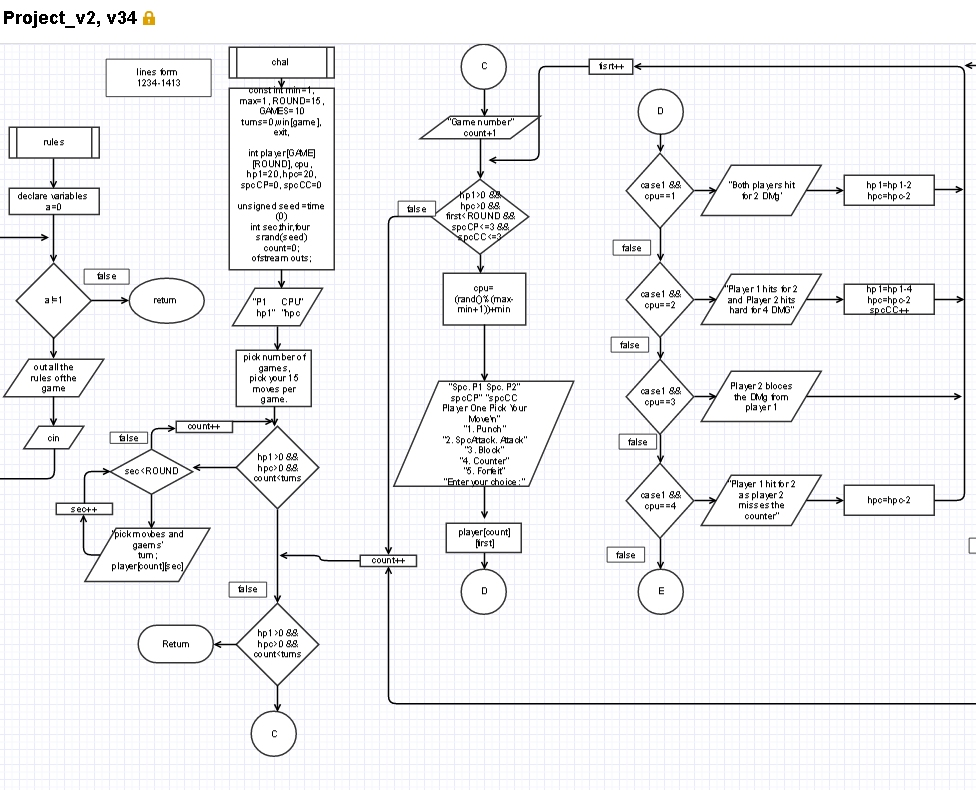
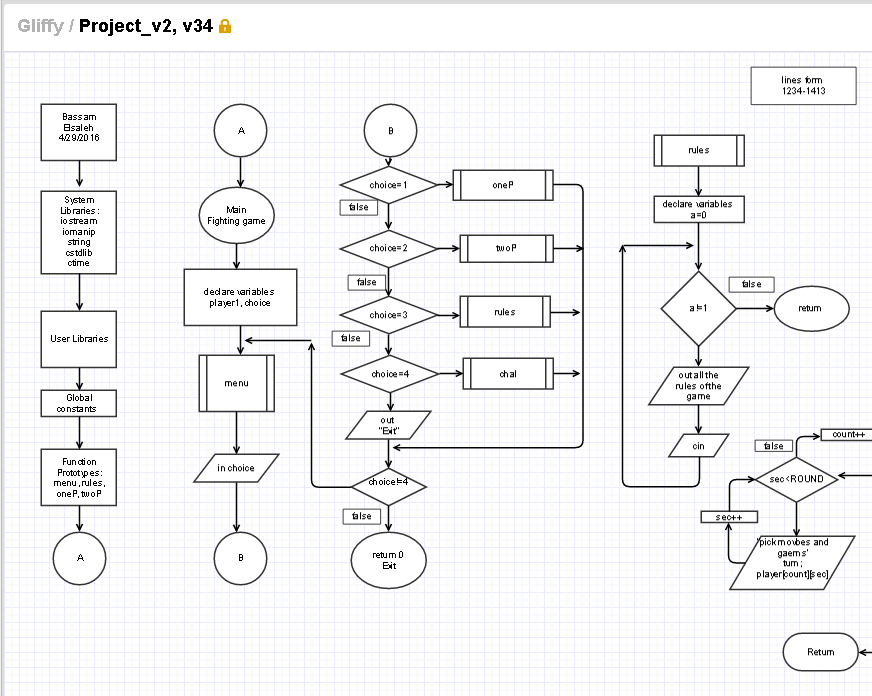
//if player one has more health

//if player two has more health

Flow Chart #1

First set of flow charts to Project 1

Flow Chart #2

Second set of flow charts for Progect 2

Constructs used

Do loops: I mainly used the do loop in the menu so that it runes right into the menu and allowing the player to pick their game mode.

For loops: I used this in the rules section when I didn’t need to increment but I set a variable for what the user will input to exit the statement when they press 1

Switch statements: I believe that using the switch statement was the best way to code with the least amount of lines, rather than using an outstanding amount of it and if else statements I decided to run a switch statement that only runs through the second players options that exists inside player ones switch statement.

If, else if and else statements: my code was riddled with this type of code to decide the outcome of the player’s hp and win lose.

Rand: This was only used in single player where the computer would need to pick its on moves. So I had to bring in cstdlip file and ctime to make sure the random generator wasn’t reporting the same random numbers.

Functions: I decided to use function in my project because doing one player allows for a whole set of rules and outcomes that I found were better to keep all on its own so that the in 2 player mode there wouldn’t be any miscommunication in the code.

fstream: I decided to send an out file that will occur if you win the computer in single player.

Data Types: I used primarily isn’t in my codding no floats were needed because I didn’t have any decimals, I used ofstream to read out to a file, and had set the constant in for a max and an min for the random generator. Along with the random generator I needed and unsigned seed and srand to make the random work.

Arrays: I used these only in the new mode with double arrays, one representing the game number then the second being the round in that game.

Search: I only put this in mainly because I had to. So I tried squeezing it in where it searches the win of the first player.

Sources

Gaddis, T. (2012). Starting out with C. Boston: Pearson Addison-Wesley.

Rcc Class. Dr. Lehr.