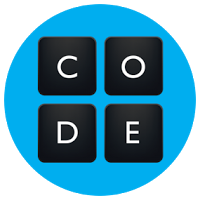
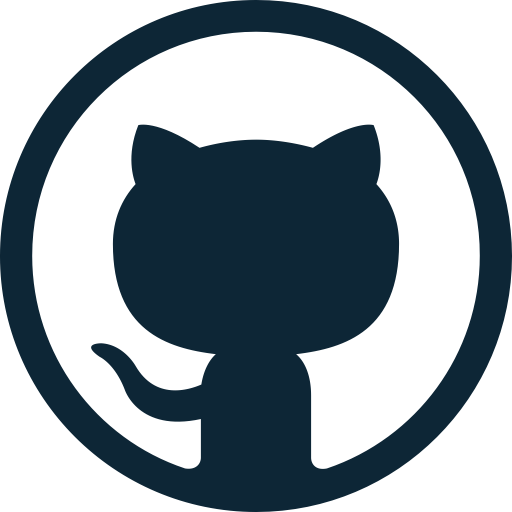
Software Engineering



Assignment 3



Patrick Lӧwe & Mark Hartnett

Contents

1. Specification 3
2. User Manual 4
3. Structured Programming Chart 10
4. Data Dictionary 11
5. Test Case Dataset 12
6. Development Log 13
7. Code 14
8. References 30
9. Declaration of Authorship 30

## GitHub: https://github.com/patricklowe/Game

# Specification

## Introduction

Using C we created a program that would accept between 2 & 6 players, allowing each user to create a Name, Type and then be assigned to randomly generated slots. Once assigned each player can then move forwards or backwards 1 slot, or attack the nearest player.

# Narrative

The name of this program is *main.c*. This program will allow the user to create players with unique names and types, then automatically assign them appropriate attributes and insert them into a randomly generated slot. It requires a password to access and if an invalid password is entered it will reload the main function after 3 seconds. Once each player has a slot, they have the option to move slots or attack the nearest player. After each player has chosen an option the program will offer the user another round (to continue attacking or moving slots), otherwise it will exit the program gracefully.

User Manual

Step 1

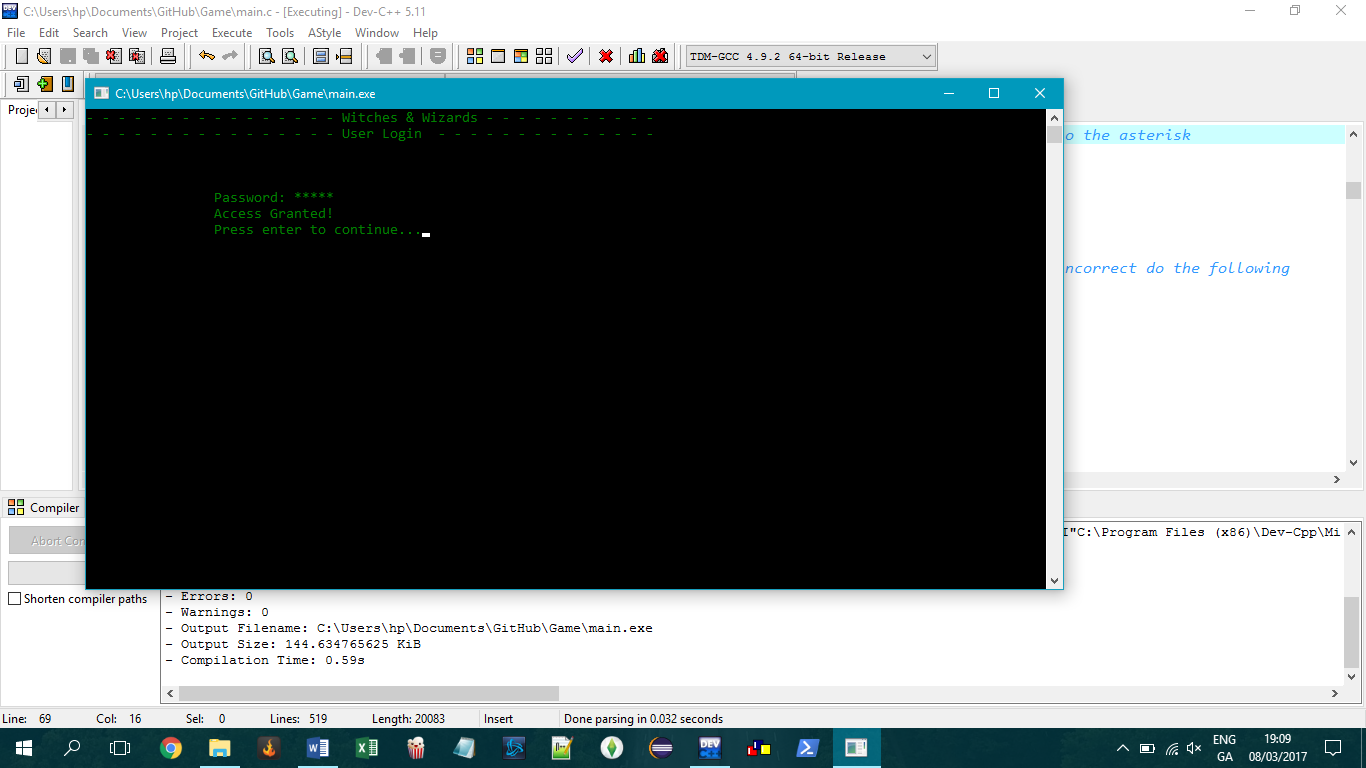


Fig.1 Login accepted

When the program starts up there will be a banner with the program name and current function details. The user is prompted to enter a password, if invalid the program will wait 3 seconds before reloading the password page (Fig.2)

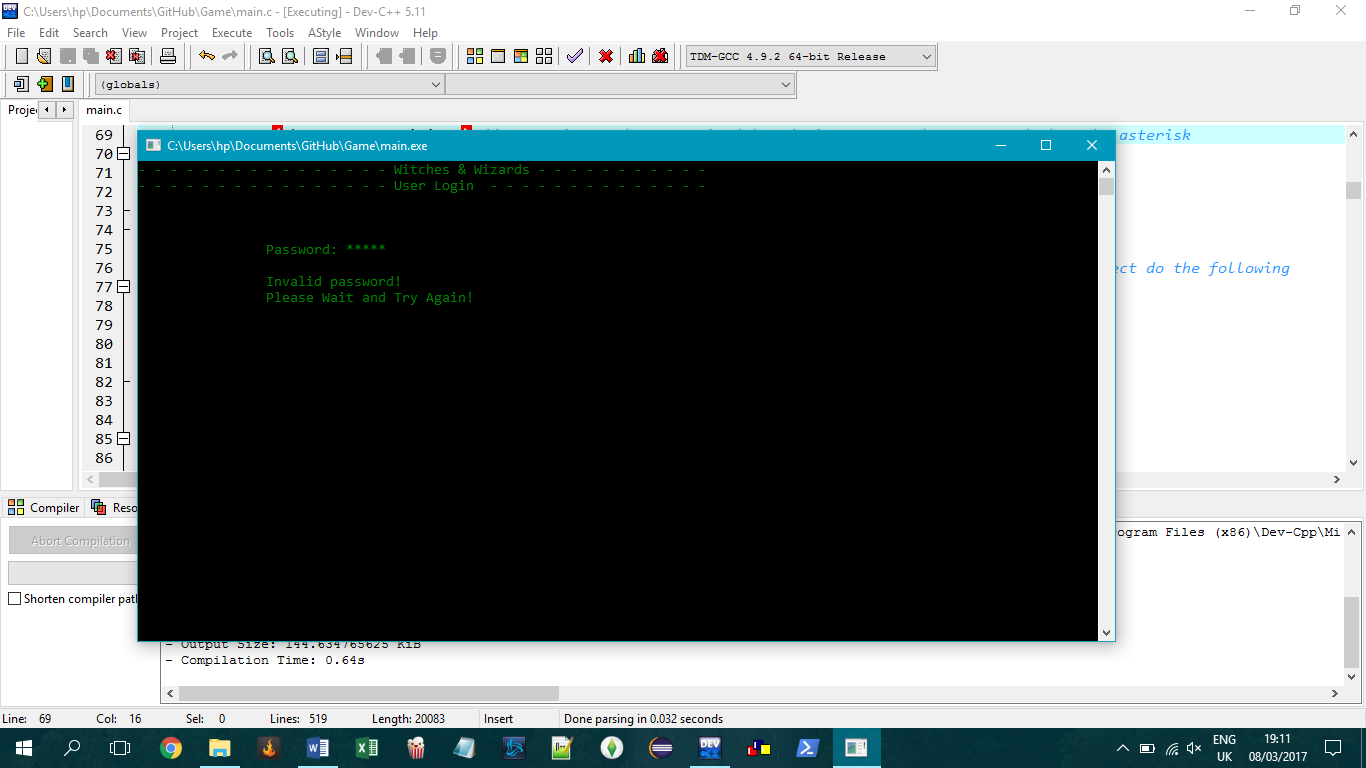


Fig.2 Login failed

If the password is valid, the user will have access to the game (Fig.1)

Step 2

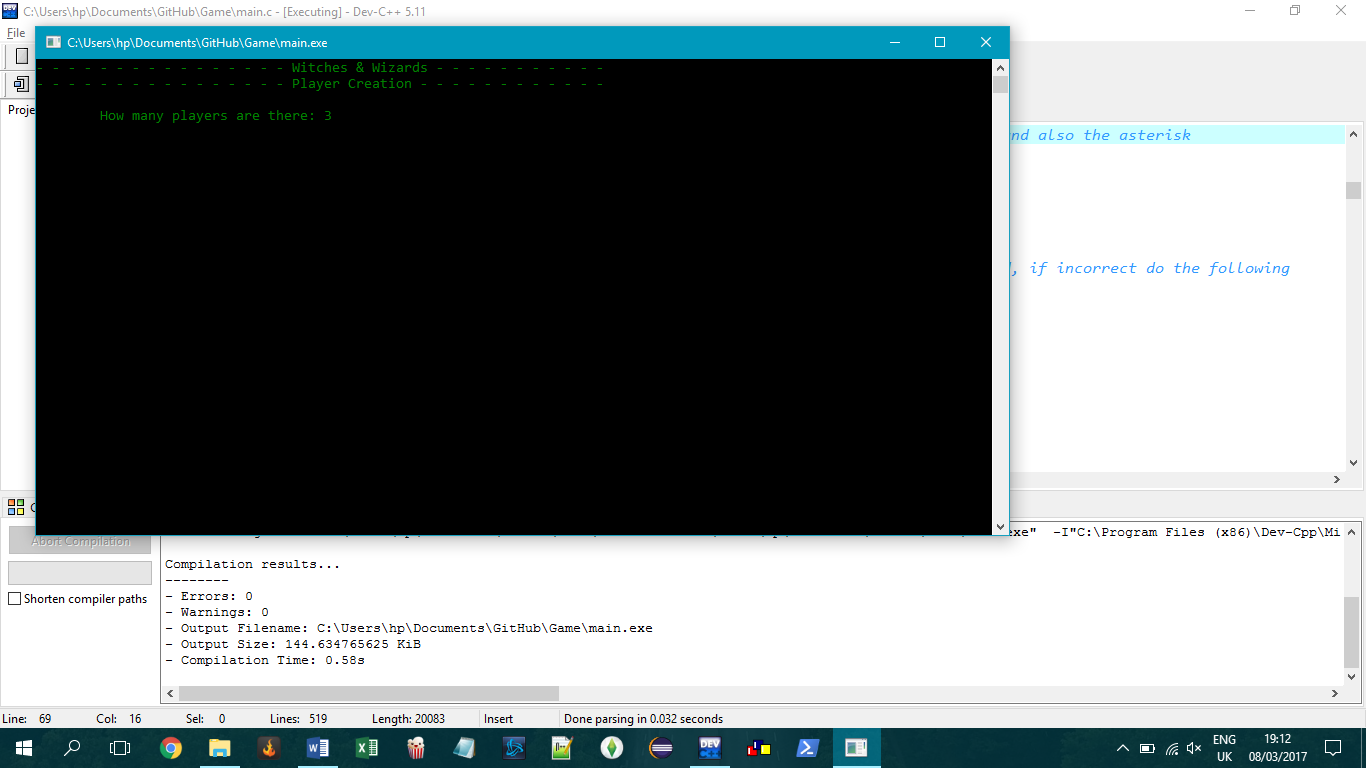


Fig.3 Player Limit

The User will then be asked to input the number of players. If they enter less than 2 it will correct them to 2 minimum (Fig.4) If they enter more than 6 it will correct them to 6 maximum (Fig.5)

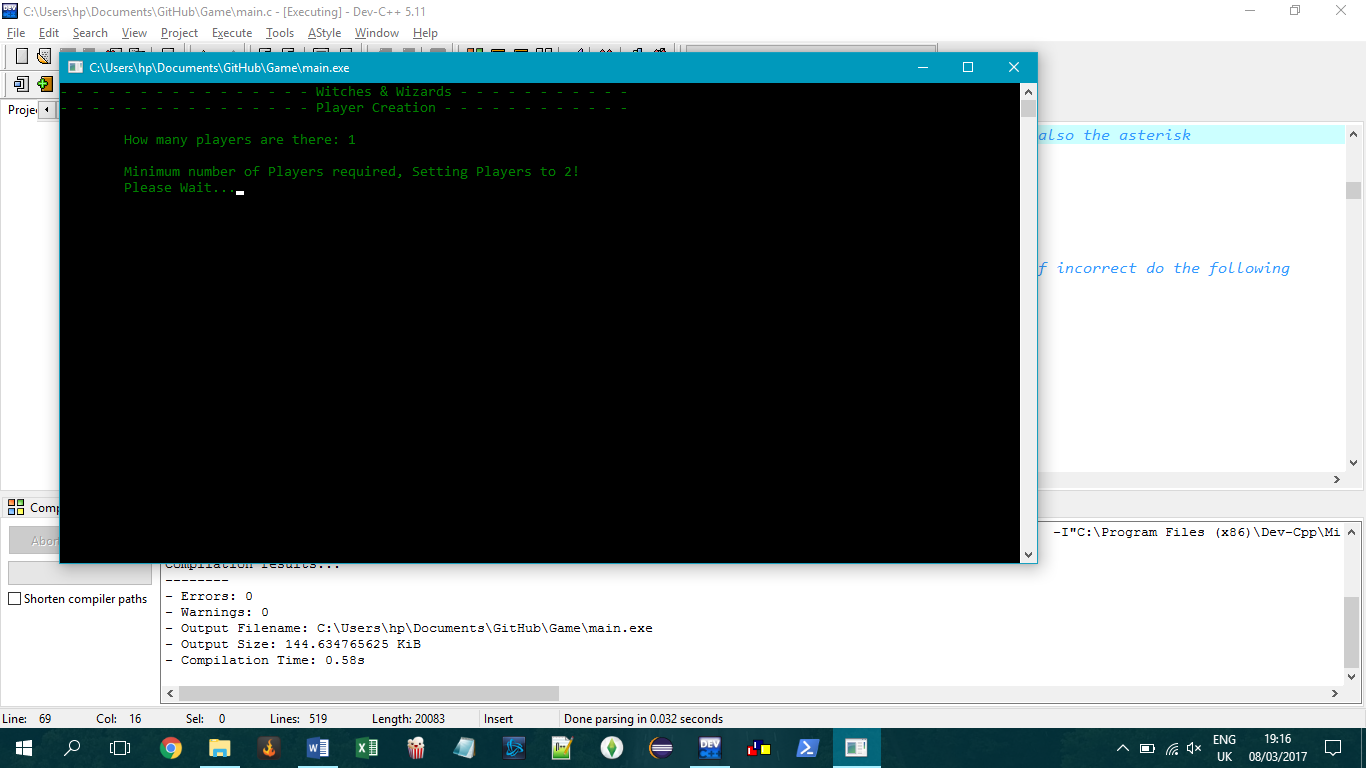


Fig.4 Minimum Player Limit

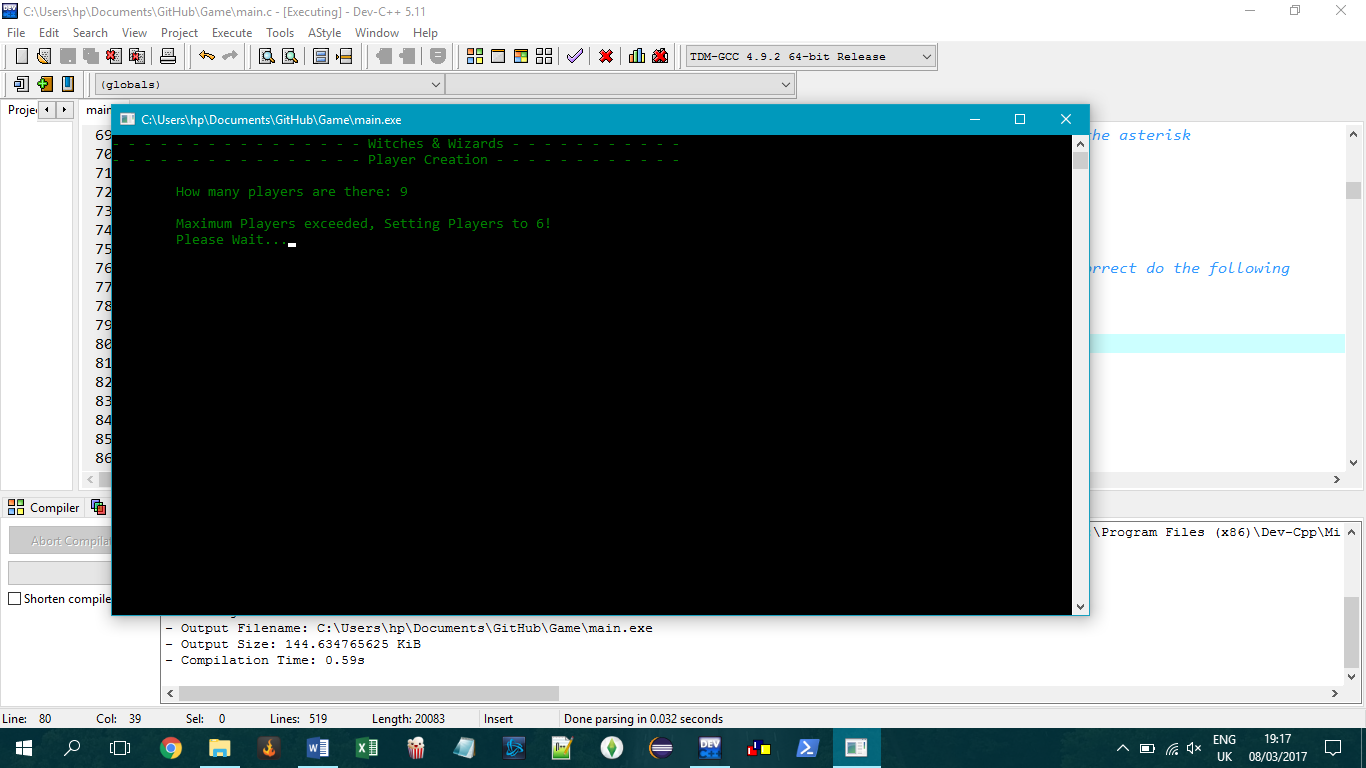


Fig.5 Maximum Player Limit

Step 3

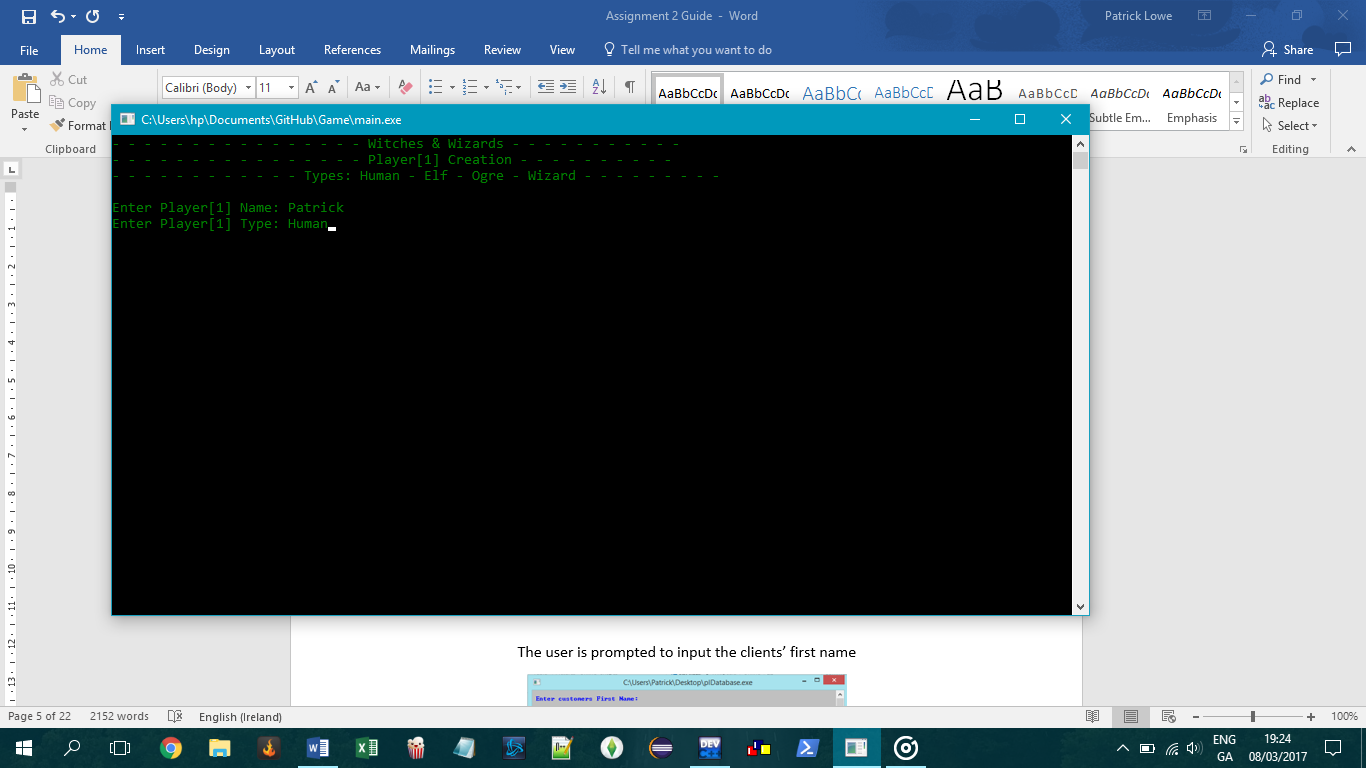


Fig.6 Character Creation

The user can now enter their character name and character type, the available types are labelled above (Fig.6)

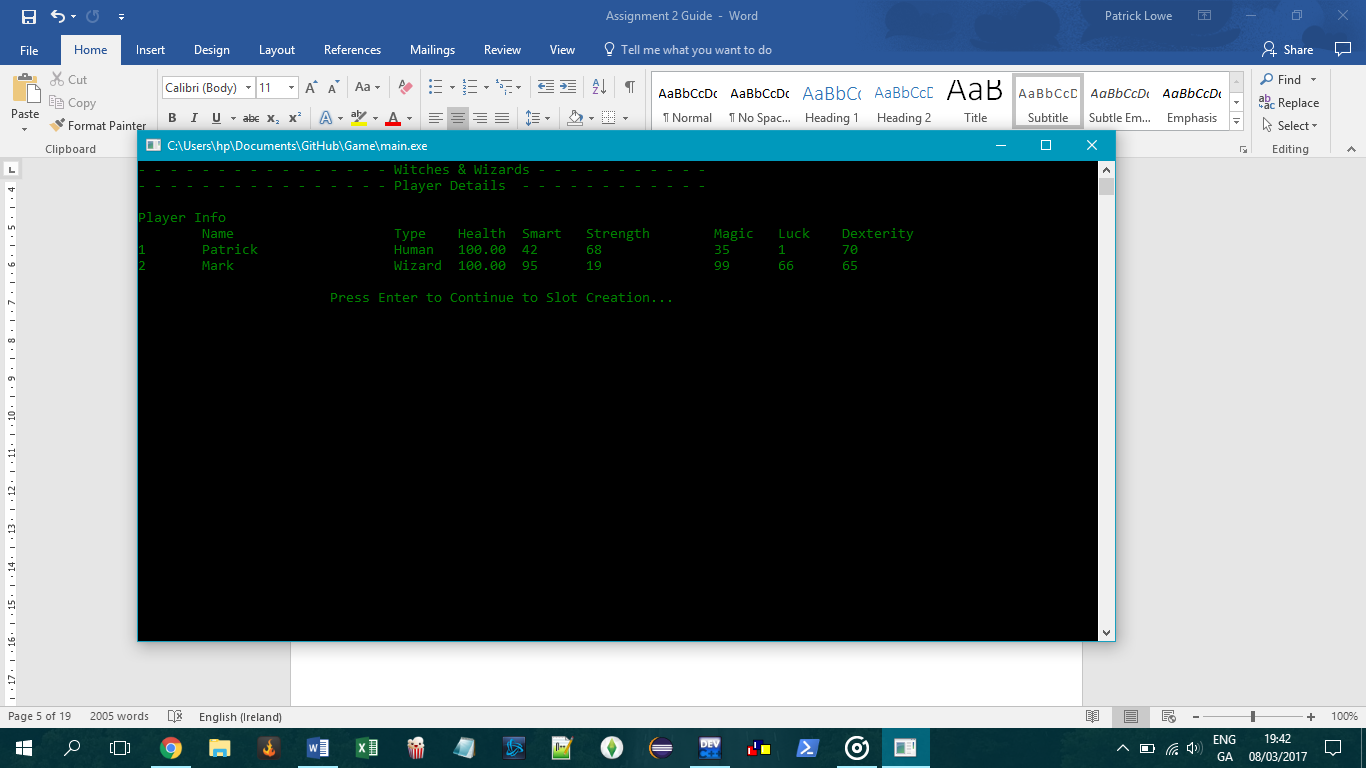


Fig.7 List of Player Details

After the players names and types have been accepted the program will autogenerate random values for their attributes, depending on character Type (Fig.7)

Step 4

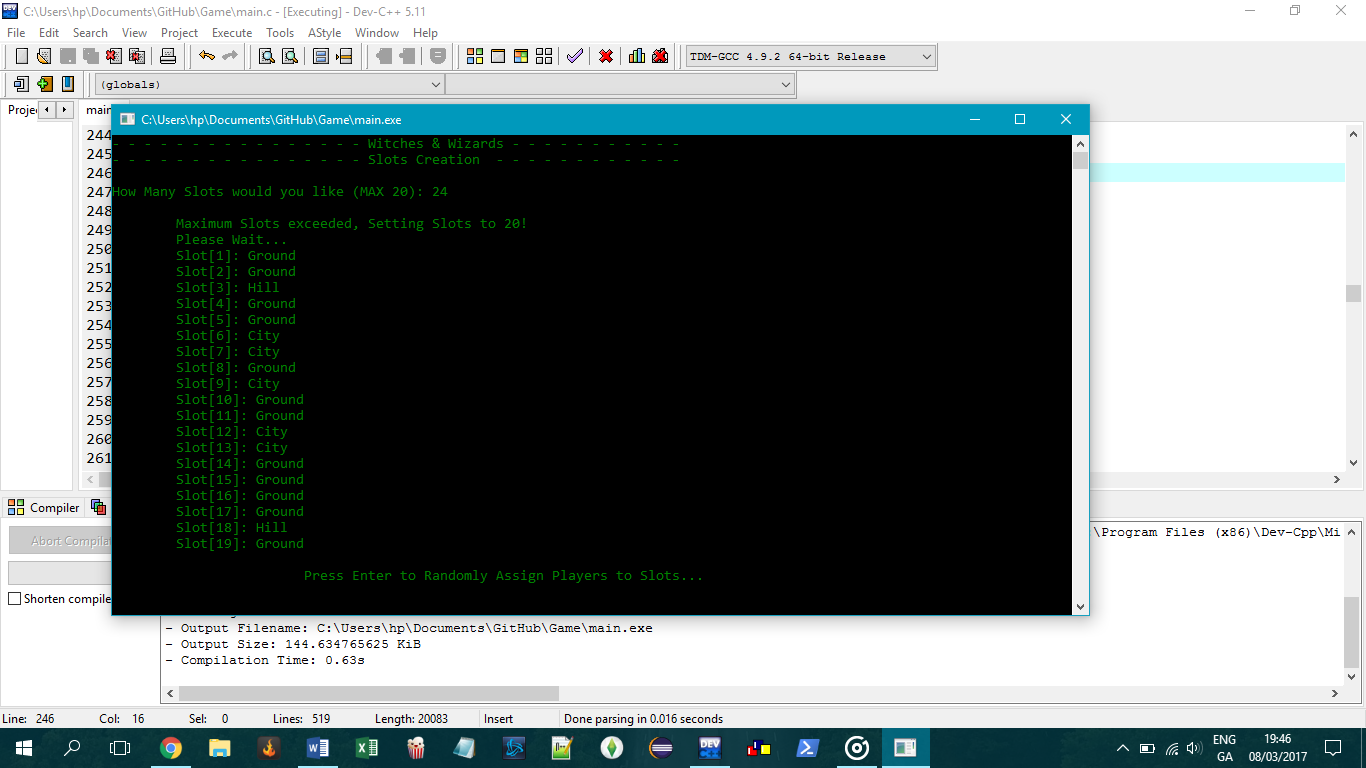


Fig.8 List of Maximum Randomly Generated Slots

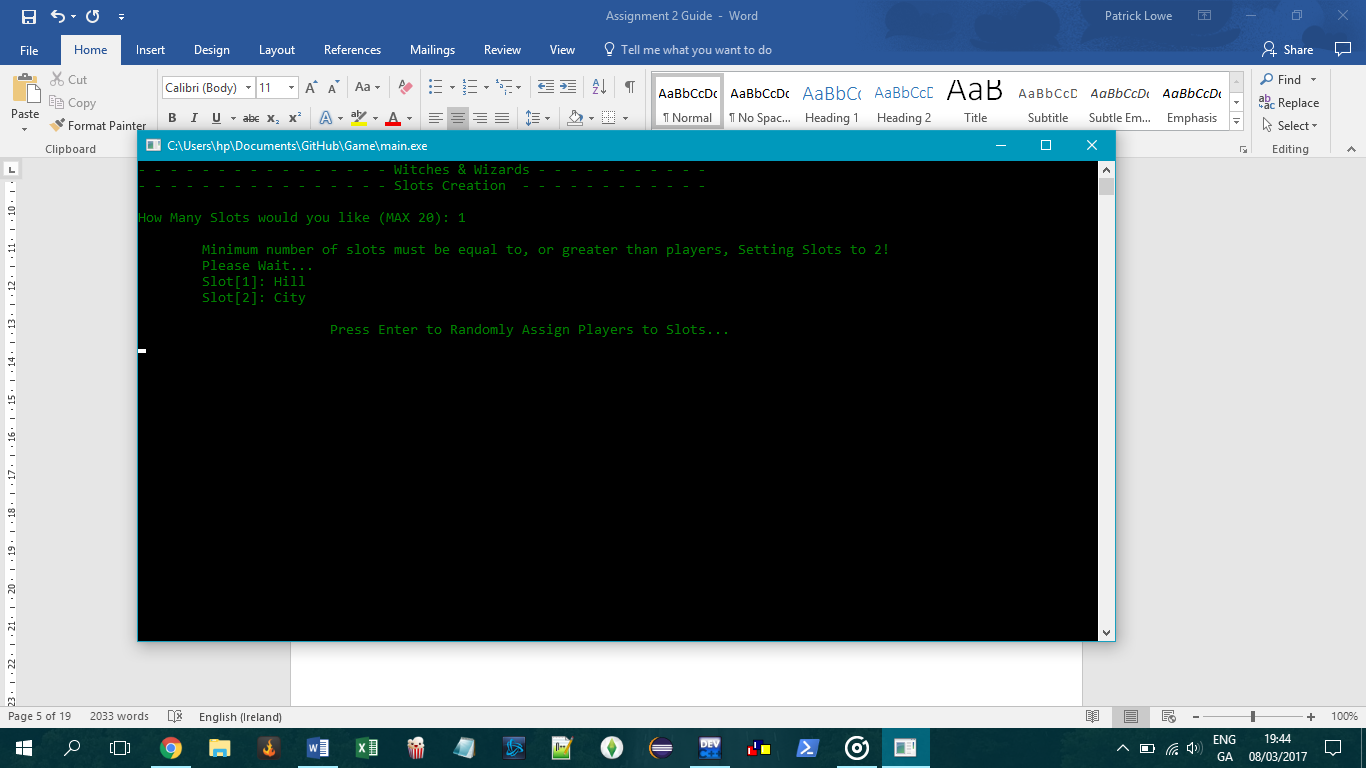


Fig.9 List of Minimum Randomly Generated Slots

After character creation, the user will be asked to enter a number for slots to be generated. If the user enters a number lower than total players the minimum slots will be created based on player limit (Fig.9)

If the user specifies more than 20 slots the maximum of 20 slots will be randomly generated (Fig.8)

Step 5

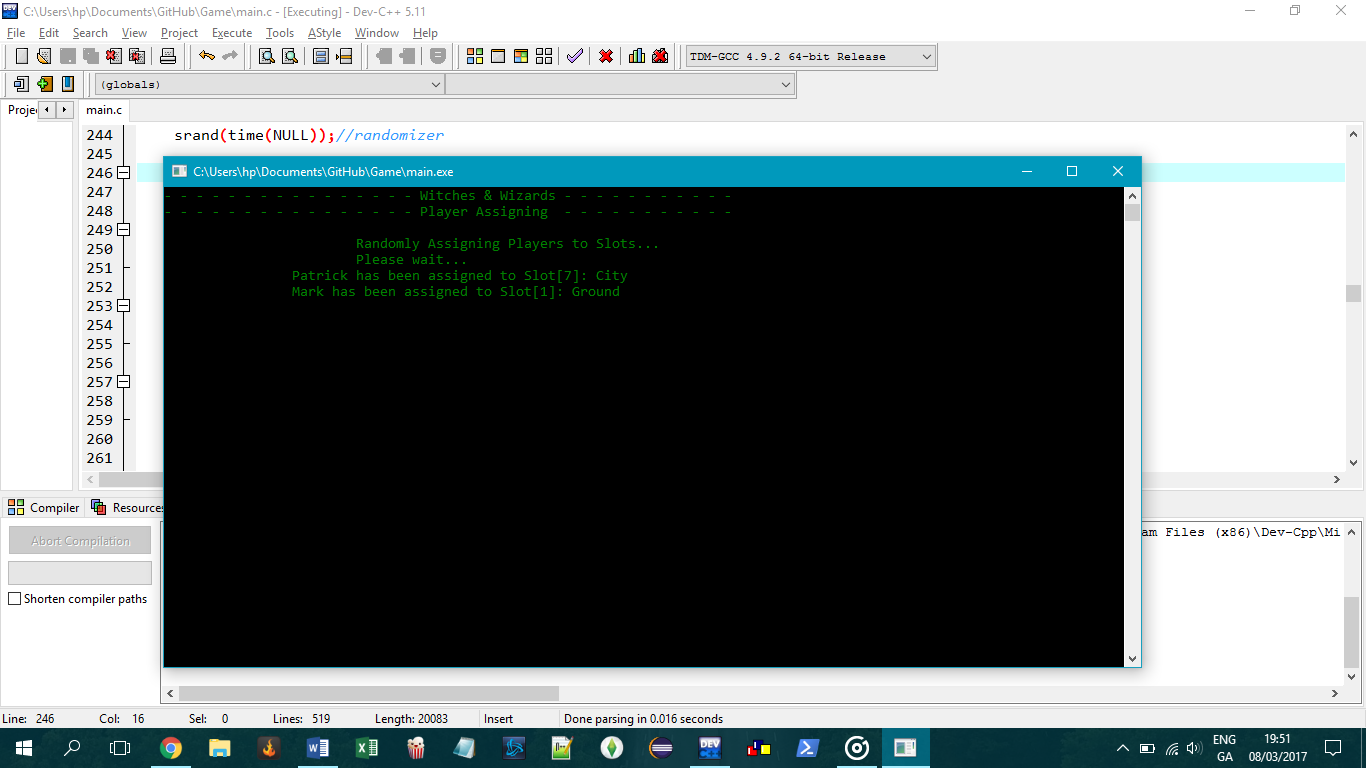


Fig.10 Players Randomly Assigned to Slots

After pressing enter players will automatically be randomly assigned to slots (Fig.10)

Step 6

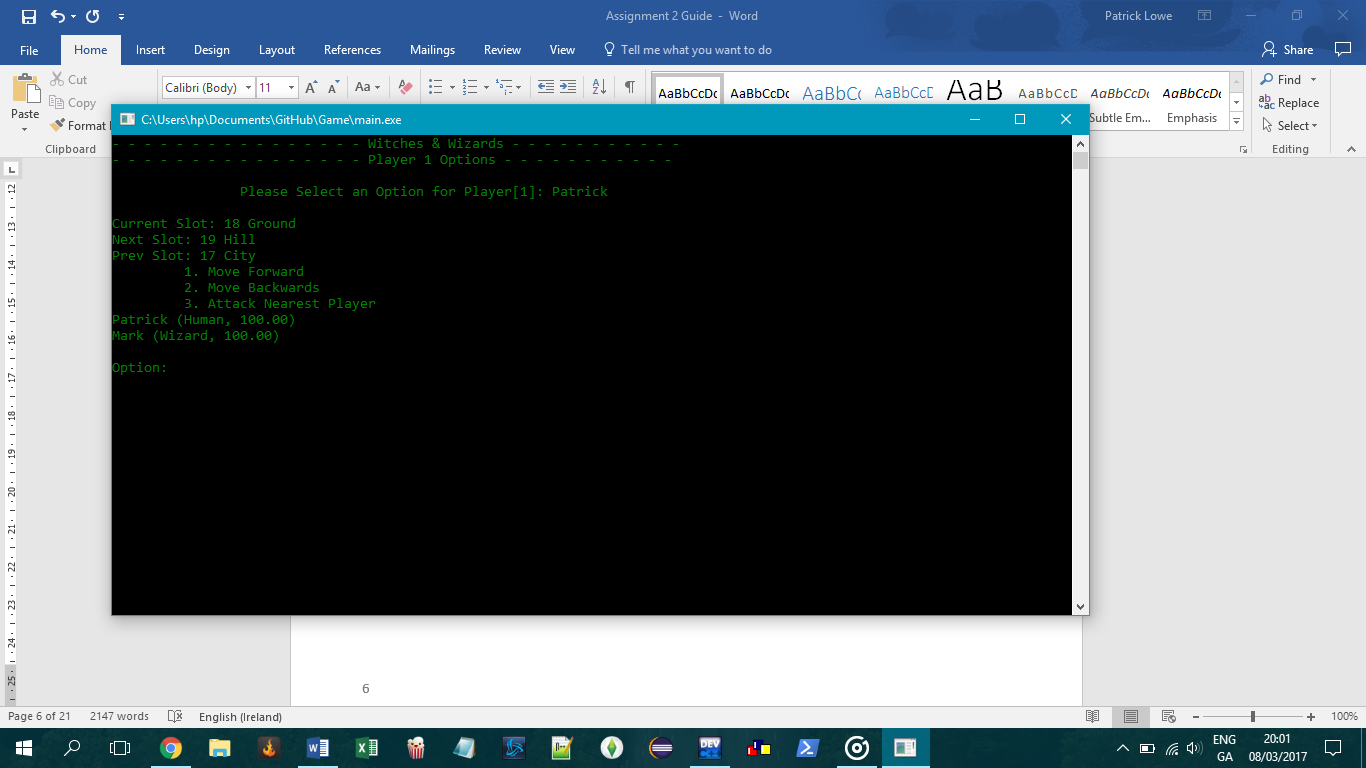


Fig.11 Player 1 Options

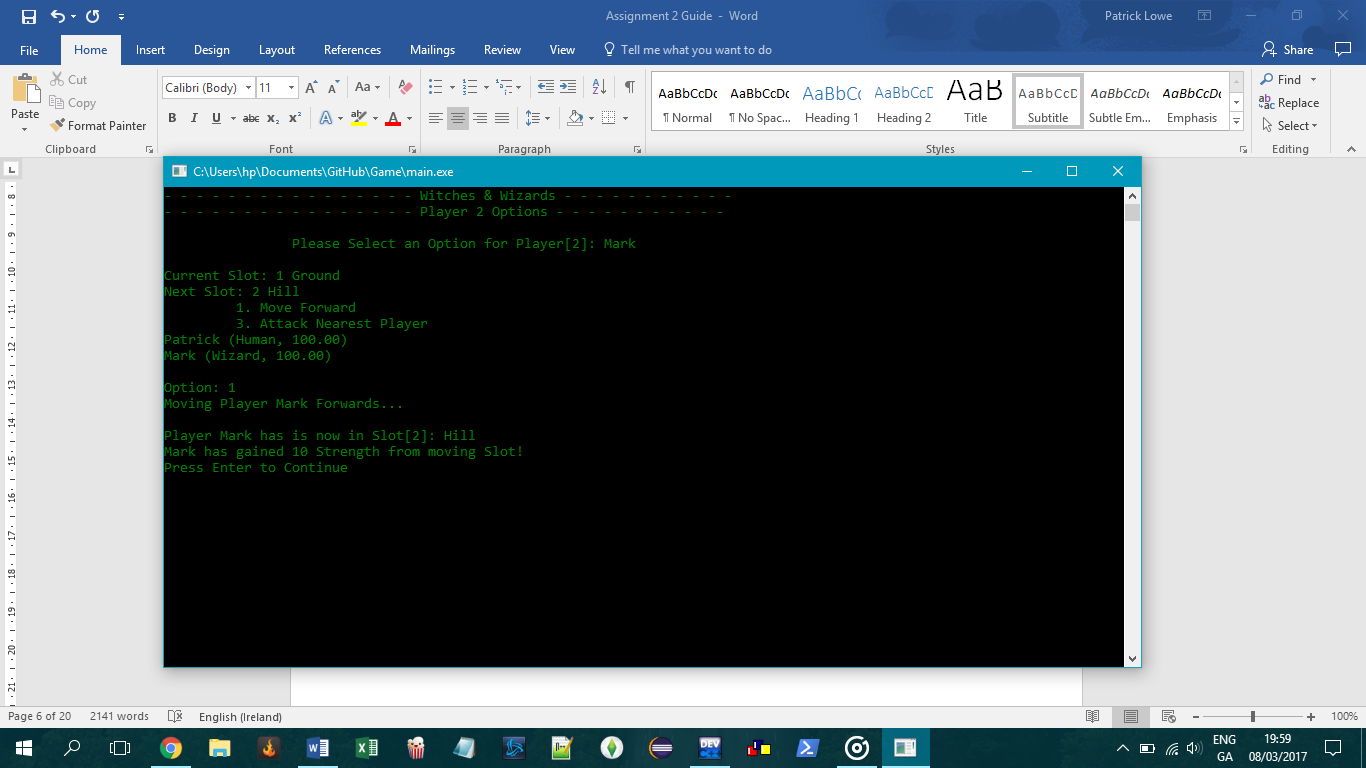


Fig.12 Moving Forwards

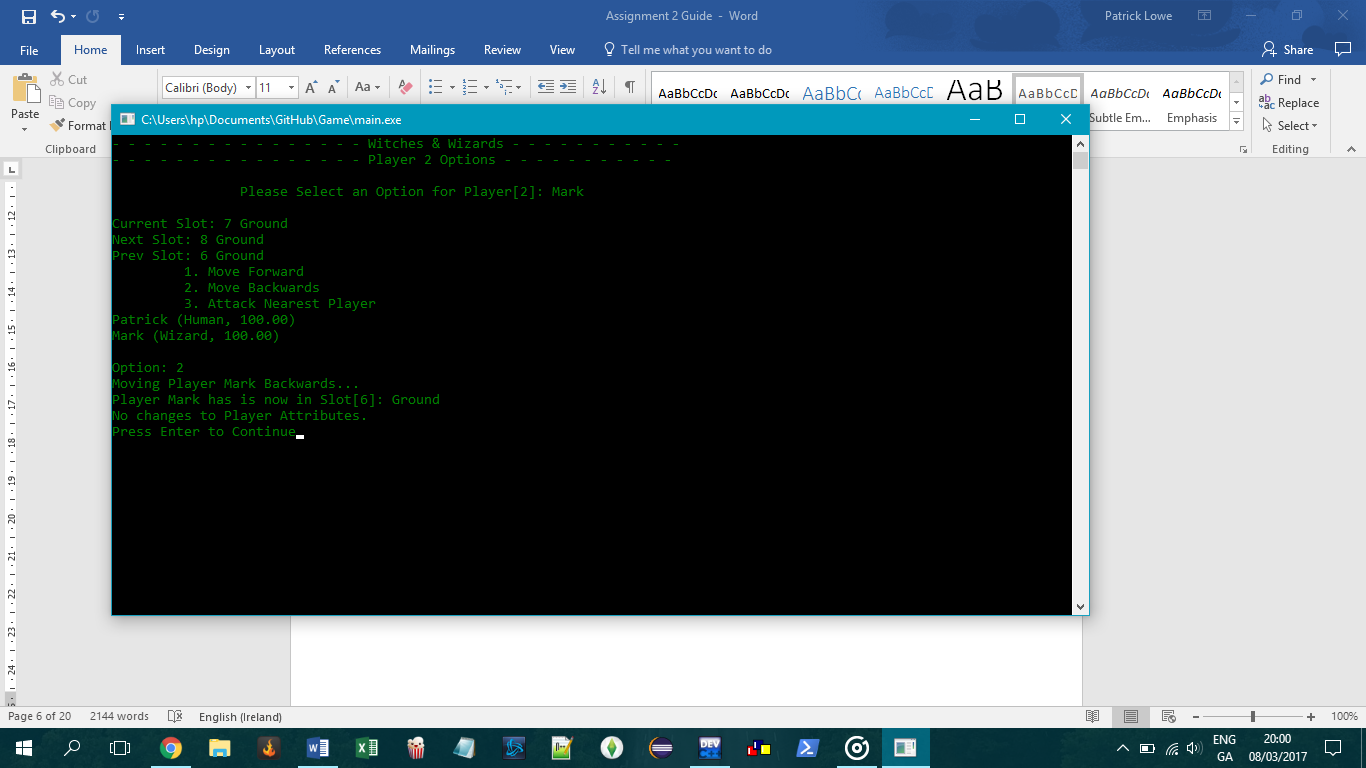


Fig.13 Moving Backwards

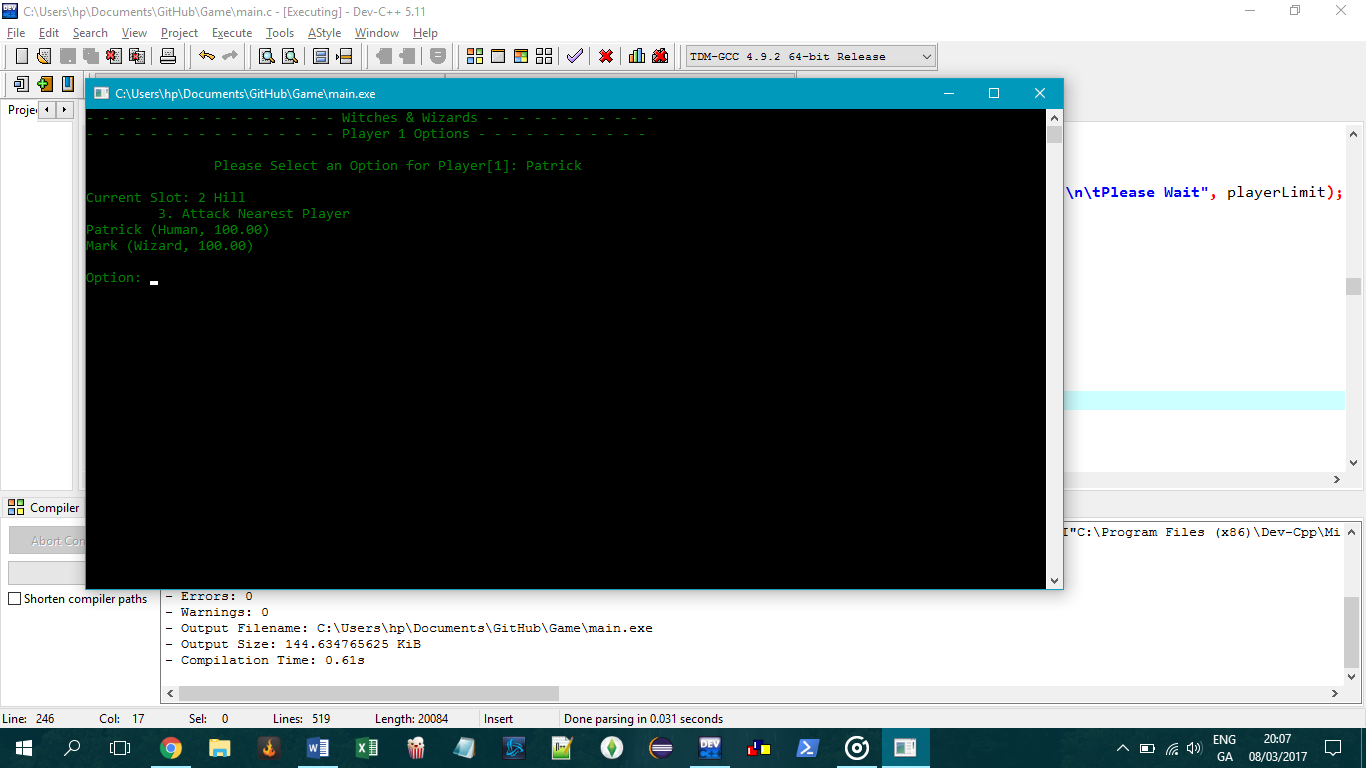


Fig.14 No Available Slots

Users will be shown their Name, current slot, next slot, previous slot, and options based on their current position (Fig.11). If they are at the beginning they can only move forward (Fig.12). Hence, if they are at the end of the slots they cannot move forward, only backwards (Fig.13). If no slots are available, the player can only attack (Fig.14). The details of all players will be displayed before an option is selected. (Fig.11-14)

Step 7

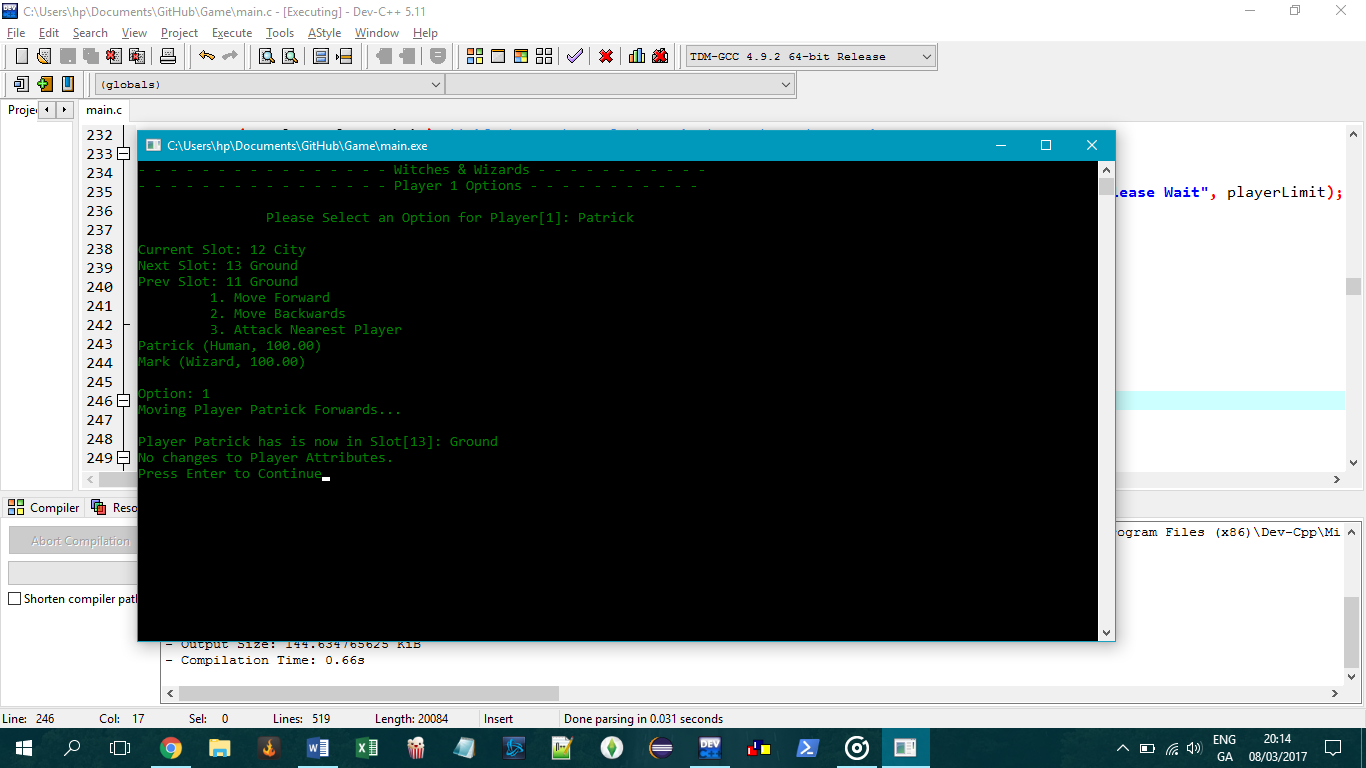


Fig.15 Moving Forwards

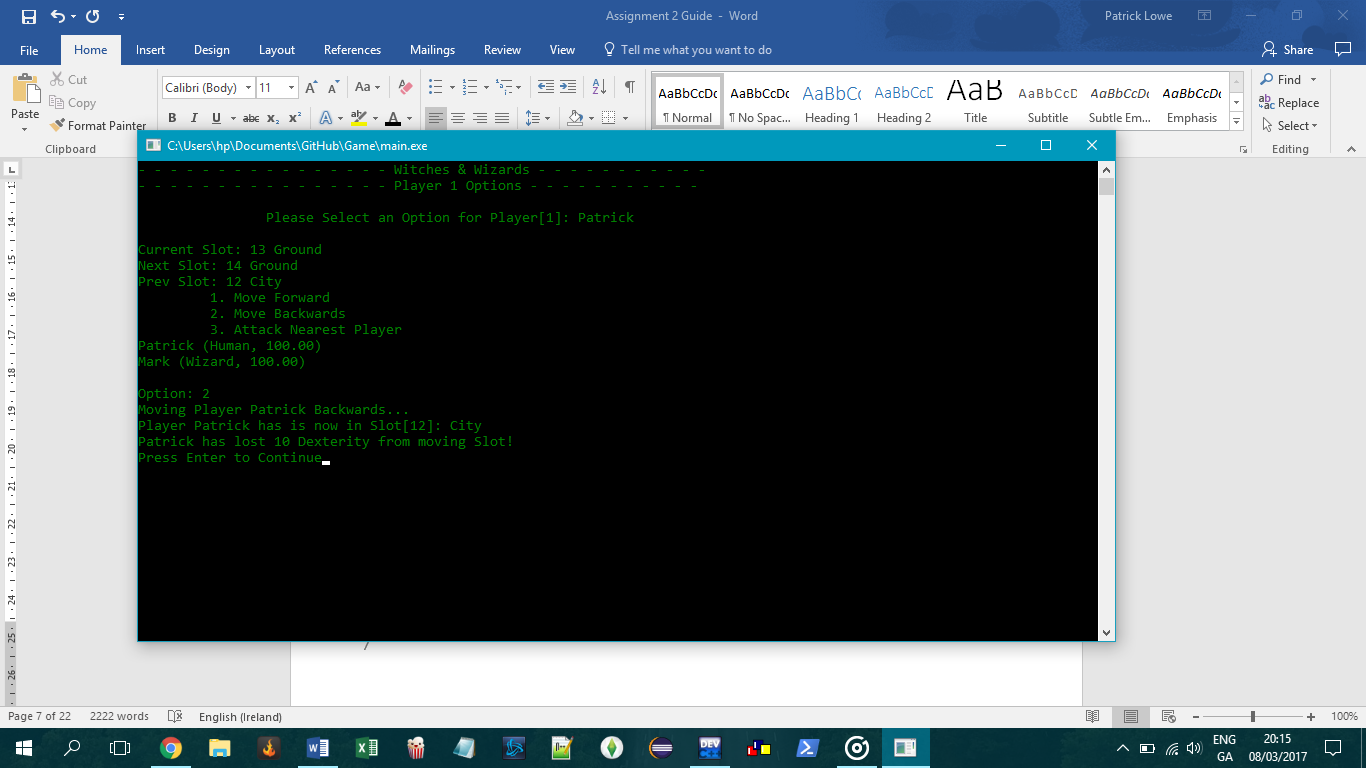


Fig.16 Moving Backwards

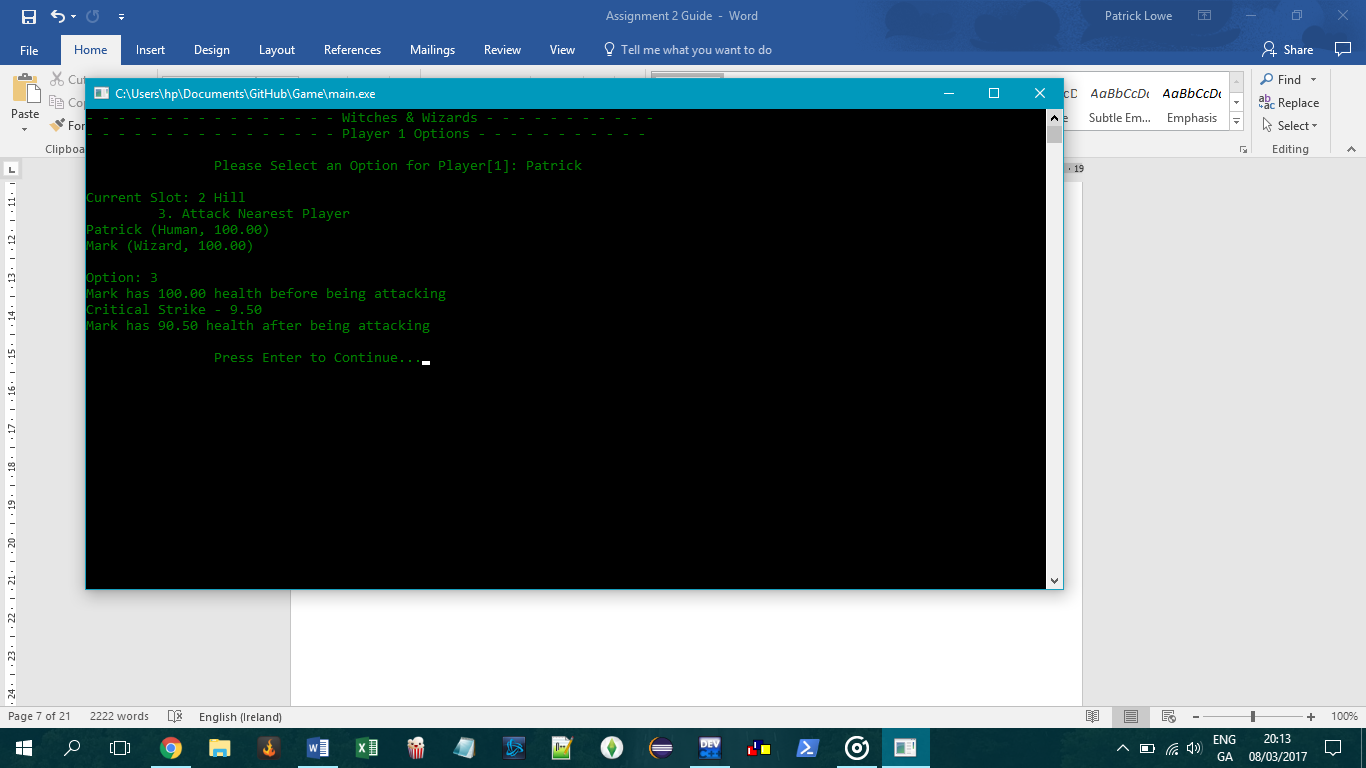


Fig.17 Attacking

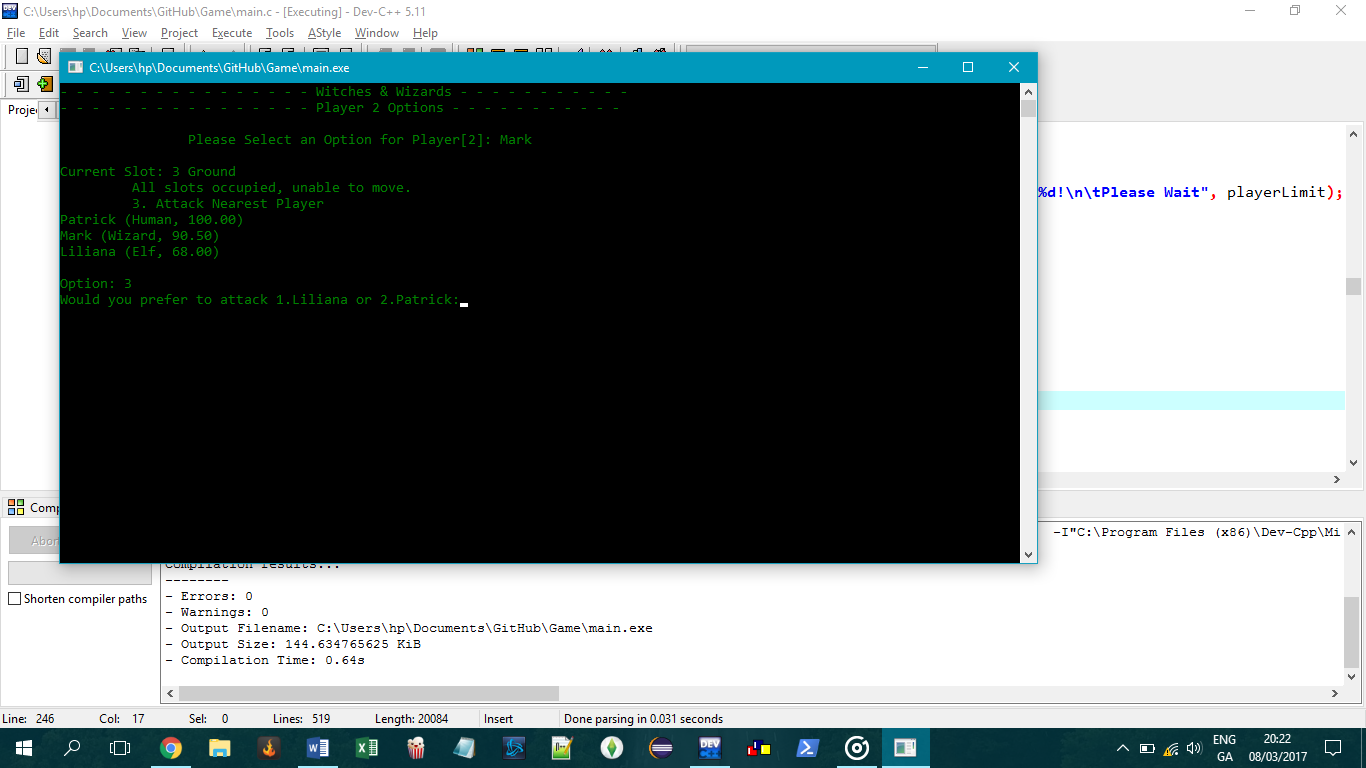


Fig.18 Choosing between 2 Nearest Players

If the user selects to move forwards the program will move them to the next slot displaying their new position and any attribute changes (Fig.15) If the user selects to move backwards the program will move them to the previous slot displaying their new position and any attribute changes (Fig.16) If the user selects to attack the nearest player, the program will show the attacked players health before the attack, how much damage the attack dealt, and the current health of the attacked player after (Fig.17). If there are 2 nearby players of equal distance, the program will allow the user to select between the 2 (Fig.18).

Step 8

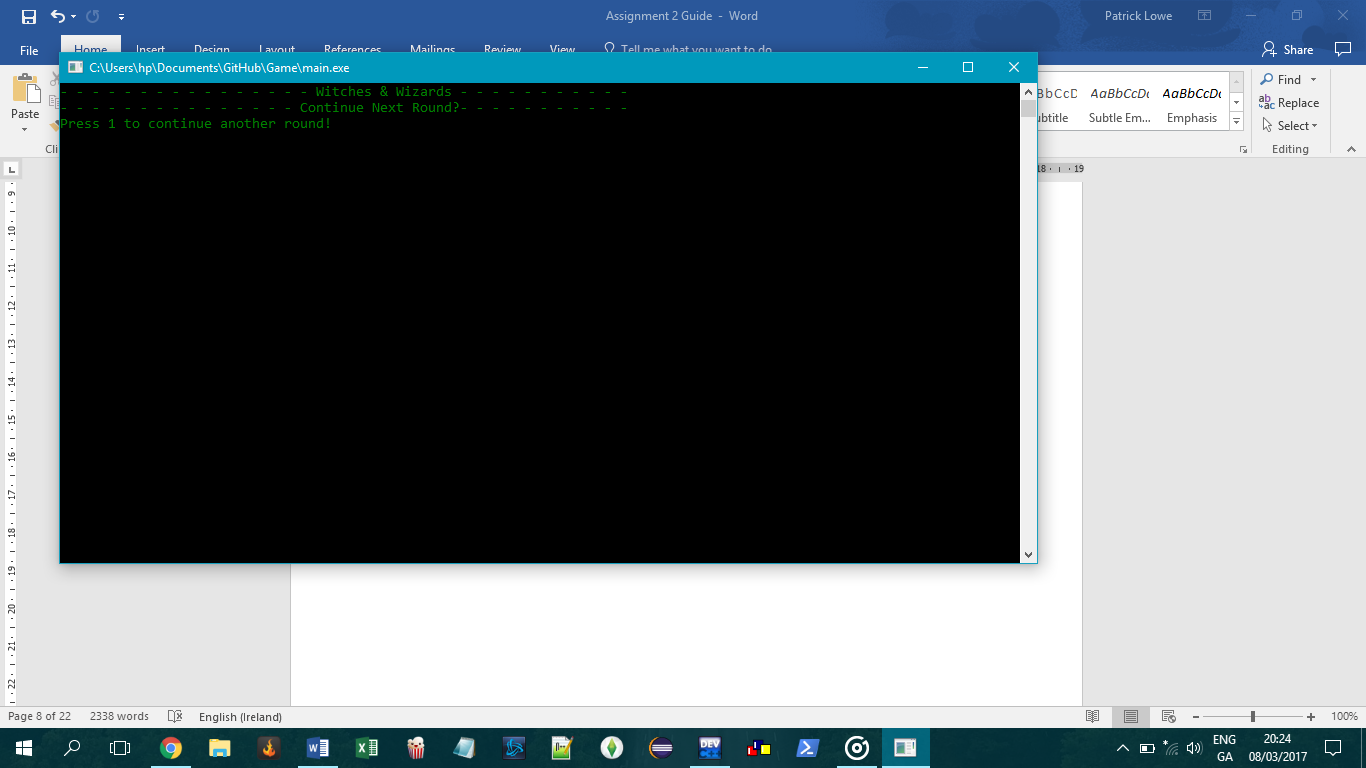


Fig.19 Continuing Another Round

The next screen allows the user to continue another round by pressing 1, or else the program loads the exit function (Fig.20)

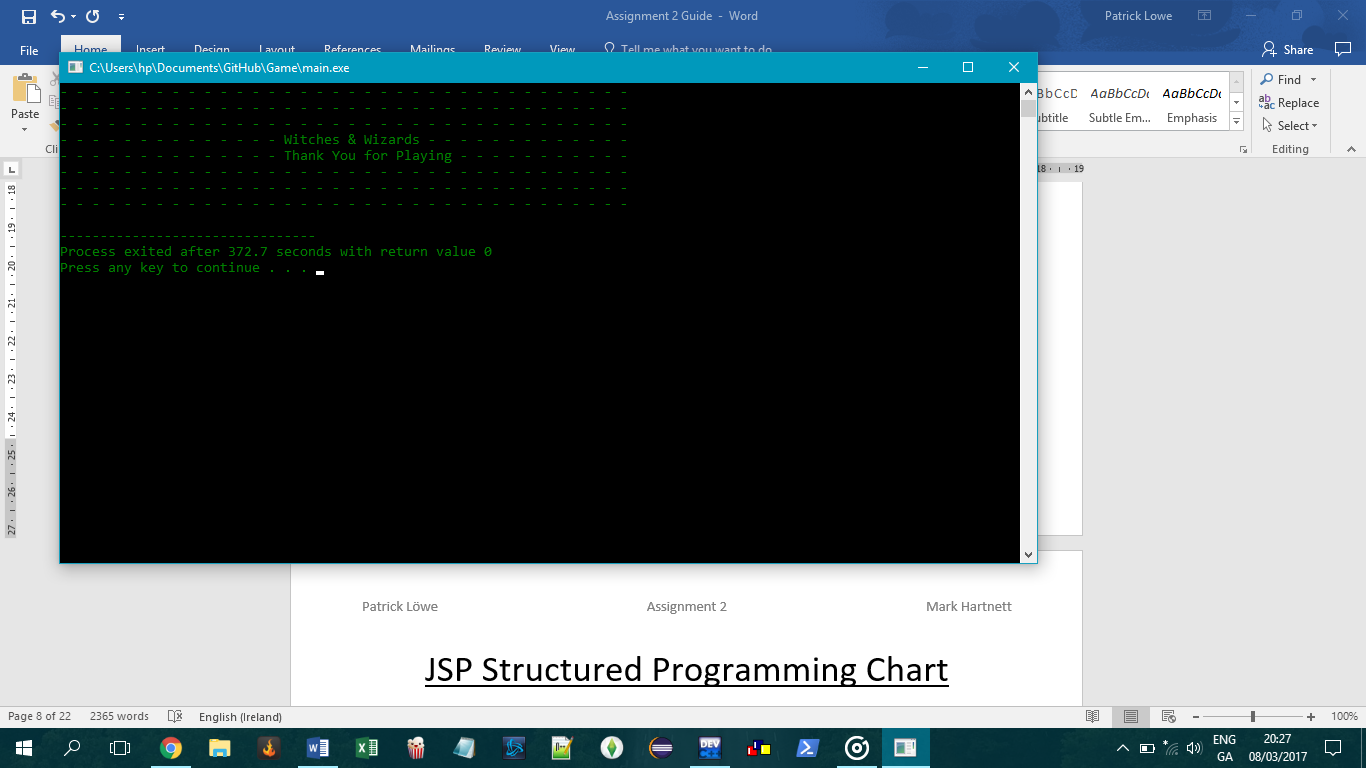


Fig.20 Exiting the Program

JSP Structured Programming Chart

A.

Login

B.

Player Limit

K.

Slots

M.

Slot Type

L.

Slot Num

D.

Type

C.

Name

O.

Options

F.

Health

N.

Assign to Player

E.

Smart

H.

Magic

G.

Strength

\

S.

Attack Player

R.

Move Backward

Q.

Move Forward

P.

Move Positions

J.

Dexterity

I.

Luck

Data Dictionary

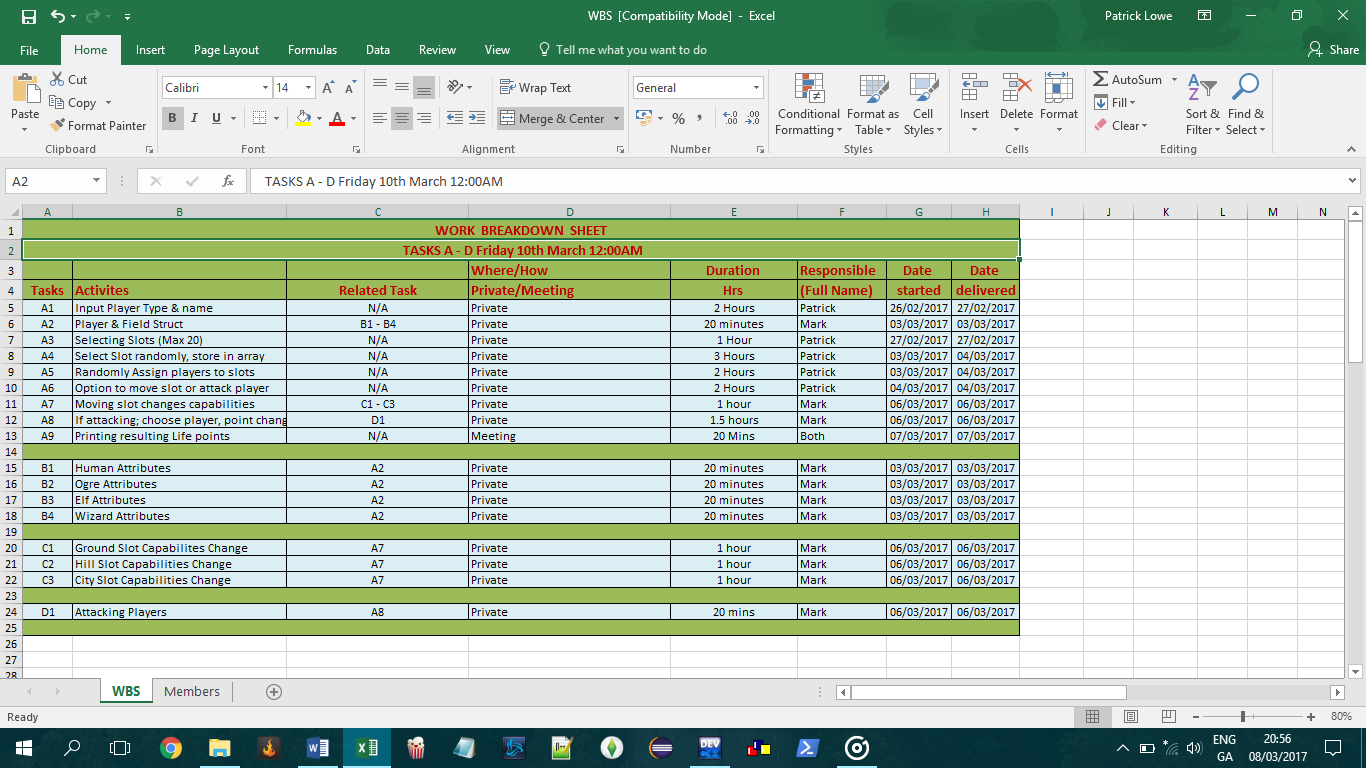
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***main.c*** | | | | |
| **Physical Name** | **Logical Name** | **Format** | **Purpose** | **Test Case Dataset Ref** |
| pName | Character Name | String | Store character Name | Page 12 |
| pType | Character Type | String | Store Characters Type | Page 12 |
| Lifepoints | Health | Float | Displaying Health to 2 decimal places | Page 12 |
| Smart | Attribute | integer | Store Attribute | Page 12 |
| Strength | Attribute | Integer | Store Attribute | Page 12 |
| Magic | Attribute | Integer | Store Attribute | Page 12 |
| Luck | Attribute | Integer | Store Attribute | Page 12 |
| Dexterity | Attribute | Integer | Store Attribute | Page 12 |
| pSlotNum | Character Slot Number | Integer | Store Player Position | Page 12 |
| pSlotType | Character Slot Type | String | Store Position Type | Page 12 |
| occupied | Occupied Status | integer | Show if current slot is occupied or free | Page 12 |
| \*arrSlots | Slot Types | Array | Stores 3 Slot Types | Page 12 |
| password | Password | string | Stores correct password | Page 12 |
| playerLimit | Number of Players | Integer | Correction of players (Min/Max) | Page 12 |
| numSlots | Number of Slots | integer | Correction of slots (Min/Max) | Page 12 |

Test Case Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Datum Physical name** | **Data Dictionary Reference** | **Input** | **Result** | **Reason** |
| pName | Page 11 | Patrick | Valid | Correct Format |
|  | Page 11 | 8 | Invalid | Integer |
| pType | Page 11 | Human | Valid | Correct Type |
|  | Page 11 | Warlock | Invalid | Incorrect Type |
| Password | Page 11 | ‘login’ | Valid | Correct Password |
|  | Page 11 | Anything else | Invalid | Incorrect Password |
| playerLimit | Page 11 | 2-6 | Valid | Within bounds |
|  | Page 11 | <2 or >6 | Invalid | Out of Bounds |
| numSlots | Page 11 | 2-20 | Valid | Within Limits |
|  | Page 11 | <2 or >20 | Invalid | Out of Bounds |

Development Log

Note: Work Breakdown Sheet attached with Submission



UML Diagram

Begin

New Round Or Exit

Next Player

Attack Player

Move Slot

Player Details

Players into Slots

Slot Assignment

Set Limit

Failed

Incorrect Limit

Player Limit

Failed

Passed

Enter Password

Password Accepted

Reload

Incorrect Password

Code

/\* Patrick Löwe - Mark Hartnett

COMP10050 - Software Engineering

Assignment 2

PASSWORD FOR TESTING = login \*/

// include files - library

#include "stdio.h"

#include "malloc.h"

#include "stdlib.h"

#include "string.h"

#include "time.h"

#include "conio.h"

// Declare variables

char fStart(); // Loads the main menu

void fReload(); //Reload function if password is wrong

void humanAttributes(int a); //function for human Attributes

void ogreAttributes(int a);//function for Ogre Attributes

void elfAttributes(int a);//function for Elf Attributes

void wizardAttributes(int a);//function for Wizard Attributes

int numSlots; // Stores the number of Slots, Max = 20 & Minimum = number of players

int playerLimit;//sets player limit to 6

// Structure for player attributes

struct attributes {

char pName[20]; // Player Name

char pType[6]; // Player Type

float lifePoints; //Health

int smart;

int strength;

int magic;

int luck;

int dexterity;

int pSlotNum; //current slot number

char pSlotType[6];// current slot type

}; // End of structure

// attributes for zones (slots)

struct zone{

char pSlotType[6]; //stores slot type

int pSlotNum; // slot number, before being assigned to a player

int occupied;//0 indicates slot is not occupied, 1 indicates slot is occupied

};

struct attributes player[6];

struct zone slots[20];

char \*arrSlots[3] = {"Ground","City","Hill"};

// main function

float main(void){

// Setting up the screen

system("color 02"); // Grey background, blue font

const char password[15] = "login"; // const will not allow the password to change. The password is "login"

char pass[8]; //the users enters a password, upto 15 characters

char h; // h is a placeholder

int position = 0; // position is the position of the password (i.e. how many characters where entered)

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - User Login - - - - - - - - - - - - - -\n\n");

printf("\n\n\t\tPassword: ");

do { // function to gather user input while also hidding printed characters with an \*

h = getch();

if( isprint(h) ) // isprint makes sure the character can be printed

{

pass[ position++ ] = h;

printf("%c", '\*'); // everytime a character is entered it is replaced or 'hidden' behind and asterisk

}

else if( h == 8 && position ) // everytime a character is deleted, it removes the entry and also the asterisk

{

pass[ position-- ] = '\0';

printf("%s", "\b \b");

}

} while( h != 13 );

if(strcmp(pass, password) != 0) // this will compare the correct password and the one entered, if incorrect do the following

{

printf("\n\n\t\tInvalid password!\n\t\tPlease Wait and Try Again! \n \n");

sleep(3);// pause program for 3 sec

system("CLS"); //clears the screen

fReload();//Loads the reload function

}

else // otherwise, if correct do this

{

printf("\n\t\tAccess Granted!\n\t\tPress enter to continue...");

getch();

system("cls"); // clear the screen

fStart(); // load the start function

}

getchar(); // if the password is invalid, press a key to continue

return 0; // For safe exit

}

// reloading main function if password fails

void fReload(void){

main(); //reload main function

}

// "main" program menu

char fStart(void){

int i=0;

// Setting up the screen

system("CLS");

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - Player Creation - - - - - - - - - - - -\n\n");

printf("\tHow many players are there: ");

scanf("%d", &playerLimit);

if(playerLimit>6) //if user input exceeds the 6 player limit set it to max

{

playerLimit = 6;//set the limit to 6

printf("\n\tMaximum Players exceeded, Setting Players to 6!\n\tPlease Wait.");

printf(".");

sleep(1);

printf(".");

sleep(1);

printf(".\n");

sleep(1);//wait 1 sec

}

else if(playerLimit<2) // prevents 1 player or less being entered as its a multiplayer game

{

playerLimit = 2;//set the limit to 2

printf("\n\tMinimum number of Players required, Setting Players to 2!\n\tPlease Wait.");

printf(".");

sleep(1);

printf(".");

sleep(1);

printf(".\n");

sleep(1);//wait 1 sec

}

for(i=1; i<=playerLimit; i++){//loop to set each characters attributes

system("CLS");

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - Player[%d] Creation - - - - - - - - - -\n", i);

printf("- - - - - - - - - - - - Types: Human - Elf - Ogre - Wizard - - - - - - - - -\n\n");

fflush(stdin);

printf("Enter Player[%d] Name: ",i);

scanf("%7s",player[i].pName); // stores up to 7 characters in player name

fflush(stdin);

printf("Enter Player[%d] Type: ", i);

scanf("%s",player[i].pType); // stores player type

if(strcmp(player[i].pType,"Human")==0){

humanAttributes(i);//when the character is a human call function to give this character attributes of a human

}

else if(strcmp(player[i].pType,"Ogre")==0){

ogreAttributes(i);//when the character is an ogre call function to give this character attributes of an ogre

}

else if(strcmp(player[i].pType,"Elf")==0){

elfAttributes(i);//when the character is an elf call function to give this character attributes of an elf

}

else if(strcmp(player[i].pType,"Wizard")==0){

wizardAttributes(i);//when the character is a wizard call function to give this character attributes of a wizard

}

}

system("CLS");//clears screen

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - Player Details - - - - - - - - - - - -\n\n");

printf("Player Info\n");

printf("\tName\t\t\tType\tHealth\tSmart\tStrength\tMagic\tLuck\tDexterity");

for(i=1; i<=playerLimit; i++) // loop through players, printing details

{

printf("\n%d\t%s\t\t\t%s\t%.2f\t%d\t%d\t\t%d\t%d\t%d",i,player[i].pName,player[i].pType,player[i].lifePoints,player[i].smart,player[i].strength,player[i].magic,player[i].luck,player[i].dexterity);

}

printf("\n\n\t\t\tPress Enter to Continue to Slot Creation...");

getch();//wait for button press

fSlots();//load slot creation function

}

//sets the random attributes for a human

void humanAttributes(int a){

// srand(time(NULL));

player[a].lifePoints = 100.00;//life points set at 100

player[a].smart=rand() % (100 +1 -1) + 1;//random numer between 1 and 100

player[a].strength=rand() % (100 +1 -1) + 1;//random numer between 1 and 100

player[a].magic=rand() % (100 +1 -1) + 1;//random numer between 1 and 100

player[a].luck=rand() % (100 +1 -1) + 1;//random numer between 1 and 100

player[a].dexterity=rand() % (100 +1 -1) + 1;//random numer between 1 and 100

}

//sets the random attributes for an ogre

void ogreAttributes(int a){

player[a].lifePoints = 100;//life points set at 100

player[a].magic=0;//magic set at 0

player[a].strength=rand() % (100 + 1 - 80) + 80;//random number between 80 and 100

player[a].dexterity=rand() % (100 + 1 - 80) + 80;//random number between 80 and 100

while((player[a].luck+player[a].smart)>50){//ensures sum of luck and smartness is <=50

player[a].luck=rand() % (50 + 1 - 0) + 0;

player[a].smart=rand() % (20 + 1 - 0) + 0;

}

}

//sets the random attributes for an elf

void elfAttributes(int a){

player[a].lifePoints = 100;//life points set at 100

player[a].luck=rand() % (100 + 1 - 60) + 60;//random number between 60 and 100

player[a].smart=rand() % (100 + 1 - 70) + 70;//random number between 70 and 100

player[a].strength=rand() % (50 + 1 - 1) + 1;//random number between 1 and 50

player[a].magic=rand() % (79 + 1 - 51) + 51;//random number between 79 and 51

player[a].dexterity=rand() % (100 +1 -1) + 1;//random number between 1 and 100

}

//sets the random attributes for a wizard

void wizardAttributes(int a){

player[a].lifePoints = 100;//life points set at 100

player[a].luck=rand() % (100 + 1 - 50) + 50;//random number between 50 and 100

player[a].smart=rand() % (100 + 1 - 90) + 90;//random number between 90 and 100

player[a].strength=rand() % (20 + 1 - 1) + 1;//random number between 1 and 20

player[a].magic=rand() % (100 + 1 - 80) + 80;//random number between 80 and 100

player[a].dexterity=rand() % (100 +1 -1) + 1;//random number between 1 and 100

}

// Randomly generate slots

int fSlots(){

system("CLS");//clears screen

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - Slots Creation - - - - - - - - - - - -\n\n");

printf("How Many Slots would you like (MAX 20): ");

scanf("%d", &numSlots); //stores the number of slots

if(numSlots>20)

{

numSlots = 20;//set the limit to 20

printf("\n\tMaximum Slots exceeded, Setting Slots to 20!\n\tPlease Wait");

printf(".");

sleep(1);

printf(".");

sleep(1);

printf(".\n");

sleep(1);

}

else if(numSlots<playerLimit) // if the number of slots is less than players i.e no room

{

numSlots = playerLimit;//set the limit to player limit

printf("\n\tMinimum number of slots must be equal to, or greater than players, Setting Slots to %d!\n\tPlease Wait", playerLimit);

printf(".");

sleep(1);

printf(".");

sleep(1);

printf(".\n");

sleep(1);

}

int j=0;

srand(time(NULL));//randomizer

for(j=1; j<=numSlots; j++){

int r = rand() % 100; //random between 0 and 100

int t = r % 3; // bring the value down to between 0 and 3

if(t == 0){

strcpy(slots[j].pSlotType, arrSlots[0]);// assign the 1st element(defined above) to slot type

}

else if(t == 1){

strcpy(slots[j].pSlotType, arrSlots[1]);// assign the 2nd element(defined above) to slot type

}

else if(t == 2){

strcpy(slots[j].pSlotType, arrSlots[2]); // assign the 3rd element(defined above) to slot type

}

slots[j].pSlotNum = j;

printf("\tSlot[%d]: %s\n", slots[j].pSlotNum, slots[j].pSlotType);

slots[j].occupied=0;//all slots set as unnocupied to start

slots[0].occupied=1;// Slot 0 is not used, for simplicity

}

printf("\n\t\t\tPress Enter to Randomly Assign Players to Slots...\n");

getch();

fAssign();

} // End of Write

// Assign players to slots

int fAssign(){

system("CLS");

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - Player Assigning - - - - - - - - - - -\n\n");

printf("\t\t\tRandomly Assigning Players to Slots...\n");

printf("\t\t\tPlease wait.");

sleep(1);

printf(".");

sleep(1);

printf(".\n");

sleep(1);

int i,j;

srand(time(NULL));//randomizer

for(i=1; i<=playerLimit; i++){

int r = rand() % 100; //random between 0 and 100

int t = r % numSlots+1; // bring the value down to between 0 and number of slots specified by player

if (slots[t].occupied == 0){

player[i].pSlotNum = t;

slots[t].occupied = 1; // assign the slot num to the correct player

}

else if(slots[t].occupied == 1 && ++t<=playerLimit){

player[i].pSlotNum = t; //assign the player to the new slot

slots[t].occupied = 1;//set new slot to occupied

}

strcpy(player[i].pSlotType, slots[t].pSlotType); //assign the string slot type to players slot type;

slots[t].occupied = 1;//indicates slot is occupied

printf("\t\t%s has been assigned to Slot[%d]: %s\n", player[i].pName, slots[t].pSlotNum, slots[t].pSlotType); //Prints Plyaer name, SLot number from previous function, and slot type

}

getch();

fOption();

}

// Options for current player to Attack or Move Slot

int fOption(){

system("CLS");

int iSwap, iSwap2,i,j,c,prevSlotNum;

for(i=1; i<=playerLimit; i++){

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - Player %d Options - - - - - - - - - - -\n\n",i);

int b= player[i].pSlotNum; // hold current player position, before moving backward

b = --b;//moves backwards 2 saces to accurately represent 1 space in the interface

int f= player[i].pSlotNum; // hold current player position, before moving forward

f = ++f; // moves player forward 1 space

printf("\t\tPlease Select an Option for Player[%d]: %s\n", i, player[i].pName);

printf("\nCurrent Slot: %d %s\n", player[i].pSlotNum, player[i].pSlotType);

if(slots[f].occupied == 0 && f<=numSlots){ // if next slot is available

printf("Next Slot: %d %s\n", slots[f].pSlotNum, slots[f].pSlotType);

}

if(slots[b].occupied == 0 && b>0){ //if previous slot is available and not slot 0 (not used)

printf("Prev Slot: %d %s\n", slots[b].pSlotNum, slots[b].pSlotType);

}

if(slots[f].occupied == 0 && f<=numSlots){ //moving forward slot is free

printf("\t 1. Move Forward\n");

}

if(slots[b].occupied == 0 && b>0){ //moving back slot is free

printf("\t 2. Move Backwards\n");

}

if(slots[b].occupied == 1 && slots[f].occupied == 1){

printf("\t All slots occupied, unable to move.\n");

}

printf("\t 3. Attack Nearest Player\n");

for(j=1; j<=playerLimit;j++){

printf("%s (%s, %.2f)\n", player[j].pName, player[j].pType, player[j].lifePoints);

}

printf("\nOption: ");

scanf("%d", &iSwap);

if(iSwap == 1){

moveForward(i); //move current player forwards

}

else if(iSwap == 2){

moveBack(i);//move current player backwards

}

else if(iSwap ==3){

fAttack(i);//attack the nearest player

}

else{

printf("Invalid Entry!");

}

system("CLS");

}

fContinue();

}

//moving current player forward 1 slot if not occuppied

int moveForward(int i){

printf("Moving Player %s Forwards.", player[i].pName);

sleep(1);

printf(".");

sleep(1);

printf(".\n");

int f= player[i].pSlotNum; //f is the current slot number, will be incremented

slots[f].occupied=0;//no longer occupied

f = f+1;//incrementing player slot didn't work, reassigned to f after increasing

player[i].pSlotNum = f;//player i's slot position changes to the slot behind

strcpy(player[i].pSlotType, slots[f].pSlotType);//player i's slot position changes to the slot behind

slots[f].occupied=1;//becomes occupied

printf("\nPlayer %s has is now in Slot[%d]: %s", player[i].pName, player[i].pSlotNum,player[i].pSlotType);

CapabilitiesChange(i);//load function to update players capabilities based on new slot

}

//Moving current player back 1 slot if not occuppied

int moveBack(int i){

printf("Moving Player %s Backwards.", player[i].pName);

sleep(1);

printf(".");

sleep(1);

printf(".");

int b= player[i].pSlotNum;//b is the current players slot

slots[b].occupied=0;//no longer occupied

b = b-1;// decremented b (--b) didn't work in testing, so reassigned to b after decreasing

player[i].pSlotNum = b;//player i's slot position changes to the slot behind

strcpy(player[i].pSlotType, slots[b].pSlotType);//player i's slot position changes to the slot behind

slots[b].occupied=1;//becomes occupied

printf("\nPlayer %s has is now in Slot[%d]: %s", player[i].pName, player[i].pSlotNum,player[i].pSlotType);

CapabilitiesChange(i);

}

//function changes the capabilities of a player that moves

int CapabilitiesChange(int i){

if(strcmp(player[i].pSlotType, "Hill")==0){

if(player[i].dexterity<50){

player[i].strength -= 10;

printf("\n%s has lost 10 Strength from moving Slot!", player[i].pName);

}

else if(player[i].dexterity>=60){

player[i].strength += 10;

printf("\n%s has gained 10 Strength from moving Slot!\nPress Enter to Continue", player[i].pName);

}

getch();

}

else if(strcmp(player[i].pSlotType, "City")==0){

if(player[i].smart>60){

player[i].magic += 10;

printf("\n%s has gain 10 Magic from moving Slot!", player[i].pName);

}

else if(player[i].smart<=50){

player[i].dexterity -= 10;

printf("\n%s has lost 10 Dexterity from moving Slot!\nPress Enter to Continue", player[i].pName);

}

getch();

}

else {

printf("\nNo changes to Player Attributes.\nPress Enter to Continue");

getch();

}

}

//Function to locate the nearest player

int nearestPlayer(int i){

int a, b,instruct;

a=i+1; //a is the next slot

b=i-1;// b is the previous slot

while(a<numSlots || b>=0){//while loop finds the nearest player to attack

if(slots[a].occupied==1 && slots[b].occupied==1 && b!=0){

printf("Would you prefer to attack 1.%s or 2.%s:", player[a].pName, player[b].pName);// when two players are the same distance from the player ask who they want to attack

scanf("%d", &instruct);

if(instruct==1){

printf("Attacking %s\n", player[a].pName);

return a;

}

else if(instruct==2){

printf("Attacking %s\n", player[b].pName);

return b;

}

else{

printf("Invalid Entry\n");

break;

}

}

else if(slots[a].occupied==1 && slots[b].occupied==0){//closest player is in a slot infront

printf("Attacking Player %s ahead of you!\n", player[a].pName);

return a;

}

else if(slots[a].occupied==0 && slots[b].occupied==1){//closest player is in a slot behind

printf("Attacking Player %s behind you!\n", player[b].pName);

return b;

}

a++; // increase to next slot until player is found

b--;// decrease to previous slot until player is found

}

}

// Function when Attacking a player

int fAttack(int i){//reduces life points when attack is preformed

int n = nearestPlayer(i); // n is the player being attacked

float damage;

if(player[n].strength<=70){

printf("%s has %.2f health before being attacking\n", player[n].pName, player[n].lifePoints);

sleep(1);

damage = (0.5)\*player[n].strength;

player[n].lifePoints = player[n].lifePoints - (0.5)\*player[n].strength;

printf("Critical Strike - %.2f\n", damage);

sleep(1);

printf("%s has %.2f health after being attacking\n", player[n].pName, player[n].lifePoints);

}

else if(player[n].strength>70){

printf("%s has %.2f health before being attacking\n", player[n].pName, player[n].lifePoints);//display value of attacked player

damage = (0.3)\*player[n].strength;

sleep(1);

player[n].lifePoints = player[n].lifePoints - (0.3)\*player[n].strength;

printf("Normal Strike - %.2f\n", damage);//display value of attack

sleep(1);

printf("%s has %.2f health after being attacking\n", player[n].pName, player[n].lifePoints);//Display attacked players health

}

printf("\n\t\tPress Enter to Continue...");

getch();//wait for button press

}

int fContinue(){ //Function for reloading a second round

int iOpt;

printf("- - - - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - Continue Next Round?- - - - - - - - - - -\n");

printf("Press 1 to continue another round!\n");

scanf("%d", &iOpt);

if(iOpt == 1){

fOption();//reload the fOptions menu for another round

}

else{

printf("\n\t\t\tInvalid Entry! \n\t\t\tExiting Program!");

sleep(3); //Sleep or "wait" for 3 sec

close();//Load the close function

}

}

//Function to exit the program

int close(){

system("CLS");

printf("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - Witches & Wizards - - - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - Thank You for Playing - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");

printf("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");

exit(0); //closes the program

}

References

#### Cboard.cprogramming.com

#### CProgramming.language-tutorial.com

#### StackOverflow.com

#### Tutorialspoint.com

Declaration of Authorship

7 – March – 2017

Patrick Lowe Mark Hartnett