ADGP 105 Year 1 Assessment 2
Barlow,Brock ID# 1113 Submitted:10/20/2015

Description of the Problem:
Name: Moving in Grid
Problem Statement: The player needs to move in the grid to play the game.
Problem Specification: The code would not compute with inputs. The code would compile, but the player could not move. Using a function to move the player, I had no parameters being passed. This problem was fixed by passing a parameter into the function.

Name: Dynamic Memory Allocation
Problem Statement: This assessment requires the use of dynamic memory allocation
Problem Specification: Do to the design of my code, I had a hard time trying to implement this into my code. Doing research on dynamic memory allocation, I was able to implement it.

Name: Reading from a file
Problem Statement: This assessment requires the use of reading from a file
Problem Specification: Do to the design of my code, I had a hard time trying to implement this into my code. Inputing an extra case trigger in my switch statement, the code will now read from a file.

Input Streams:
Name: cin >> userInput
Description: Used to trigger the void movePlayer function switch statements
Format: The data variable is of type char
Size: Will take in one char input
Sample: userInput is ‘s’. Will move the player down in the grid. Will then print to the screen: “Your location now is: “ and will display new location

Name: cin >> shootInput
Description: Used to trigger the void checkIfICanShoot function
Format: The data variable is of type char
Size: Will take in one char input
Sample: When the player is one room away from the wumpus, the game will ask “Do you want to shoot the Wumpus? (y or n)”. If the player choose ‘y’ the player will kill the wumpus. If ‘n’, the game will make a joke and will allow the player to keep playing.

Input Items:
Description: Input is used to move around the grid, used to shoot the wumpus, used not to shoot the wumpus, quit the game and check position
Type: Input data type is char
Range of acceptable values: ‘w’, ‘a’, ‘s’, ‘d’, ‘q’, ‘c’, ‘y’, ‘n’

Output Streams:

<u>Name:</u> cout << "---------Welcome to Wumpus Shooter!---------",

```
cout << "-------------------------------------------",
cout << "Your mission is to find and kill the Wumpus."
cout << "If you kill it, you win."
cout << "If the Wumpus finds you, it will eat you."
cout << "If you get eaten, its game over."
cout << "Good luck!"
cout << "-------------------------------------------"
cout << "Game controls:"
cout << "--------------"
cout << "To move up, type: <----> w"
cout << "To move down, type: <--> s"
cout << "To move right, type: <-> d"
cout << "To move left, type: <--> a"
cout << "To check last location, type: <-> c"
cout << "To quit, type: <-------> q"
cout << "-------------------------------------------"
cout << "Your starting location is: 0,0."
cout << "-------------------------------------------"
cout << endl;
cout << endl;
cout << endl;
cout << "You entered the Wumpus room!" << endl;
cout << "You fool! You have gotten yourself eaten!" << endl;
cout << endl;
cout << "Game Over." << endl;
cout << endl;
cout << endl;
cout << "You have fallen down a pit and died!" << endl;
cout << endl;
cout << "Game Over." << endl;
cout << endl;
cout << endl;
cout << "You feel a breeze of cold air." << endl;
cout << "Becareful. There is a pit near by." << endl;
cout << endl;
cout << endl;
cout << "You smell a horrible smell!" << endl;
cout << "The Wumpus is near, ready yourself!" << endl;
cout << endl;
cout << endl; //new line
cout << "You fired at the Wumpus' location." << endl;
```

```
cout << "You hit the Wumpus!" << endl;
cout << "You did it! You won!" << endl;
cout << endl;
cout << "Game Over." << endl;
cout << endl;
cout << endl;
cout << "What? Why would you not take this chance?" << endl;
cout << endl;
cout << "You moved right." << endl;
cout << "Your position is now: " << Robot.position.x << "," << Robot.position.y << endl;
cout << endl;
cout << endl;
cout << "You can't go that way!" << endl;
cout << "You have walked off the grid!" << endl;
cout << endl;
cout << "Game Over." << endl;
cout << endl;
cout << endl; //new line
cout << "You moved left." << endl;
cout << "Your position is now: " << Robot.position.x << "," << Robot.position.y << endl;
cout << endl;
cout << endl;
cout << "You moved up." << endl;
cout << "Your position is now: " << Robot.position.x << "," << Robot.position.y << endl;
cout << endl;
cout << endl;
cout << "You moved down." << endl;
cout << "Your position is now: " << Robot.position.x << "," << Robot.position.y << endl;
cout << endl;
cout << "Last known location: " << Robot.position.x << "," << Robot.position.y << endl;
```

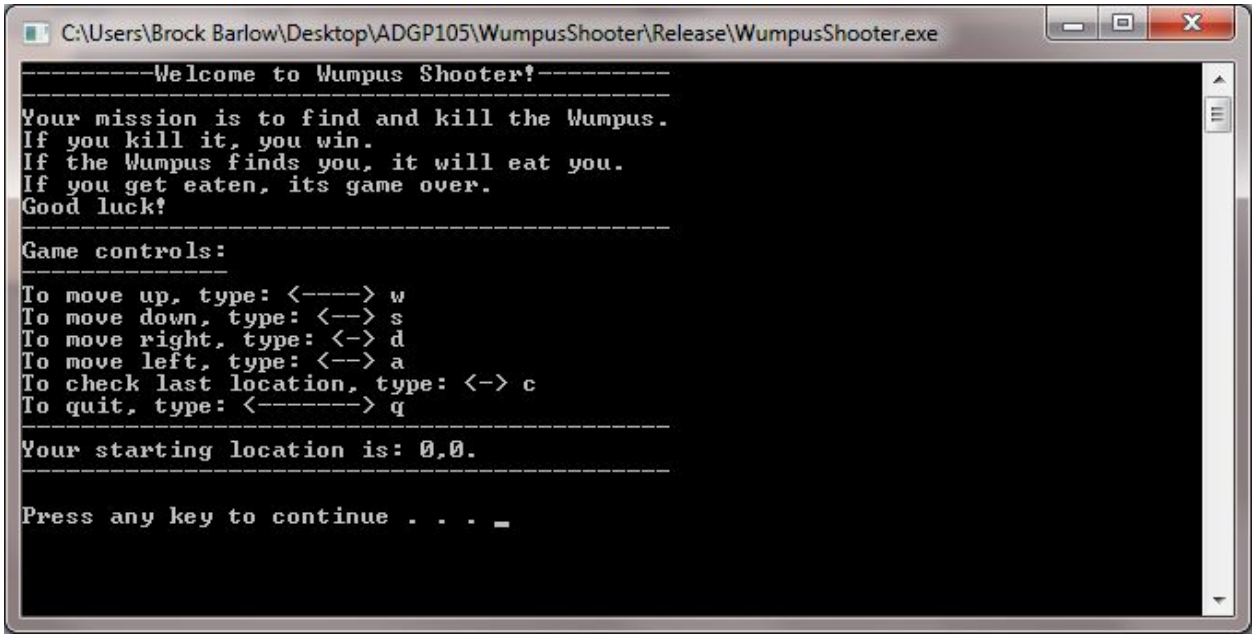
Description: These output streams are used to inform the player about their position, whether they died or not, if they can shoot the wumpus, if they can feel a breeze, if they fall down a pit and their last known position
Format: Standard output text
Size: No minimum, no maximum
Sample: cout << “You moved up.” << endl; will be printed to the screen after the player has moved up in the game

Output Items:
Description: The output is used to inform the player
Type: Text data
Range of acceptable values: Whatever data is inside “”

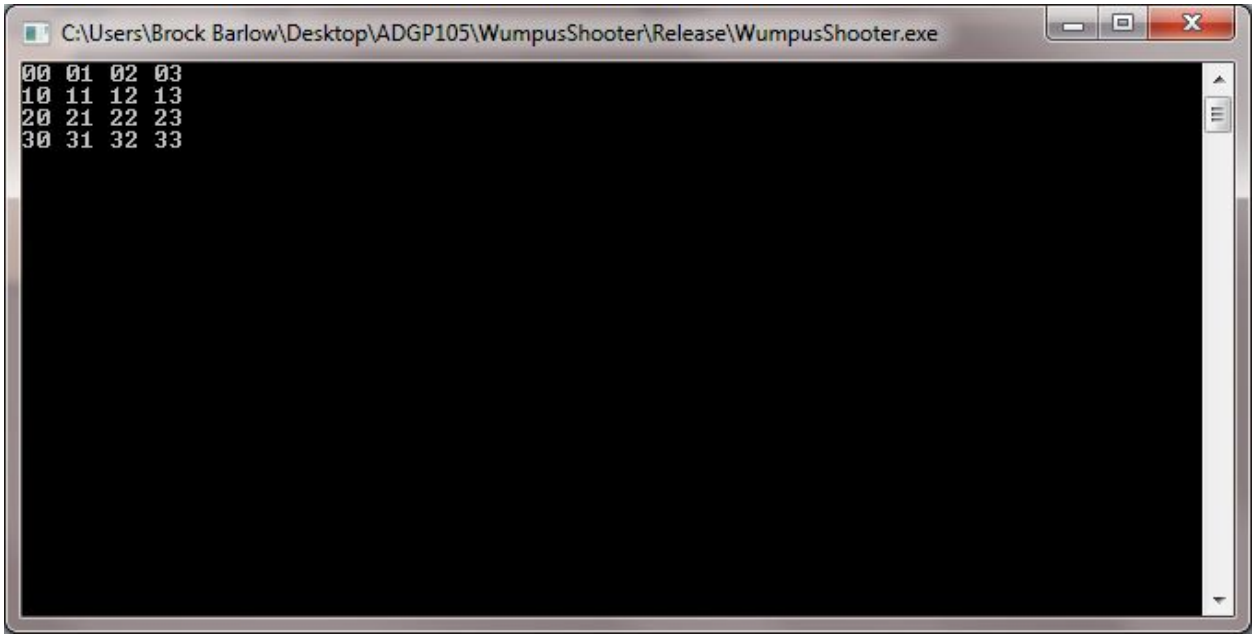
User Interface Information:

<u>Description:</u>

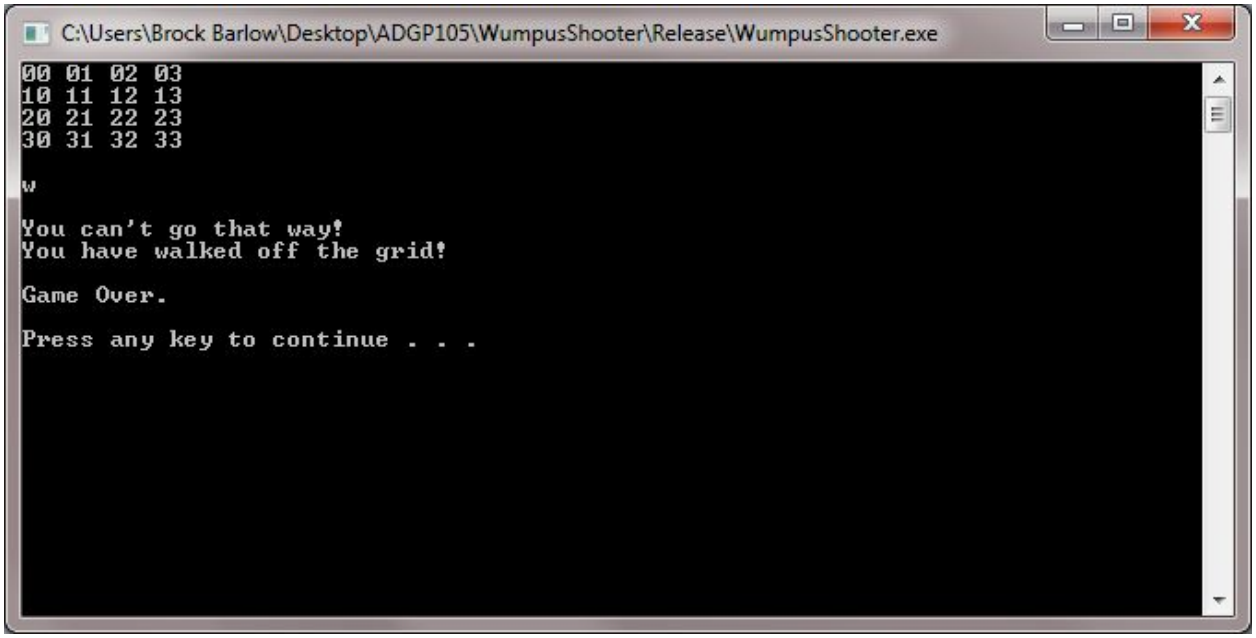
When the game starts, this title screen will appear:



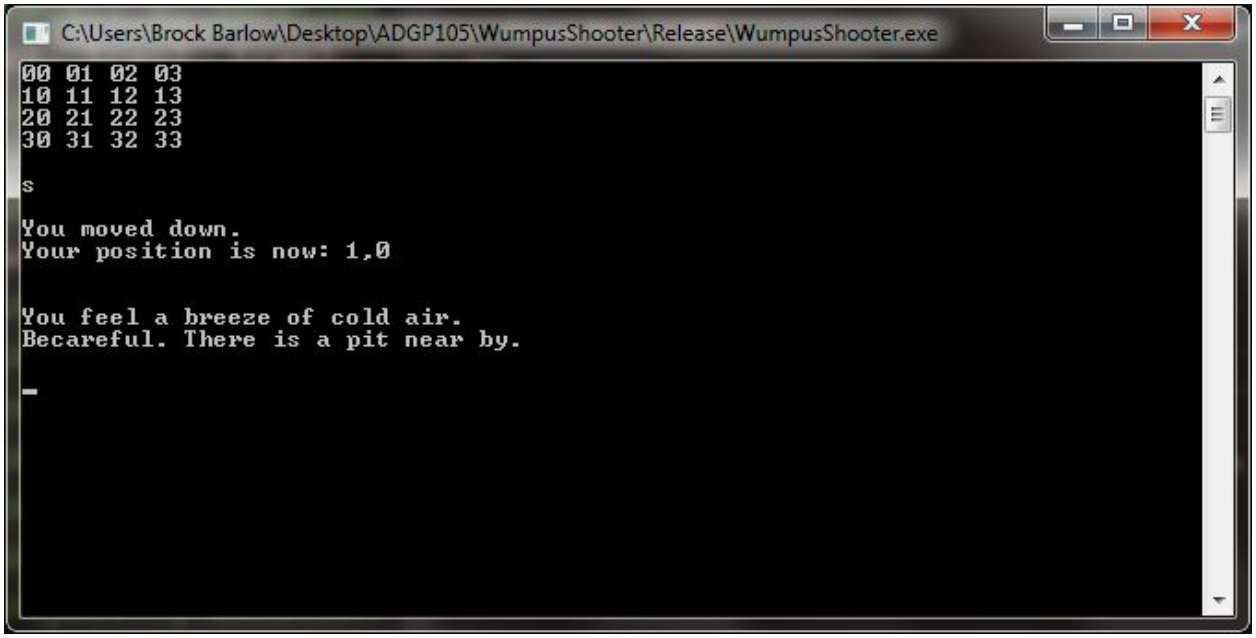
The screen will clear and will be replaced with the grid:



If the player moves off the grid, this will happen:



If the player enters a room with a breeze, this is what they will see:



If the player enters a room with a pit, this will happen:

```
C:\Users\Brock Barlow\Desktop\ADGP105\WumpusShooter\Release\WumpusShooter.exe
00 01 02 03
10 11 12 13
20 21 22 23
30 31 32 33

s

You moved down.
Your position is now: 1,0


You feel a breeze of cold air.
Becareful. There is a pit near by.

s

You moved down.
Your position is now: 2,0


You have fallen down a pit and died!

Game Over.

Press any key to continue . . .
```

If the player enters a room with a smell, this will happen:

```
C:\Users\Brock Barlow\Desktop\ADGP105\WumpusShooter\Release\WumpusShooter.exe

You feel a breeze of cold air.
Becareful. There is a pit near by.

d

You moved right.
Your position is now: 1,1


You feel a breeze of cold air.
Becareful. There is a pit near by.

s

You moved down.
Your position is now: 2,1


You smell a horrible smell!
The Wumpus is near, ready yourself!


Do you want to shoot the Wumpus? (y or n)
```

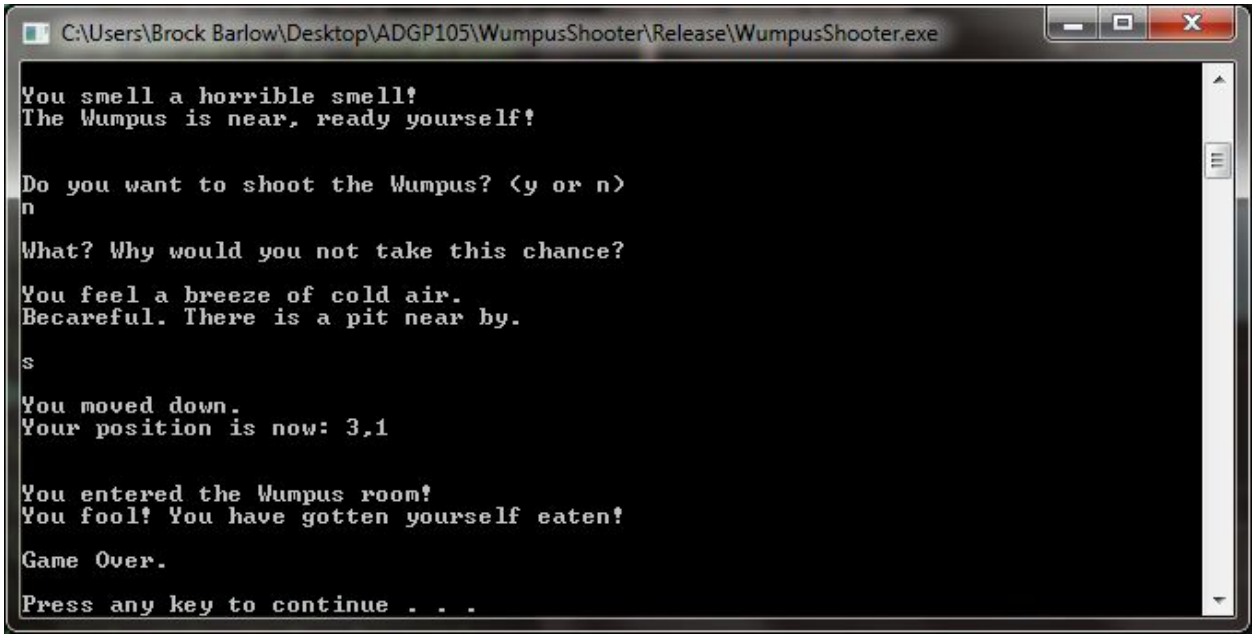
When the player kills the wumpus, this happens:

```
C:\Users\Brock Barlow\Desktop\ADGP105\WumpusShooter\Release\WumpusShooter.exe

You feel a breeze of cold air.
Becareful. There is a pit near by.

s

You moved down.
Your position is now: 2,1


You smell a horrible smell!
The Wumpus is near, ready yourself!


Do you want to shoot the Wumpus? (y or n)
y

You fired at the Wumpus' location.
You hit the Wumpus!
You did it! You won!

Game Over.

Press any key to continue . . .
```

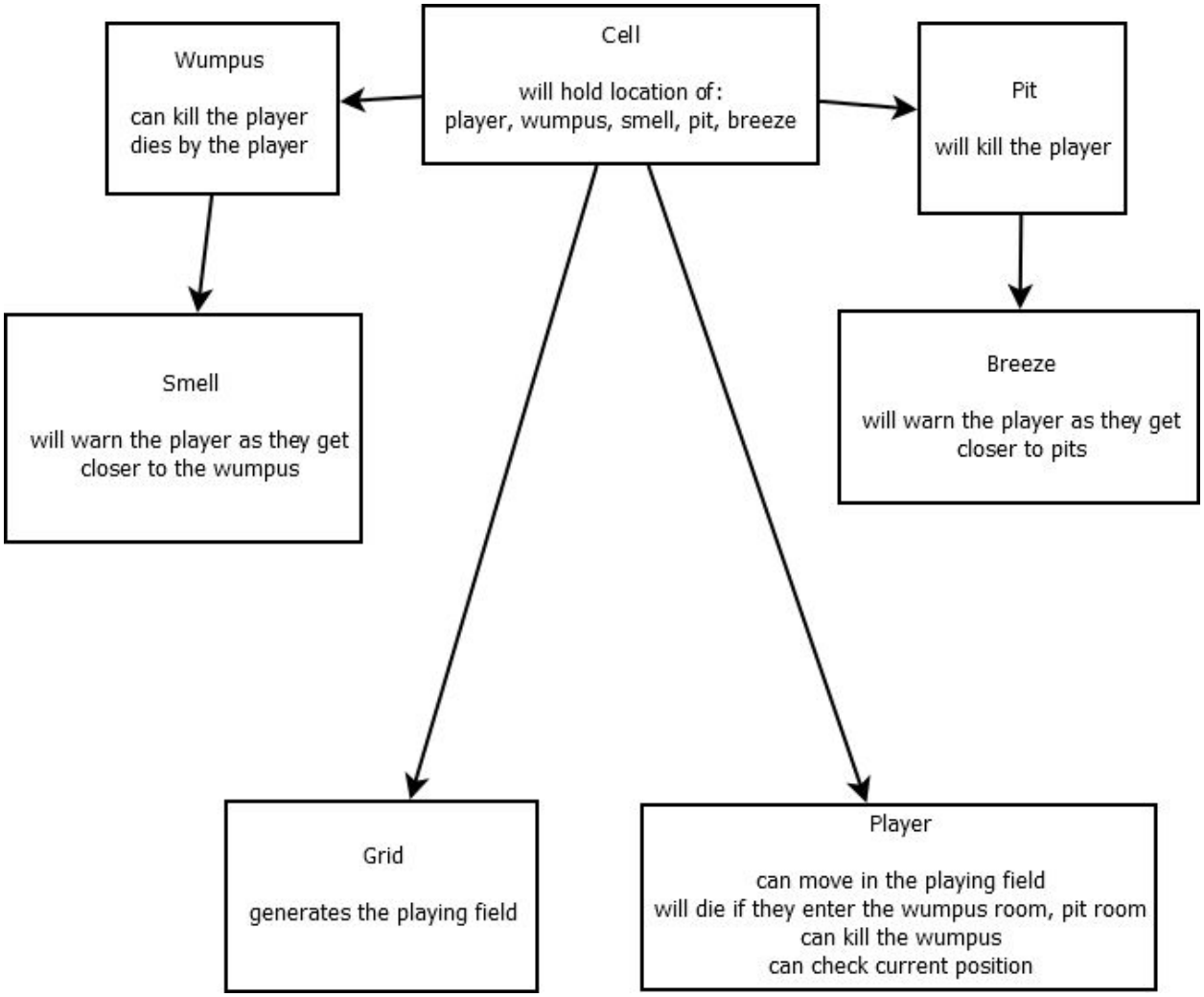
If the player does not kill the wumpus, this happens:

```
C:\Users\Brock Barlow\Desktop\ADGP105\WumpusShooter\Release\WumpusShooter.exe
Your position is now: 1,1


You feel a breeze of cold air.
Becareful. There is a pit near by.

s

You moved down.
Your position is now: 2,1


You smell a horrible smell!
The Wumpus is near, ready yourself!


Do you want to shoot the Wumpus? (y or n)
n

What? Why would you not take this chance?

You feel a breeze of cold air.
Becareful. There is a pit near by.
```

If the player enters the wumpus room, this happens:



Design Documentation:



Information about the Objects:

Class Information:

Name: class Cell

Description: Used to form grid and hold positions of the player, wumpus, smell, pit, breeze, and shoot locations.

Class Attributes: int x, int y

```
Cell(int posx, int posy)
        {
                x = posx;
                y = posy;
        }
bool operator==(Cell &other) {
                if (x == other.x && y == other.y)
                        return true; //return true flag
                return false; //return false flag
        }
```

Description: Will assign positions to the various game items and form grid
Type: integer data type
Range of acceptable values: no min, no max

Name: struct Shoot
Description: Used to set shoot locations
Class Attributes: Cell aPosition
Type: integer data type
Range of acceptable values: no min, no max

Name: class Player
Description: Used to make player and set player locations
Class Attributes: Cell position
Player() {}
Type: integer data type
Range of acceptable values: no min, no max

Name: struct Pit
Description: Used to set pit locations
Class Attributes: Cell pPosition
Type: integer data type
Range of acceptable values: no min, no max

Name: class Monster
Description: Used to make wumpus and set wumpus locations
Class Attributes: Cell mposition
Monster() {}
Type: integer data type
Range of acceptable values: no min, no max

Name: struct Smell
Description: Used to set smell locations
Class Attributes: Cell sPosition
Type: integer data type
Range of acceptable values: no min, no max

Name: struct Breeze
Description: Used to set breeze locations
Class Attributes: Cell bPosition
Type: integer data type
Range of acceptable values: no min, no max

Information about the Main Application:

The main cpp has the following global variables:
Player Robot; //player object
Monster Wumpus; //monster object
Cell grid[16]; //cell grid object array that will be 4X4.
Smell smelly1, smelly2, smelly3;
//smell object 1, location 3,0 (position x 3, position y 0)
//smell object 2, location 2,1 (position x 2, position y 1)
//smell object 3, location 3,2 (position x 3, position y 2)
Shoot fire1, fire2, fire3;
//fire object 1, location 3,0 (position x 3, position y 0)
//fire object 2, location 2,1 (position x 2, position y 1)
//fire object 3, location 3,2 (position x 3, position y 2)
Pit trap1, trap2, trap3;
//trap object 1, location 2,0 (position x 2, position y 0)
//trap object 2, location 0,1 (position x 0, position y 1)
//trap object 3, location 2,3 (position x 2, postiion y 3)
Breeze wind1, wind2, wind3, wind4, wind5, wind6, wind7, wind8, wind9;
//wind object 1, location 1,0 (position x 1, position y 0)
//wind object 2, location 1,1 (position x 1, position y 1)
//wind onject 3, location 2,2 (position x 2, position y 2)
//wind object 4, location 3,0 (position x 3, position y 0)
//wind object 5, location 2,1 (position x 2, position y 1)
//wind onject 6, location 0,0 (position x 0, position y 0)
//wind object 7, location 0,2 (position x 0, position y 2)
//wind object 8, location 1,3 (position x 1, position y 3)
//wind onject 9, location 3,3 (position x 3, position y 3)

The main cpp has the following prototype functions
void GenerateGrid(int, int, Cell[]); //prototype for generate grid
void movePlayer(char); //prototype for player movement and shooting
void checkWumpusDeath(Monster); //prototype for checking wumpus death
void checkForSmell(Smell, Smell, Smell); //prototype for checking smells
void checkIfICanShoot(Shoot, Shoot, Shoot); //prototype for checking shoot locations
void checkPitDeath(Pit, Pit, Pit); //prototype for checking pit death
void checkForBreeze(Breeze, Breeze, Breeze, Breeze, Breeze, Breeze, Breeze, Breeze, Breeze); //prototype for checking wind

//This is the main function in the main cpp
//In this function: the player will be given info about the game, the grid will generate,
//the player, wumpus, wumpus smells, and shoot locations will be given a starting value in the grid,

```
//and the while true loop will become active.
void main() //main function
{
	cout << "---------Welcome to Wumpus Shooter!---------" << endl; //info
	cout << "-------------------------------------------" << endl; //blank line
	cout << "Your mission is to find and kill the Wumpus." << endl; //info
	cout << "If you kill it, you win." << endl; //info
	cout << "If the Wumpus finds you, it will eat you." << endl; //info
	cout << "If you get eaten, its game over." << endl; //info
	cout << "Good luck!" << endl; //info
	cout << "-------------------------------------------" << endl; //blank line
	cout << "Game controls:" << endl; //info
	cout << "--------------" << endl; //blank line
	cout << "To move up, type: <----> w" << endl; //info
	cout << "To move down, type: <--> s" << endl; //info
	cout << "To move right, type: <-> d" << endl; //info
	cout << "To move left, type: <--> a" << endl; //info
	cout << "To check last location, type: <-> c" << endl; //info
	cout << "To quit, type: <-------> q" << endl; //info
	cout << "-------------------------------------------" << endl; //blank line
	cout << "Your starting location is: 0,0." << endl; //info
	cout << "-------------------------------------------" << endl; //info
	cout << endl; //new line
	system("pause"); //pauses screen
	system("cls"); //clears screen

	//Dynamic Memory Allocation
	int *p1;
	p1 = new int; //point to an unknown integer
	*p1 = 4;
	int *p2;
	p2 = new int; //point to an unknown integer
	*p2 = 4;

	//will take in '4'rows, '4'cols and cell grid array (which will hold rows ans cols)
	GenerateGrid(*p1, *p2, grid); //sends the value *p1(rows), *p2(cols) and grid (cell array)
to GenerateGrid function
	cout << endl; //new line

	//These set the positions for the player, wumpus, wumpus smells and shoot locations
	Robot.position = { 0,0 }; //players starting position
	Wumpus.mPosition = { 3,1 }; //wumpus starting position
	smelly1.sPosition = { 3,0 }; //wumpus smell one position
```

```
	smelly2.sPosition = { 2,1 }; //wumpus smell two position
	smelly3.sPosition = { 3,2 }; //wumpus smell three position
	fire1.aPosition = { 3,0 }; //shoot location one position
	fire2.aPosition = { 2,1 }; //shoot location two position
	fire3.aPosition = { 3,2 }; //shoot location three position
	trap1.pPosition = { 2,0 }; //trap one position
	trap2.pPosition = { 0,1 }; //trap two position
	trap3.pPosition = { 2,3 }; //trap three position
	wind1.bPosition = { 1,0 }; //wind one position
	wind2.bPosition = { 1,1 }; //wind two position
	wind3.bPosition = { 2,2 }; //wind three position
	wind4.bPosition = { 3,0 }; //wind four position
	wind5.bPosition = { 2,1 }; //wind five position
	wind6.bPosition = { 0,0 }; //wind six position
	wind7.bPosition = { 0,2 }; //wind seven position
	wind8.bPosition = { 1,3 }; //wind eight position
	wind9.bPosition = { 3,3 }; //wind nine position

	//while loop: while true, take in userInput and send it to the movePlayerAndShoot
function.
	while (true) //While true, do this:
	{
		char userInput, shootInput; //char variables userInput and shootInput
		cin >> userInput; //get userInput
		movePlayer(userInput); //send userInput to movePlayerAndShoot function
		checkWumpusDeath(Wumpus); //send wumpus data to function
		checkForSmell(smelly1, smelly2, smelly3); //send smell1, smell2, and smell3
data to function
		checkIfICanShoot(fire1, fire2, fire3); //send fire1, fire2, and fire3 data to function
		checkPitDeath(trap1, trap2, trap3); //send trap1, trap2, and trap3 data to function
		checkForBreeze(wind1, wind2, wind3, wind4, wind5, wind6, wind7, wind8,
wind9); //send wind1, wind2, wind3, wind4, wind5, wind6, wind7, wind8, and wind9 data to
function
	}
}

//This function will generate a grid which represents the playing field.
//This function will not return a value.
void GenerateGrid(int rows, int cols, Cell g[]) //takes in rows, cols, and cell grid array.
{
	for (int i = 0; i < rows; i++) //first for loop, goes through i
	{
		for (int j = 0; j < cols; j++) //second for loop, goes through j
```

```
                {
                        g[i].x = i; //value i will be placed in int x
                        g[j].y = j; //value j will be placed in int y
                        cout << g[i].x << g[j].y << " "; //print out grid
                }
                cout << endl; //makes grid look like a grid (square)
        }
}

//This function will check if the player dies in the wumpus room
//This function will not return a value
void checkWumpusDeath(Monster m) //gets Wumpus value
{

        Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
        Cell currentWumpusLocation = Wumpus.mPosition; //currentWumpusLocation has same
value as Wumpus.mPosition

        if (currentRobotLocation == currentWumpusLocation) //if player pos. equals monster
pos. do this:
        {
                cout << endl; //new line
                cout << "You entered the Wumpus room!" << endl; //info
                cout << "You fool! You have gotten yourself eaten!" << endl; //info
                cout << endl; //new line
                cout << "Game Over." << endl; //info
                cout << endl; //new line
                system("pause"); //pauses screen
                exit(0); //exits game
        }
}

//This function will check if there is a pit in the current cell the player is in
//This function will not return a value
//This checks for the first, second, and thrid pits
//The player will die if they enter a pit cell
void checkPitDeath(Pit p, Pit p2, Pit p3)
{
        Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
        Cell pitLocationOne = trap1.pPosition; //pitLocationOne has same value as
trap1.pPosition
```

```
	Cell pitLocationTwo = trap2.pPosition; //pitLocationTwo has same value as
trap2.pPosition
	Cell pitLocationThree = trap3.pPosition; //pitLocationThree has same value as
trap3.pPosition

	if (currentRobotLocation == pitLocationOne) //if player pos. equals pit pos. do this:
	{
		cout << endl; //new line
		cout << "You have fallen down a pit and died!" << endl; //info
		cout << endl; //new line
		cout << "Game Over." << endl; //info
		cout << endl; //new line
		system("pause"); //pauses screen
		exit(0); //exits game
	}
	if (currentRobotLocation == pitLocationTwo) //if player pos. equals pit pos. do this:
	{
		cout << endl; //new line
		cout << "You have fallen down a pit and died!" << endl; //info
		cout << endl; //new line
		cout << "Game Over." << endl; //info
		cout << endl; //new line
		system("pause"); //pauses screen
		exit(0); //exits game
	}
	if (currentRobotLocation == pitLocationThree) //if player pos. equals pit pos. do this:
	{
		cout << endl; //new line
		cout << "You have fallen down a pit and died!" << endl; //info
		cout << endl; //new line
		cout << "Game Over." << endl; //info
		cout << endl; //new line
		system("pause"); //pauses screen
		exit(0); //exits game
	}
}

//This function will check if there is a breeze in the current cell the player is in
//This function will not return a value
//This checks for the first, second, thrid, four, fifth, sixth, seventh, eighth, and nineth breeze
void checkForBreeze(Breeze b, Breeze b2, Breeze b3, Breeze b4, Breeze b5, Breeze b6,
Breeze b7, Breeze b8, Breeze b9)
{
```

```
	Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
	Cell breezeLocationOne = wind1.bPosition; //breezeLocationOne has same value as
wind1.bPosition
	Cell breezeLocationTwo = wind2.bPosition; //breezeLocationTwo has same value as
wind2.bPosition
	Cell breezeLocationThree = wind3.bPosition; //breezeLocationThree has same value as
wind3.bPosition
	Cell breezeLocationFour = wind4.bPosition; //breezeLocationFour has same value as
wind4.bPosition
	Cell breezeLocationFive = wind5.bPosition; //breezeLocationFive has same value as
wind5.bPosition
	Cell breezeLocationSix = wind6.bPosition; //breezeLocationSix has same value as
wind6.bPosition
	Cell breezeLocationSeven = wind7.bPosition; //breezeLocationSeven has same value as
wind7.bPosition
	Cell breezeLocationEight = wind8.bPosition; //breezeLocationEight has same value as
wind8.bPosition
	Cell breezeLocationNine = wind9.bPosition; //breezeLocationNine has same value as
wind9.bPosition

	if (currentRobotLocation == breezeLocationOne) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationTwo) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationThree) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
```

```
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationFour) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationFive) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationSix) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationSeven) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationEight) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
```

```
	if (currentRobotLocation == breezeLocationNine) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
}

//This function will check if there is a smell in the current cell the player is in
//This function will not return a value
//This checks for the first, second, and thrid smells
void checkForSmell(Smell s, Smell s2, Smell s3) //gets smelly1, smelly2, and smelly3 values
{
	Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
	Cell smellLocationOne = smelly1.sPosition; //smellLocationOne has same value as
smelly1.sPosition
	Cell smellLocationTwo = smelly2.sPosition; //smellLocationTwo has same value as
smelly2.sPosition
	Cell smellLocationThree = smelly3.sPosition; //smellLocationThree has same value as
smelly3.sPosition

	if (currentRobotLocation == smellLocationOne) //if player pos. equals smell1 pos. do this:
	{
		cout << endl; //new line
		cout << "You smell a horrible smell!" << endl; //info
		cout << "The Wumpus is near, ready yourself!" << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == smellLocationTwo) //if player pos. equals smell2 pos. do this:
	{
		cout << endl; //new line
		cout << "You smell a horrible smell!" << endl; //info
		cout << "The Wumpus is near, ready yourself!" << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == smellLocationThree) //if player pos. equals smell3 pos. do
this:
	{
		cout << endl; //new line
		cout << "You smell a horrible smell!" << endl; //info
```

```
                cout << "The Wumpus is near, ready yourself!" << endl; //info
                cout << endl; //new line
        }
}

//This function will check if the player can shot the wumpus.
//This function will not return a value
//This checks for the first, second, and thrid shoot locations.
void checkIfICanShoot(Shoot f, Shoot f2, Shoot f3) //gets fire1, fire2, and fire3 values
{
        char shootInput; //char variable shootInput
        Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
        Cell fireLocationOne = fire1.aPosition; //fireLocationOne has same value as
fire1.aPosition
        Cell fireLocationTwo = fire2.aPosition; //fireLocationTwo has same value as
fire2.aPosition
        Cell fireLocationThree = fire3.aPosition; //fireLocationThree has same value as
fire3.aPosition

        if (currentRobotLocation == fireLocationOne) //if player pos. equals fire1 pos. do this:
        {
                cout << endl; //new line
                cout << "Do you want to shoot the Wumpus? (y or n)" << endl; //info
                cin >> shootInput; //get shootInput
                switch (shootInput) //switch statement will take in shootInput
                {
                case 'y': //For case 'y', the player will shoot the wumpus and win the game
                        cout << endl; //new line
                        cout << "You fired at the Wumpus' location." << endl; //info
                        cout << "You hit the Wumpus!" << endl; //info
                        cout << "You did it! You won!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
                        system("pause"); //pauses screen
                        exit(0); //exits game
                        break; //breaks loop
                case 'n': //For case 'n', the player will not shoot the wumpus
                        cout << endl; //new line
                        cout << "What? Why would you not take this chance?" << endl; //info
                        break; //breaks loop
                }
```

```
}
if (currentRobotLocation == fireLocationTwo) //if player pos. equals fire2 pos. do this:
{
        cout << endl; //new line
        cout << "Do you want to shoot the Wumpus? (y or n)" << endl; //info
        cin >> shootInput; //get shootInput
        switch (shootInput) //switch statement will take in shootInput
        {
        case 'y': //For case 'y', the player will shoot the wumpus and win the game
                cout << endl; //new line
                cout << "You fired at the Wumpus' location." << endl; //info
                cout << "You hit the Wumpus!" << endl; //info
                cout << "You did it! You won!" << endl; //info
                cout << endl; //new line
                cout << "Game Over." << endl; //info
                cout << endl; //new line
                system("pause"); //pauses screen
                exit(0); //exits game
                break; //breaks loop
        case 'n': //For case 'n', the player will not shoot the wumpus
                cout << endl; //new line
                cout << "What? Why would you not take this chance?" << endl; //info
                break; //breaks loop
        }
}
if (currentRobotLocation == fireLocationThree) //if player pos. equals fire3 pos. do this:
{
        cout << endl; //new line
        cout << "Do you want to shoot the Wumpus? (y or n)" << endl; //info
        cin >> shootInput; //get shootInput
        switch (shootInput) //switch statement will take in shootInput
        {
        case 'y': //For case 'y', the player will shoot the wumpus and win the game
                cout << endl; //new line
                cout << "You fired at the Wumpus' location." << endl; //info
                cout << "You hit the Wumpus!" << endl; //info
                cout << "You did it! You won!" << endl; //info
                cout << endl; //new line
                cout << "Game Over." << endl; //info
                cout << endl; //new line
                system("pause"); //pauses screen
                exit(0); //exits game
                break; //breaks loop
```

```
                case 'n': //For case 'n', the player will not shoot the wumpus
                        cout << endl; //new line
                        cout << "What? Why would you not take this chance?" << endl; //info
                        break; //breaks loop
                }
        }
}

//This function will allow the player to move around in the grid.
//This function will not return a value.
void movePlayer(char userInput) //takes in char variable userInput
{
        switch (userInput) //switch statement will take in userInput
        {
                //For case 'd', the player will move right.
                //The players y position will increase by one.
                //The new position will be printed to the screen.
                //If the players y position is 3 and they try to move in this direction again, the
player
                //will be told they can't go that way and will fall off the play field which results
                //in a game over and the game exits.
        case 'd': //if userInput = d, do this:
                if (Robot.position.y < 3) //if position y is less than 3, do this:
                {
                        cout << endl; //new line
                        cout << "You moved right." << endl; //info
                        Robot.position.y++; //position increases
                        cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                        cout << endl; //new line
                }
                else //if position y = 3, do this:
                {
                        cout << endl; //new line
                        cout << "You can't go that way!" << endl; //info
                        cout << "You have walked off the grid!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
                        system("pause"); //pauses screen
                        exit(0); //exits game
                }
                break; //breaks loop
```

```
                    //For case 'a', the player will move left.
                    //The players y position will decrease by one.
                    //The new position will be printed to the screen.
                    //If the players y position is 0 and they try to move in this direction again,
the player
                    //will be told they can't go that way and will fall off the play field which
results
                    //in a game over and the game exits.
        case 'a': //if userInput = a, do this:
                if (Robot.position.y > 0) //if position y is greater than 0, do this:
                {
                        cout << endl; //new line
                        cout << "You moved left." << endl; //info
                        Robot.position.y--; //position decreases
                        cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                        cout << endl; //new line
                }
                else //if position y = 0, do this:
                {
                        cout << endl; //new line
                        cout << "You can't go that way!" << endl; //info
                        cout << "You have walked off the grid!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
                        system("pause"); //pauses screen
                        exit(0); //exits game
                }
                break; //breaks loop
                    //For case 'w', the player will move up.
                    //The players x position will decrease by one.
                    //The new position will be printed to the screen.
                    //If the players x position is 0 and they try to move in this direction again,
the player
                    //will be told they can't go that way and will fall off the play field which
results
                    //in a game over and the game exits.
        case 'w': //if userInput = w, do this:
                if (Robot.position.x > 0) //if position x is greater than 0, do this:
                {
                        cout << endl; //new line
                        cout << "You moved up." << endl; //info
```

```
                        Robot.position.x--; //position decreases
                        cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                        cout << endl; //new line
                }
                else //if position x = 0, do this:
                {
                        cout << endl; //new line
                        cout << "You can't go that way!" << endl; //info
                        cout << "You have walked off the grid!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
                        system("pause"); //pauses screen
                        exit(0); //exits game
                }
                break; //breaks loop
                          //For case 's', the player will move down.
                          //The players x position will increase by one.
                          //The new position will be printed to the screen.
                          //If the players x position is 3 and they try to move in this direction again,
the player
                          //will be told they can't go that way and will fall off the play field which
results
                          //in a game over and the game exits.
        case 's': //if userInput = s, do this:
                if (Robot.position.x < 3) //if position x is less than 3, do this:
                {
                        cout << endl; //new line
                        cout << "You moved down." << endl; //info
                        Robot.position.x++; //position increases
                        cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                        cout << endl; //new line
                }
                else //if postion x = 3, do this:
                {
                        cout << endl; //new line
                        cout << "You can't go that way!" << endl; //info
                        cout << "You have walked off the grid!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
```

```
                        system("pause"); //pauses screen
                        exit(0); //exits game
                }
                break; //breaks loop
                //For case 'c', the player will beable to check there current position.
        case 'c':
                file.open("textfile.txt", ios_base::in);
                cout << "Last known location: " << Robot.position.x << "," << Robot.position.y <<
endl;
                file.close();
                break;
                //For case 'q', the player will quit the game.
                //This can be done at any time.
        case 'q': //if userInput = q, do this:
                exit(0); //closes game
        }
}
```

Program Code (Source Code):

Shoot.h:

```
#pragma once
#include <iostream> //c++ lib.
#include <string> //c++ lib.
#include "Cell.h" //header file

//Shoot struct: Used to set the position for shoot locations (public)
struct Shoot //shoot struct
{
        Cell aPosition; //used for grid position
};
```

Player.h

```
#pragma once
#include <iostream> //c++ lib.
#include <string> //c++ lib.
#include "Cell.h" //header file

//Player class: holds position for player (public)
//Has a default constructor
class Player //player class
{
public: //public data
        Cell position; //used for grid position
```

```
	Player() //default constructor
	{

	}
};
```

Pit.h

```
#pragma once
#include <iostream> //c++ lib.
#include <string> //c++ lib.
#include "Cell.h" //header file

//Pit struct: Used to set the position for pit locations (public)
struct Pit //pit struct
{
	Cell pPosition; //used for grid position
};
```

Monster.h:

```
#pragma once
#include <iostream> //c++ lib.
#include <string> //c++ lib.
#include "Cell.h" //header file

//Monster class: holds position for wumpus (public)
//Has a default constructor
class Monster //monster class
{
public: //public data
	Cell mPosition; //used for grid position

	Monster() //default constructor
	{

	}
};
```

Main.cpp:

```
#include "Cell.h" //include header file
#include "Breeze.h" //include header file
#include "Monster.h" //include header file
#include "Pit.h" //include header file
```

```
#include "Player.h" //include header file
#include "Shoot.h" //include header file
#include "Smell.h" //include header file
using namespace std; //namespace lib.
ofstream file; //file output
ifstream files; //file input

//global variables
Player Robot; //player object
Monster Wumpus; //monster object
Cell grid[16]; //cell grid object array that will be 4X4.
Smell smelly1, smelly2, smelly3;
//smell object 1, location 3,0 (position x 3, position y 0)
//smell object 2, location 2,1 (position x 2, position y 1)
//smell object 3, location 3,2 (position x 3, position y 2)
Shoot fire1, fire2, fire3;
//fire object 1, location 3,0 (position x 3, position y 0)
//fire object 2, location 2,1 (position x 2, position y 1)
//fire object 3, location 3,2 (position x 3, position y 2)
Pit trap1, trap2, trap3;
//trap object 1, location 2,0 (position x 2, position y 0)
//trap object 2, location 0,1 (position x 0, position y 1)
//trap object 3, location 2,3 (position x 2, postiion y 3)
Breeze wind1, wind2, wind3, wind4, wind5, wind6, wind7, wind8, wind9;
//wind object 1, location 1,0 (position x 1, position y 0)
//wind object 2, location 1,1 (position x 1, position y 1)
//wind onject 3, location 2,2 (position x 2, position y 2)
//wind object 4, location 3,0 (position x 3, position y 0)
//wind object 5, location 2,1 (position x 2, position y 1)
//wind onject 6, location 0,0 (position x 0, position y 0)
//wind object 7, location 0,2 (position x 0, position y 2)
//wind object 8, location 1,3 (position x 1, position y 3)
//wind onject 9, location 3,3 (position x 3, position y 3)

//prototype functions
void GenerateGrid(int, int, Cell[]); //prototype for generate grid
void movePlayer(char); //prototype for player movement and shooting

void checkWumpusDeath(Monster); //prototype for checking wumpus death
void checkForSmell(Smell, Smell, Smell); //prototype for checking smells
void checkIfICanShoot(Shoot, Shoot, Shoot); //prototype for checking shoot locations
void checkPitDeath(Pit, Pit, Pit); //prototype for checking pit death
```

```
void checkForBreeze(Breeze, Breeze, Breeze, Breeze, Breeze, Breeze, Breeze, Breeze,
Breeze); //prototype for checking wind

//This is the main function.
//In this function: the player will be given info about the game, the grid will generate,
//the player, wumpus, wumpus smells, and shoot locations will be given a starting value in the
grid,
//and the while true loop will become active.
void main() //main function
{
	cout << "---------Welcome to Wumpus Shooter!---------" << endl; //info
	cout << "-------------------------------------------" << endl; //blank line
	cout << "Your mission is to find and kill the Wumpus." << endl; //info
	cout << "If you kill it, you win." << endl; //info
	cout << "If the Wumpus finds you, it will eat you." << endl; //info
	cout << "If you get eaten, its game over." << endl; //info
	cout << "Good luck!" << endl; //info
	cout << "-------------------------------------------" << endl; //blank line
	cout << "Game controls:" << endl; //info
	cout << "--------------" << endl; //blank line
	cout << "To move up, type: <----> w" << endl; //info
	cout << "To move down, type: <--> s" << endl; //info
	cout << "To move right, type: <-> d" << endl; //info
	cout << "To move left, type: <--> a" << endl; //info
	cout << "To check last location, type: <-> c" << endl; //info
	cout << "To quit, type: <-------> q" << endl; //info
	cout << "-------------------------------------------" << endl; //blank line
	cout << "Your starting location is: 0,0." << endl; //info
	cout << "-------------------------------------------" << endl; //info
	cout << endl; //new line
	system("pause"); //pauses screen
	system("cls"); //clears screen

	//Dynamic Memory Allocation
	int *p1;
	p1 = new int; //point to an unknown integer
	*p1 = 4;
	int *p2;
	p2 = new int; //point to an unknown integer
	*p2 = 4;

	//will take in '4'rows, '4'cols and cell grid array (which will hold rows ans cols)
```

```
	GenerateGrid(*p1, *p2, grid); //sends the value *p1(rows), *p2(cols) and grid (cell array)
to GenerateGrid function
	cout << endl; //new line

	//These set the positions for the player, wumpus, wumpus smells and shoot locations
	Robot.position = { 0,0 }; //players starting position
	Wumpus.mPosition = { 3,1 }; //wumpus starting position
	smelly1.sPosition = { 3,0 }; //wumpus smell one position
	smelly2.sPosition = { 2,1 }; //wumpus smell two position
	smelly3.sPosition = { 3,2 }; //wumpus smell three position
	fire1.aPosition = { 3,0 }; //shoot location one position
	fire2.aPosition = { 2,1 }; //shoot location two position
	fire3.aPosition = { 3,2 }; //shoot location three position
	trap1.pPosition = { 2,0 }; //trap one position
	trap2.pPosition = { 0,1 }; //trap two position
	trap3.pPosition = { 2,3 }; //trap three position
	wind1.bPosition = { 1,0 }; //wind one position
	wind2.bPosition = { 1,1 }; //wind two position
	wind3.bPosition = { 2,2 }; //wind three position
	wind4.bPosition = { 3,0 }; //wind four position
	wind5.bPosition = { 2,1 }; //wind five position
	wind6.bPosition = { 0,0 }; //wind six position
	wind7.bPosition = { 0,2 }; //wind seven position
	wind8.bPosition = { 1,3 }; //wind eight position
	wind9.bPosition = { 3,3 }; //wind nine position

	//while loop: while true, take in userInput and send it to the movePlayerAndShoot
function.
	while (true) //While true, do this:
	{
		char userInput, shootInput; //char variables userInput and shootInput
		cin >> userInput; //get userInput
		movePlayer(userInput); //send userInput to movePlayerAndShoot function
		checkWumpusDeath(Wumpus); //send wumpus data to function
		checkForSmell(smelly1, smelly2, smelly3); //send smell1, smell2, and smell3
data to function
		checkIfICanShoot(fire1, fire2, fire3); //send fire1, fire2, and fire3 data to function
		checkPitDeath(trap1, trap2, trap3); //send trap1, trap2, and trap3 data to function
		checkForBreeze(wind1, wind2, wind3, wind4, wind5, wind6, wind7, wind8,
wind9); //send wind1, wind2, wind3, wind4, wind5, wind6, wind7, wind8, and wind9 data to
function
	}
}
```

```
//This function will generate a grid which represents the playing field.
//This function will not return a value.
void GenerateGrid(int rows, int cols, Cell g[]) //takes in rows, cols, and cell grid array.
{
	for (int i = 0; i < rows; i++) //first for loop, goes through i
	{
		for (int j = 0; j < cols; j++) //second for loop, goes through j
		{
			g[i].x = i; //value i will be placed in int x
			g[j].y = j; //value j will be placed in int y
			cout << g[i].x << g[j].y << " "; //print out grid
		}
		cout << endl; //makes grid look like a grid (square)
	}
}

//This function will check if the player dies in the wumpus room
//This function will not return a value
void checkWumpusDeath(Monster m) //gets Wumpus value
{

	Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
	Cell currentWumpusLocation = Wumpus.mPosition; //currentWumpusLocation has same
value as Wumpus.mPosition

	if (currentRobotLocation == currentWumpusLocation) //if player pos. equals monster
pos. do this:
	{
		cout << endl; //new line
		cout << "You entered the Wumpus room!" << endl; //info
		cout << "You fool! You have gotten yourself eaten!" << endl; //info
		cout << endl; //new line
		cout << "Game Over." << endl; //info
		cout << endl; //new line
		system("pause"); //pauses screen
		exit(0); //exits game
	}
}

//This function will check if there is a pit in the current cell the player is in
//This function will not return a value
```

```
//This checks for the first, second, and thrid pits
//The player will die if they enter a pit cell
void checkPitDeath(Pit p, Pit p2, Pit p3)
{
	Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
	Cell pitLocationOne = trap1.pPosition; //pitLocationOne has same value as
trap1.pPosition
	Cell pitLocationTwo = trap2.pPosition; //pitLocationTwo has same value as
trap2.pPosition
	Cell pitLocationThree = trap3.pPosition; //pitLocationThree has same value as
trap3.pPosition

	if (currentRobotLocation == pitLocationOne) //if player pos. equals pit pos. do this:
	{
		cout << endl; //new line
		cout << "You have fallen down a pit and died!" << endl; //info
		cout << endl; //new line
		cout << "Game Over." << endl; //info
		cout << endl; //new line
		system("pause"); //pauses screen
		exit(0); //exits game
	}
	if (currentRobotLocation == pitLocationTwo) //if player pos. equals pit pos. do this:
	{
		cout << endl; //new line
		cout << "You have fallen down a pit and died!" << endl; //info
		cout << endl; //new line
		cout << "Game Over." << endl; //info
		cout << endl; //new line
		system("pause"); //pauses screen
		exit(0); //exits game
	}
	if (currentRobotLocation == pitLocationThree) //if player pos. equals pit pos. do this:
	{
		cout << endl; //new line
		cout << "You have fallen down a pit and died!" << endl; //info
		cout << endl; //new line
		cout << "Game Over." << endl; //info
		cout << endl; //new line
		system("pause"); //pauses screen
		exit(0); //exits game
	}
```

```
}

//This function will check if there is a breeze in the current cell the player is in
//This function will not return a value
//This checks for the first, second, thrid, four, fifth, sixth, seventh, eighth, and nineth breeze
void checkForBreeze(Breeze b, Breeze b2, Breeze b3, Breeze b4, Breeze b5, Breeze b6,
Breeze b7, Breeze b8, Breeze b9)
{
	Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
	Cell breezeLocationOne = wind1.bPosition; //breezeLocationOne has same value as
wind1.bPosition
	Cell breezeLocationTwo = wind2.bPosition; //breezeLocationTwo has same value as
wind2.bPosition
	Cell breezeLocationThree = wind3.bPosition; //breezeLocationThree has same value as
wind3.bPosition
	Cell breezeLocationFour = wind4.bPosition; //breezeLocationFour has same value as
wind4.bPosition
	Cell breezeLocationFive = wind5.bPosition; //breezeLocationFive has same value as
wind5.bPosition
	Cell breezeLocationSix = wind6.bPosition; //breezeLocationSix has same value as
wind6.bPosition
	Cell breezeLocationSeven = wind7.bPosition; //breezeLocationSeven has same value as
wind7.bPosition
	Cell breezeLocationEight = wind8.bPosition; //breezeLocationEight has same value as
wind8.bPosition
	Cell breezeLocationNine = wind9.bPosition; //breezeLocationNine has same value as
wind9.bPosition

	if (currentRobotLocation == breezeLocationOne) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationTwo) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
```

```
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationThree) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationFour) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationFive) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationSix) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationSeven) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
```

```
	if (currentRobotLocation == breezeLocationEight) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == breezeLocationNine) //if player pos. equals breeze pos. do
this:
	{
		cout << endl; //new line
		cout << "You feel a breeze of cold air." << endl; //info
		cout << "Becareful. There is a pit near by." << endl; //info
		cout << endl; //new line
	}
}

//This function will check if there is a smell in the current cell the player is in
//This function will not return a value
//This checks for the first, second, and thrid smells
void checkForSmell(Smell s, Smell s2, Smell s3) //gets smelly1, smelly2, and smelly3 values
{
	Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
	Cell smellLocationOne = smelly1.sPosition; //smellLocationOne has same value as
smelly1.sPosition
	Cell smellLocationTwo = smelly2.sPosition; //smellLocationTwo has same value as
smelly2.sPosition
	Cell smellLocationThree = smelly3.sPosition; //smellLocationThree has same value as
smelly3.sPosition

	if (currentRobotLocation == smellLocationOne) //if player pos. equals smell1 pos. do this:
	{
		cout << endl; //new line
		cout << "You smell a horrible smell!" << endl; //info
		cout << "The Wumpus is near, ready yourself!" << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == smellLocationTwo) //if player pos. equals smell2 pos. do this:
	{
		cout << endl; //new line
		cout << "You smell a horrible smell!" << endl; //info
```

```
		cout << "The Wumpus is near, ready yourself!" << endl; //info
		cout << endl; //new line
	}
	if (currentRobotLocation == smellLocationThree) //if player pos. equals smell3 pos. do
this:
	{
		cout << endl; //new line
		cout << "You smell a horrible smell!" << endl; //info
		cout << "The Wumpus is near, ready yourself!" << endl; //info
		cout << endl; //new line
	}
}

//This function will check if the player can shot the wumpus.
//This function will not return a value
//This checks for the first, second, and thrid shoot locations.
void checkIfICanShoot(Shoot f, Shoot f2, Shoot f3) //gets fire1, fire2, and fire3 values
{
	char shootInput; //char variable shootInput
	Cell currentRobotLocation = Robot.position; //currentRobotLocation has same value as
Robot.position
	Cell fireLocationOne = fire1.aPosition; //fireLocationOne has same value as
fire1.aPosition
	Cell fireLocationTwo = fire2.aPosition; //fireLocationTwo has same value as
fire2.aPosition
	Cell fireLocationThree = fire3.aPosition; //fireLocationThree has same value as
fire3.aPosition

	if (currentRobotLocation == fireLocationOne) //if player pos. equals fire1 pos. do this:
	{
		cout << endl; //new line
		cout << "Do you want to shoot the Wumpus? (y or n)" << endl; //info
		cin >> shootInput; //get shootInput
		switch (shootInput) //switch statement will take in shootInput
		{
		case 'y': //For case 'y', the player will shoot the wumpus and win the game
			cout << endl; //new line
			cout << "You fired at the Wumpus' location." << endl; //info
			cout << "You hit the Wumpus!" << endl; //info
			cout << "You did it! You won!" << endl; //info
			cout << endl; //new line
			cout << "Game Over." << endl; //info
			cout << endl; //new line
```

```
                system("pause"); //pauses screen
                exit(0); //exits game
                break; //breaks loop
        case 'n': //For case 'n', the player will not shoot the wumpus
                cout << endl; //new line
                cout << "What? Why would you not take this chance?" << endl; //info
                break; //breaks loop
        }
}
if (currentRobotLocation == fireLocationTwo) //if player pos. equals fire2 pos. do this:
{
        cout << endl; //new line
        cout << "Do you want to shoot the Wumpus? (y or n)" << endl; //info
        cin >> shootInput; //get shootInput
        switch (shootInput) //switch statement will take in shootInput
        {
        case 'y': //For case 'y', the player will shoot the wumpus and win the game
                cout << endl; //new line
                cout << "You fired at the Wumpus' location." << endl; //info
                cout << "You hit the Wumpus!" << endl; //info
                cout << "You did it! You won!" << endl; //info
                cout << endl; //new line
                cout << "Game Over." << endl; //info
                cout << endl; //new line
                system("pause"); //pauses screen
                exit(0); //exits game
                break; //breaks loop
        case 'n': //For case 'n', the player will not shoot the wumpus
                cout << endl; //new line
                cout << "What? Why would you not take this chance?" << endl; //info
                break; //breaks loop
        }
}
if (currentRobotLocation == fireLocationThree) //if player pos. equals fire3 pos. do this:
{
        cout << endl; //new line
        cout << "Do you want to shoot the Wumpus? (y or n)" << endl; //info
        cin >> shootInput; //get shootInput
        switch (shootInput) //switch statement will take in shootInput
        {
        case 'y': //For case 'y', the player will shoot the wumpus and win the game
                cout << endl; //new line
                cout << "You fired at the Wumpus' location." << endl; //info
```

```
                        cout << "You hit the Wumpus!" << endl; //info
                        cout << "You did it! You won!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
                        system("pause"); //pauses screen
                        exit(0); //exits game
                        break; //breaks loop
                case 'n': //For case 'n', the player will not shoot the wumpus
                        cout << endl; //new line
                        cout << "What? Why would you not take this chance?" << endl; //info
                        break; //breaks loop
                }
        }
}

//This function will allow the player to move around in the grid.
//This function will not return a value.
void movePlayer(char userInput) //takes in char variable userInput
{
        switch (userInput) //switch statement will take in userInput
        {
                //For case 'd', the player will move right.
                //The players y position will increase by one.
                //The new position will be printed to the screen.
                //If the players y position is 3 and they try to move in this direction again, the
player
                //will be told they can't go that way and will fall off the play field which results
                //in a game over and the game exits.
        case 'd': //if userInput = d, do this:
                if (Robot.position.y < 3) //if position y is less than 3, do this:
                {
                        cout << endl; //new line
                        cout << "You moved right." << endl; //info
                        Robot.position.y++; //position increases
                        cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                        cout << endl; //new line
                }
                else //if position y = 3, do this:
                {
                        cout << endl; //new line
                        cout << "You can't go that way!" << endl; //info
```

```
                cout << "You have walked off the grid!" << endl; //info
                cout << endl; //new line
                cout << "Game Over." << endl; //info
                cout << endl; //new line
                system("pause"); //pauses screen
                exit(0); //exits game
            }
            break; //breaks loop
                //For case 'a', the player will move left.
                //The players y position will decrease by one.
                //The new position will be printed to the screen.
                //If the players y position is 0 and they try to move in this direction again,
the player
                //will be told they can't go that way and will fall off the play field which
results
                //in a game over and the game exits.
        case 'a': //if userInput = a, do this:
            if (Robot.position.y > 0) //if position y is greater than 0, do this:
            {
                cout << endl; //new line
                cout << "You moved left." << endl; //info
                Robot.position.y--; //position decreases
                cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                cout << endl; //new line
            }
            else //if position y = 0, do this:
            {
                cout << endl; //new line
                cout << "You can't go that way!" << endl; //info
                cout << "You have walked off the grid!" << endl; //info
                cout << endl; //new line
                cout << "Game Over." << endl; //info
                cout << endl; //new line
                system("pause"); //pauses screen
                exit(0); //exits game
            }
            break; //breaks loop
                //For case 'w', the player will move up.
                //The players x position will decrease by one.
                //The new position will be printed to the screen.
                //If the players x position is 0 and they try to move in this direction again,
the player
```

```
                        //will be told they can't go that way and will fall off the play field which
results
                        //in a game over and the game exits.
        case 'w': //if userInput = w, do this:
                if (Robot.position.x > 0) //if position x is greater than 0, do this:
                {
                        cout << endl; //new line
                        cout << "You moved up." << endl; //info
                        Robot.position.x--; //position decreases
                        cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                        cout << endl; //new line
                }
                else //if position x = 0, do this:
                {
                        cout << endl; //new line
                        cout << "You can't go that way!" << endl; //info
                        cout << "You have walked off the grid!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
                        system("pause"); //pauses screen
                        exit(0); //exits game
                }
                break; //breaks loop
                        //For case 's', the player will move down.
                        //The players x position will increase by one.
                        //The new position will be printed to the screen.
                        //If the players x position is 3 and they try to move in this direction again,
the player
                        //will be told they can't go that way and will fall off the play field which
results
                        //in a game over and the game exits.
        case 's': //if userInput = s, do this:
                if (Robot.position.x < 3) //if position x is less than 3, do this:
                {
                        cout << endl; //new line
                        cout << "You moved down." << endl; //info
                        Robot.position.x++; //position increases
                        cout << "Your position is now: " << Robot.position.x << "," <<
Robot.position.y << endl; //info
                        cout << endl; //new line
                }
```

```
                else //if postion x = 3, do this:
                {
                        cout << endl; //new line
                        cout << "You can't go that way!" << endl; //info
                        cout << "You have walked off the grid!" << endl; //info
                        cout << endl; //new line
                        cout << "Game Over." << endl; //info
                        cout << endl; //new line
                        system("pause"); //pauses screen
                        exit(0); //exits game
                }
                break; //breaks loop
                //For case 'c', the player will beable to check there current position.
        case 'c':
                file.open("textfile.txt", ios_base::in);
                cout << "Last known location: " << Robot.position.x << "," << Robot.position.y <<
endl;
                file.close();
                break;
                //For case 'q', the player will quit the game.
                //This can be done at any time.
        case 'q': //if userInput = q, do this:
                exit(0); //closes game
        }
}
```

Smell.h:

```
#pragma once
#include <iostream> //c++ lib.
#include <string> //c++ lib.
#include "Cell.h" //header file

//Smell struct: Used to set the position for wumpus smell (public)
struct Smell //smell struct
{
        Cell sPosition; //used for grid position
};
```

Cell.h:

```
#pragma once
#include <iostream> //c++ lib.
#include <string> //c++ lib.
#include "Cell.h" //header file
```

```
//Smell struct: Used to set the position for wumpus smell (public)
struct Smell //smell struct
{
	Cell sPosition; //used for grid position
};
```

Breeze.h:

```
#pragma once
#include <iostream> //c++ lib.
#include <string> //c++ lib.
#include "Cell.h" //header file

//Smell struct: Used to set the position for wumpus smell (public)
struct Smell //smell struct
{
	Cell sPosition; //used for grid position
};
```

Operating Directions:

<u>ReadMe file:</u>

Objective:To hunt and kill the Wumpus that lives within the cave. During your hunt, you'll have to dodge pits while moving through the cave.

Game Controls:To move in this game, use the "w", "s", "a", and "d" keys. To move up, type "w". To move down, type "s". To move left, type "a". To move right, type "d". To quit the game at any time. type "q".

Rules:The cave is a 4x4 grid. Your starting position is 0,0. If you move outside of the grid, the game will end. If you enter the Wumpus room, you will die and the game ends. If you enter a pit room, you will die and the game ends. You will feel a breeze when you get closer to pits. You will smell a horrible smell as you near the Wumpus.

The debug folder, release folder, wumpusshooter folder, breeze.png, grid.png, killwumpus.png, moveoffgrid.png, nokill.png, pit.png, smell.png, titlescreen.png, wumpuskillyou.png, wumpusshooter.exe, wumpusshooter.sdf, wumpusshooter.sln, wumpusshooterdesigndoc.dia and wumpusshooterdesigndoc.png can be found in the ADGP105 folder.

To run the game, double click the wumpusshooter.exe in the main ADGP105 folder or go into the release folder, find the wumpusshooter.exe and double click.