1.1 Problem

Name: ADGP 105 Retro Game Assessment
Problem Statement: Students create a top-down shooter retro style 2D game, demonstrating the ability to design, plan and build a simple game, create and code “bug and error free” program and demonstrate an understanding of C++ programming constructs, classes, functions and data structures.
Problem Specification: Make a text based game that contains a 4x4 2D Grid, where the player uses hints to navigate across the grid. I must show that I have knowledge in what we have covered throughout the course. The program must grab and keep track of the player’s location as he navigates across the map. My program must also understand how to read from a file and grab information from the file.

1.2 Input Information.

1.2.1 Input Streams:

Name: char uInput
Description: This input was used to obtain the user’s choice for beginning the game. Its purpose was to give the user a choice of starting or exiting the game.
Format: N/A
Size: 1 Line
Sample: cin >> uInput

Name: char uInput
Description: This input was used to obtain the user's choice of movement. It identified if the player wanted to move North,South, East, West. The purpose of this was to give the player a choice of movement on the grid.
Format: N/A
Size: 1 Line
Sample: cin >> uInput

1.2.2 Input Items:
char uInput(1):
Description: This was used to ask to user “y/n” which was prompted at the beginning of the game. It takes in if the user wants to continue on to the game or exit out of the game.
Type: Real
Range of acceptables values: Only accepts y/Y or n/N. All other values are invalid.

char uInput(2):
Description: This was used to ask the user which direction he wanted to go in. The directional keys were W for north, A for West, S for South, D for East.
Type: Real

Range of acceptables values: Only accepts W,A,S,D keys and all other values will not work.

1.3 Output Information

Output Streams:

Name: Map.txt (Text File of Information)
Description: My program reads from the Map.txt file, which contains information on the game as well as the grid that the player will be playing on. It keeps the player informed about the game and his/her position on the map.
Format: The data is displayed in chronological order and it is displayed exactly how it is displayed on the text file.
Size: The total size of the stream is 7 lines.
Sample:

Name: Movement
Description: Once the user begins the game, the player is prompted to question that asks for the direction that the player wants to move in.
Format: The player is prompted with a simple question asking which direction he would like to go in. That question is then followed by four option the player could choose from. Those options consist of North, South , East, West.
Size: This consists of six lines of code.
Sample: cout <<"Where do you want to move?\n";

Output items:
Map.txt
Movement:

1.4 User Interface Information

1.4.1 Description

At the beginning of the game, the user will be prompted to the information stored on the Map.txt file.The Map.txt files consists of the grid and information on the game itself. Followed by the information stored on that file, the user would be asked if he/she would like to begin playing the game. If the user chooses no, then the game will be closed. When the user chooses to start the game, he/she will be asked, which direction they would like to move in. All information on this page will be displayed in the English language.

II. Design Documentation

II.1 System Architecture Description

II. 2 Information on the Objects

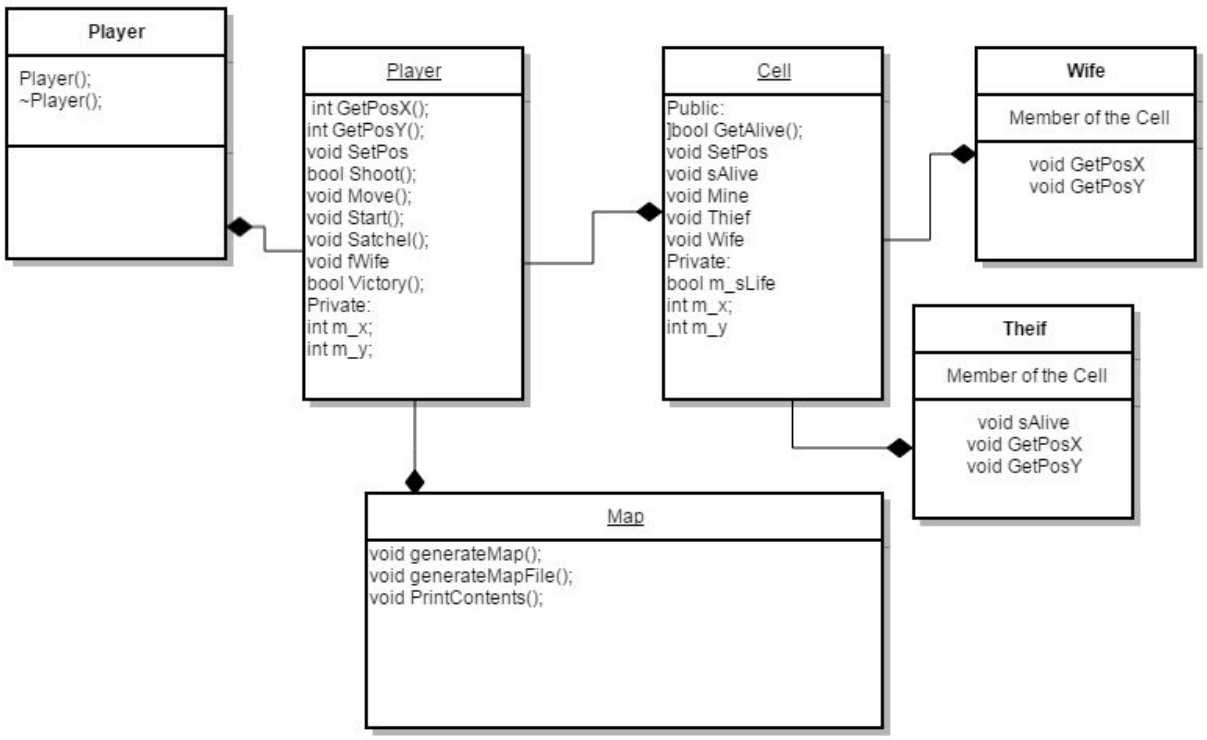
Class Information
Name: Map
Description: This is used to store the constructor and destructor for the class map. It is also used to store data and other information of the map.

Name: Cell
Description:

Name: Player
Description:

Class Attributes
Map:
Name: void generateMap();
Name: void generateMapfile();
Name: void PrintContents();
Cell:
Name: GetPosX();
Name: GetPosY();
Name: GetAlive();
Name: SetPos(int a_x)
Name: sAlive(bool s_a)
Name: Mine(int a_x, int a_y)
Name: Thief(Player &a_rfplayer, int a_x, int a_y)
Name: Wife(Player &a_rfplayer, int a_x, int a_y)
Player:
Name: GetPosX();
Name: GetPosY();
Name: ammo
Name: SetPos(int a_x, int a_y)
Name: Shoot();
Name: Move();
Name: Satchel();
Name: fWife(int &a_wX, int &a_wY)
Name: Victory();

II.4 Design Diagrams



III. Implementation Documentation

All code was implemented and testing within Visual Studio's 2015.

III.1 Program Code

Header Code:

Player.h

```
#ifndef PLAYER_H
#define PLAYER_H
#include <iostream>
#include <string>
#include "Map.h"
using namespace std;
//class resembling player
class Player
{
	//public data set for Player
public:
	//Constructor for Player
```

```
        Player();
        //Destructor for Player
        ~Player();

        int GetPosX();  //position x
        int GetPosY();  //position y

        void SetPos(int a_px,int a_py); //sets position  for player
        bool Shoot(); //enables player to shoot
        void Move(); //enables movement for the player
        void Start(); //Function that starts the game for the player
        void Satchel(); //enables the player to have an inventory.
        void fWife(int &a_rfX, int &a_rfY);
        bool Victory(); //Decides if the player has successfully obtained the objective and brought
it back safely.

        //private data set for Player
private:
        //private variable for m_x
        int m_x;
        //private variable for m_y
        int m_y;

};

#endif PLAYER_H
```

CGrid.h

```
#ifndef CGRID_H
#define CGRID_H
#include "Player.h"

using namespace std;
//creates a class for Cell
class Cell
{
//public class for player
public:
        //Constructor for Cell
        Cell();
        //Destructor for Cell
        ~Cell();
```

```
        //Identifies current position of player
        Cell dCell(int a_px,int a_py);

        int GetPosX();
        int GetPosY();

        bool GetAlive();
        //Sets Position
        void SetPos(int a_px, int a_py);
        //Checks to see if player is alive
        void sAlive(bool a_l);
        //Checks to see the position of the landmine
        void Mine(int a_x, int a_y);
        //Checks to see position of theif and player to indicate if player survives
        void Thief(Player &a_rfplayer, int a_x, int a_y);
        //Checks to see position of Wife and indicates if player picked up wife and returned back
safely.
        void Wife(Player &a_rfplayer, int a_x, int a_y);

//private class for player
private:
        //Creates a boolean to see if Player alive
        bool m_sLife;
        //Private variable in memory for x
        int m_x;
        //Private variable in memory for y
        int m_y;

};
#endif CGRID_H
```

Map.h

```
#ifndef MAP_H
#define MAP_H
#include <iostream>
#include <cstdlib>
using namespace std;
//class representing map
class Map
{
```

```
//public data set for map
public:
	//Constructor for Map
	Map();
	//Destructor for Map
	~Map();

	void generateMap();
	void generateMapfile();
	void PrintContents();
//private data set for map
};



#endif MAP_H
```

Source Code:
CGrid.cpp

```
#include "CGrid.h"
#include "Player.h"
//Constructor for Cell
Cell::Cell()
{
	m_x = 0;
	m_y = 0;

	m_sLife = true;
}
//Destructor for Cell
Cell::~Cell()
{

}
//Structure for storing Cells x and y
Cell Cell::dCell(int a_px, int a_py)
{
Cell cCell;
```

```
cCell.m_x = a_px;
cCell.m_y = a_py;

return cCell;
}
//Obtains the private variable m_x from the class Player
int Cell::GetPosX()
{
	return m_x;
}
//Obtains the private variable m_y from the class Player
int Cell::GetPosY()
{
	return m_y;
}
void Cell::SetPos(int a_px, int a_py)
{
	m_x = a_px;

	m_y = a_py;
}
//Cell::GetAlive is used to get the private variable m_sLife. So it can be used outside of the
class.
bool Cell::GetAlive()
{
	//Returns the private variable m_sLife
	return m_sLife;
}
void Cell::sAlive(bool l)
{
	//makes the private variable m_sLife = 1
	m_sLife = l;
}
void Cell::Mine(int a_px, int a_py)
{
	Cell Mine;

	Mine.SetPos(1, 0);
	//Sets the positions of warnings for the explosives.
	if ((a_px == 0 && a_py == 0) ||
		(a_px == 2 && a_py == 0) ||
		(a_px == 1 && a_py == 1) ||
		(a_px == 3 && a_py == 0) ||
```

```
		(a_px == 2 && a_py == 1) ||
		(a_px == 3 && a_py == 2) ||
		(a_px == 0 && a_py == 2) ||
		(a_px == 1 && a_py == 2) ||
		(a_px == 1 && a_py == 3))
	{
		//Indicates that there is an explosive nearby
		cout << "There is an explosive nearby\n";
	}
	//Creates an if statement stating that there is a landmine at this location.
	if ((a_px == 1 && a_py == 0) &&
		(Mine.GetPosX() == 1 && Mine.GetPosY() == 0))
	{
		system("cls");
		cout << "KABOOOOOOOMMMM\n";
		system("pause");
		exit(1);
	}
	//Creates an if statement stating that there is a landmine at this location
	Mine.SetPos(3, 1);
	if ((a_px == 3 && a_py == 1) &&
		(Mine.GetPosX() == 3 && Mine.GetPosY() == 1))
	{
		system("cls");
		cout << "KABOOOOOOOMMMM\n";
		system("pause");
		exit(1);
	}
	//Creates an if statement stating that there is a landmine at this location
	Mine.SetPos(0, 3);

	if ((a_px == 0 && a_py == 3) &&
		(Mine.GetPosX() == 0 && Mine.GetPosY() == 3))
	{
		system("cls");
		cout << "KABOOOOOOOMMMM\n";
		system("pause");
		exit(1);
	}
}
//Indicates Thief's position and compares it to the position of the player.
void Cell::Thief(Player &a_rfplayer, int a_px, int a_py)
{
```

```
	//Creates a cell called Thief so it can be used throughout the program.
	Cell Theif;
	//Gives position to the Thief
	Thief.SetPos(2, 2);
	//Records position of player so it can indicate one of the senses.
	if ((a_px == 1 && a_py == 2) ||
		(a_px == 2 && a_py == 1) ||
		(a_px == 2 && a_py == 3) ||
		(a_px == 3 && a_py == 2))
	{
		cout <<"You smell a scent of perfume\n";
	}
	Thief.sAlive(a_rfplayer.Shoot());
	//Checks position of the player
	if ((a_px == 2 && a_py == 2) &&
		(Thief.GetPosX() == 2 && Thief.GetPosY() == 2) && (Thief.GetAlive() == true))
	{
		//Below happens if the position of the player is correct.
		system("cls");
		cout <<"You got backstabbed by the Thief\n";
		system("pause");
		exit(1);
	}
	//Checks to see if the thief is alive.
	if ((a_px == 2 && a_py == 2) &&
			(Thief.GetPosX() == 2 && Thief.GetPosY() == 2) && (Thief.GetAlive() ==
false))
		{
			cout <<"You've located where the thief once stood. You found your Wife's
belongings.\n";
		}

}
//Checks the position of the player's Wife and the position of the Player
void Cell::Wife(Player &a_rfPlayer, int a_px, int a_py)
{
	Cell Wife;
	Wife.SetPos(3, 3);
	//Checks position of the player.
	if ((a_px == 2 && a_py == 3) ||
		(a_px == 3 && a_py == 2))
	{
		cout << "There is a smell of Perfume\n";
```

```
	}
	if ((a_px == 3 && a_py == 3) &&
		(Wife.GetPosX() == 3 && Wife.GetPosY() == 3))
	{
		a_rfPlayer.fWife(a_px, a_py);
	}
}
```

Map.cpp

```
#include "Map.h"
#include "Player.h"
#include "CGrid.h"
#include <string>
#include <fstream>
using namespace std;
//Constructor for Map function
Map::Map()
{


}
//Destructor for Map function
Map::~Map()
{

}
//Creates grid for player to roam and explore the 4x4 grid.
void Map::generateMap()
{
	Cell iCell;
	//Sets Temp variable to equal 4
	int Temp = 4;
	//Sets size Temp times Temp which will equal 16.
	int Size = Temp * Temp;
	//Creates a pointer for Map
	Cell *Map;
	//Sets the map pointer to a new allocated array of memory
	Map = new Cell[Size];
	//Makes temp2 the square root of the size variable.
	int temp2 = sqrt(Size);
	//Sets y to the value of 0d
```

```
int y = 0;

for (int i = 0; i < Size; ++i)
{
	int x = i % temp2;

	Map[i] = iCell.dCell(y, x);

	if (x == temp2 - 1)
	{
		y++;
	}
}
//Creates a string called f. The f represents File.
string f;

ofstream File;

File.open("Map.txt", ios_base::out);

if (File.fail())
{
	cout << "File Failed to Open\n";
}
else
{
	for (int i = 0; i < Size; ++i)
	{
		//Gives the map its values
		int x = i % temp2;

		File << Map[i].GetPosY() << "," << Map[i].GetPosX() << "";

		if (x == temp2 - 1)
		{
			File << endl;
		}
	}
	//closes the file
	File.close();
}
//removes the allocated memory
delete[] Map;
```

```
}
//Generates the map file from Map.txt and displays it onto the screen
void Map::generateMapfile()
{
	//Creates a string called f representing File.
	string f;

	ifstream File;

	File.open("Map.txt", ios_base::in);
	//If file fails to open then it will display the text represented in the cout.
	if (File.fail())
	{
		//Displays on to screen if file failed to open correctly.
		cout << "File Failed to Open\n";
	}
	else
	{
		while (getline(File, f))
		{
			cout << f;
			cout << endl;
		}
	}
	cout << endl;
	//Closes the File if it failed to open/
	File.close();
}
//prints game contents that prompts player to chose the direction he wants to go in.
void Map::PrintContents()
{
	Player pPlayer;

	system("cls");
	//Generates map file above 2
	generateMapfile();
	//Prompts the user chose a location to move in.
	cout <<"Where do you want to move?\n";
	cout <<"Move North: W\n";
	cout <<"Move South: S\n";
	cout <<"Move East: D\n";
	cout <<"Move West: A\n";
```

```
	cout <<"Fire Bullet: Q\n\n";

	pPlayer.Satchel();
}
```

Main.cpp

```
#include <iostream>
#include "Map.h"
#include "Player.h"
using namespace std;

int main()
{
	//Forms a boolean that is true
	bool isDone = true;
	Player pPlayer;
	Map mMap;

	mMap.generateMapfile();
	//Asks the user if he/she wishes to begin playing the game.
	pPlayer.Start();
	//Displays content such as inventory and the players current location on the map
	mMap.PrintContents();
	cout <<"Current Position: " << pPlayer.GetPosX() << ',' << pPlayer.GetPosY();
	//while the bool isDone is true. The player is victorious
	while (isDone == true)
	{
		//Keeps track of player movement.
		pPlayer.Move();
		//Checks to see if player has completed objective and will not reedeem player
victorious if objective has not been completed.
		isDone = pPlayer.Victory();
	}
	//pauses the system.
	system("Pause");
	return 0;
}
```

Player.cpp

```
#include "Player.h"
#include "Map.h"
```

```
#include "CGrid.h"
//Constructor for player
Player::Player()
{
	m_x = 0;
	m_y = 0;
}
//Destructor for player
Player::~Player()
{

}
//Creates a struct for SuitCase and it contains whats inside
struct SuitCase
{
	//creates a string name for your suitcase
	string iName;
	//Displays the amount of items within the suitcase
	int iAmount;
};
//Contains an array to contains a suitcase that contains a Pistol with one bullet and it will contain another item in the future.
SuitCase iCase[2] = { {"Pistol", 1},{"Wife", 0} };
//Gets position of player at x
int Player::GetPosX()
{
	//returns the m_x value
	return m_x;
}
//Gets position of player at y
int Player::GetPosY()
{
	//returns the m_y value
	return m_y;
}
//Sets the position for the player
void Player::SetPos(int a_px, int a_py)
{
	m_x = a_px;

	m_y = a_py;
}
//Function that enables the player to move around the map
```

```
void Player::Move()
{
	Map mMap;
	Cell cCell;
	Player pPlayer;
	//keeps track of user input
	char uInput;
	cout << endl;

	cin >> uInput;
	//switch statement for the choices the character makes to move.
	switch (uInput)
	{
	//Actions that would happen if player where to move with w. The program will notify the
player  if he cant go in that direction.
	case 'w':
	{
		SetPos(m_x, m_y - 1);

		mMap.PrintContents();

		if (m_y < 0)
		{
			SetPos(m_x, m_y = 0);
			cout << "Cannot go in that direction\n";
		}
		//Calls a function for a Landmine
		cCell.Mine(GetPosY(), GetPosX());
		//Calls a function for the Theif
		cCell.Thief(pPlayer, GetPosY(), GetPosX());
		//Calls a function for players Wife
		cCell.Wife(pPlayer, GetPosY(), GetPosX());
		//Displays Current Position of player
		cout << "Current Position:" << GetPosY() << ',' << GetPosX() << endl;
		break;
	}
	//Actions  that would happen if player chose to hit s. The program will notify the player if
he cant go in that direction.
	case 's':
	{
		SetPos(m_x, m_y + 1);

		mMap.PrintContents();
```

```
		if (m_y > 3)
		{
			SetPos(m_x, m_y = 3);
			cout << "Cannot go in that direction\n";
		}
		//Calls a function for a Landmine
		cCell.Mine(GetPosY(), GetPosX());
		//Calls a function for the Thief
		cCell.Thief(pPlayer, GetPosY(), GetPosX());
		//Calls a function for the players Wife
		cCell.Wife(pPlayer, GetPosY(), GetPosX());
		//Displays Current Position of player
		cout <<"Current Position:" << GetPosY() << ',' << GetPosX() << endl;
		break;
	}
	//Actions that would happen if player chose to hit a. The program will notify the player if he cant go in that direction.
	case 'a':
	{
		SetPos(m_x - 1, m_y);
		mMap.PrintContents();
		if (m_x < 0)
		{
			SetPos(m_x = 0, m_y);
			cout <<"Cannot go in that direction\n";
		}

		cCell.Mine(GetPosY(), GetPosX());
		cCell.Thief(pPlayer, GetPosY(), GetPosX());
		cCell.Wife(pPlayer, GetPosY(), GetPosX());
		cout <<"Current Position:" << GetPosY() << ',' << GetPosX() << endl;
		break;
	}
	//Actions that would happen if the player chose to hit d. The program will notify the player if he cant go in that direction.
	case 'd':
	{
		SetPos(m_x + 1, m_y);
		mMap.PrintContents();

		if (m_x > 3)
		{
```

```
                        SetPos(m_x = 3, m_y);
                        cout << "Cannot go in that direction\n";
                }
                cCell.Mine(GetPosY(), GetPosX());
                cCell.Thief(pPlayer, GetPosY(), GetPosX());
                cCell.Wife(pPlayer, GetPosY(), GetPosX());
                //Displays players current position to the screen.
                cout << "Current Position:" << GetPosY() << ',' << GetPosX() << endl;
                break;
        }
        //Actions that would happen if the player chose to hit Q. This will be the action to attack
the thief.
        case 'q':
        {
                Shoot();
                break;

        }
        default:
                cout << "Invalid Choice" << endl;
                break;
        };
}
//a boolean that displays if the thief is alive or not.
bool a_Theif = true;
//Functions that idenitfies if the player managed to kill the thief.
bool Player::Shoot()
{
        Cell Theif;
        Map mMap;
        //Checks the players position and checks to see if the player has an item in his inventory.
        if (((((GetPosX() == 1 && GetPosY() == 2) &&
                (Theif.GetAlive() == true) && (iCase[0].iAmount == 1) ||
                (GetPosX() == 3 && GetPosY() == 2) &&
                (Theif.GetAlive() == true) && (iCase[0].iAmount == 1) ||
                (GetPosX() == 2 && GetPosY() == 1) &&
                (Theif.GetAlive() == true) && (iCase[0].iAmount == 1) ||
                (GetPosX() == 2 && GetPosY() == 3) &&
                (Theif.GetAlive() == true) && (iCase[0].iAmount == 1)))))
        {
                //Theif alive equals false if what above is true
                a_Theif = false;
                //Shows that the player used his item and now its value is set to 0 (empty)
```

```
                iCase[0].iAmount--;

                mMap.PrintContents();

                //Indication of killing the thief.
                cout << "You shot the thief.\n";
        }
        return a_Theif;

}
//Function that displays player inventory
void Player::Satchel()
{
        cout << "Items on Satchel:\n";

        for (int i = 0; i < 2; ++i)
        {
                cout << iCase[i].iName <<" x "<< iCase[1].iAmount << endl;
        }
}
//prompts the user to the start screen and informs the user if he wants to continue or not.
void Player::Start()
{
        Map mMap;
        char uInput;
        cout << "Welcome to Retro Madness\n";
        cout << "Would you like to continue? y/n";
        cin >> uInput;

        //Checks to see if user inputed Y or y. Any other variable will be invalid.
        if (uInput == 'Y' || uInput == 'y')
        {
                //Clears the system after the users decision is made
                system("cls");

        }
        //Checks to see if user inputed N or n. Any other variable will be invalid.
        else if (uInput == 'N' || uInput == 'n')
        {
                system("cls");
                //Displays "You have exited the game" if user chooses N or n.
                cout << "You have exited the game\n";
                exit(1);
```

```
	}
	else
	{
		//Displays Invalid choice if anything other than Y,y,N,n was inputted by the user
		cout <<"Invalid Choice game will begin anyways.\n";
		system("pause");
		system("cls");
	//ISSUE: Couldnt really figure out a way to make it to where it says invalid entry and then it restates the question. Its not major, but
	//I found a way around it.

	}
}
//Function that identifies the location of the player's wife and then notifies the player that he has found his wife.
void Player::fWife(int &a_rfX, int &a_rfY)
{
	Map mMap;

	if ((a_rfX == 3 && a_rfY == 3) && (iCase[1].iAmount == 0))
	{
		iCase[1].iAmount++;
		mMap.PrintContents();
		cout << "You've located your Wife\n";
	}
}
//A boolean statement that identifies if the player made it back to position 0,0 with his wife.
bool Player::Victory()
{
	//Checks to see if player has obtained his Wife, checks to see if it is true, if not dVictory will equal false
	bool dVictory = true;
	if (((GetPosX() == 0 && GetPosY() == 0) &&
		iCase[1].iAmount == 1))
	{
		cout << "You brought your wife back to safety\n";

		dVictory = false;
	}
	return dVictory;
}
```

IV. Verification and Validation Documentation

IV.1 Test Plan
Check the status of:

The game appearing on the screen properly.

Each option on the screen guides you to its destined location.

All switch statements are correctly formatted and displayed correctly.

Character Movement.

IV.2 Test Results

Main menu appearing on the screen properly	Success
Each option on the screen guides you to destined location	Success
All switch statements are correctly formatted and displayed correctly	Success
Character Movement	Success

IV.3 Operation Directions

1. Visit Github.com and search for the user dwilsonaieyr1
2. Go to the user's "test" repository
3. Find the folder "ADGP 105 Assessment"
4. Open the folder and run the ADPG 105 executable.