CS 202 Homework 04

Kelby Hubbard

March 8, 2020

- Repository Link: `https://github.com/krhubbard2/CS202/tree/master/Hw004`
- Git Commits: `https://github.com/krhubbard2/CS202/commits`
- This homework took approximately 18 hours to complete.

1 Design

The design for Hunt the Wumpus was very important. In the end I ended up having a few different classes interacting with each other to get the results I wanted. I had a Wumpus, Player, Hazards and Cave classes. These all had their own unique functions but allowed me to cleanly write interactions between the objects and function correctly.

2 Post Mortem

This program was probably one of the more intense ones I have written. About a quarter through I noticed my code was not very good and had a lot of repeats. I eventually decided to scrap it all and revamp it to what it is now. Clean, neat and organized by implementing more class functions rather than if/else statements.

3 Hunt the Wumpus

3.1 Hunt the Wumpus Sample Output

Listing 1: Sample Program Output

```
SAMPLE LOSS-
Hunt the Wumpus
Would you like instructions (Y/N)? n
You are in room 6
Tunnels lead to rooms 1 3 15
I hear flapping.
Move or shoot (M/S)? m
Where to? 1
You got snatched by a bat!
You were transported to room 11
You are in room 11
Tunnels lead to rooms 7 14 18
Move or shoot (M/S)? m
Where to? 14
You are in room 14
Tunnels lead to rooms 7 11 18
Move or shoot (M/S)? m
Where to? 11
You are in room 11
Tunnels lead to rooms 7 14 18
Move or shoot (M/S)? m
Where to? 18
You are in room 18
Tunnels lead to rooms 7 14 20
I hear flapping.
Move or shoot (M/S)? m
Where to? 20
You got snatched by a bat!
You were transported to room 14
You are in room 14
Tunnels lead to rooms 7 11 18
Move or shoot (M/S)? m
```

```
Where to? 7
You are in room 7
Tunnels lead to rooms 4 11 18
Move or shoot (M/S)? m
Where to? 4
You are in room 4
Tunnels lead to rooms 2 5 7
Move or shoot (M/S)? m
Where to? 5
You are in room 5
Tunnels lead to rooms 4 8 9
Move or shoot (M/S)? m
Where to? 4
You are in room 4
Tunnels lead to rooms 2 5 7
Move or shoot (M/S)? m
Where to? 2
You are in room 2
Tunnels lead to rooms 4 8 19
I smell a Wumpus.
Move or shoot (M/S)? s
What room do you want to shoot the arrow in: 4
You hit nothing. You now have 4 arrow(s).
The Wumpus walked into your room!
Ah! The Wumpus ate you! Better luck next time.

SAMPLE WIN(with tutorial)-
Hunt the Wumpus
Would you like instructions (Y/N)? y
Welcome to Hunt the Wumpus (originally developed by
   Gregory Yob). The Wumpus lives in a cave of 20
   rooms. Each room is adjoined by three tunnels
   connecting to other rooms.

Hazards:
Bottemless pits -- Two rooms have pits. If you go in
   one of these rooms you will fall and die.
```

```
Super bats -- Two other rooms have bats. If you go in
   one of these rooms a bat will grab you and carry
   you to another room at random. Which could be
   deadly.
Wumpus:
The Wumpus is not harmed by hazards. He is usally
   sleeping. He is woken by you shooting an arrow or
   you entering his room. If the Wumpus wakes he moves
    one room. If he walks into your room he eats you.
You:
Each turn you may move or shoot an arrow that can go
   to 3 rooms.
Moving: You can move one room at a time.
Arrows: You have 5 arrows. You lose if you run out of
   arrows. Each arrow can go to 1 room. You aim by
   telling the computer the room # you want the arrow
   to go through.
If the arrow hits the Wumpus, you win.
If the arrow hits you, you lose.
Good luck.

You are in room 9
Tunnels lead to rooms 1 5 10
I feel a breeze.
I smell a Wumpus.
Move or shoot (M/S)? s
What room do you want to shoot the arrow in: 5
You killed the Wumpus! You won!
```

3.2 Git Commit Messages

Date	Message
2020-03-06	Instructions if user decides they want them.
2020-03-06	Created basic variables for player, wumpus, bats, and pits
2020-03-06	Ensured random start for all variables. Nothing will start in same room.
2020-03-06	Cave layout designed.
2020-03-06	Implamented move in wumpus.cpp
2020-03-06	Fixed syntax bug.
2020-03-06	Revamping program. Doing a class Cave layout.d
2020-03-06	Implamented Cave.cpp
2020-03-06	Implamented vector of rooms in main and distinguished connections
2020-03-06	Implamented player.hpp
2020-03-06	Implamented player.cpp and player.hpp
2020-03-06	Implamented wumpus start in main.cpp
2020-03-06	Implamented hazard.hpp and hazard.cpp
2020-03-06	Generated random starts for all player, wumpus, and hazards. No repeats
2020-03-06	Cleaned code. Deleted repeated code.
2020-03-07	Implamented seeHazards in player.cpp
2020-03-07	Finished Player::seeHazards
2020-03-07	Implamented Player::surrounding to show which rooms are connected ti you
2020-03-07	Implamented move or shoot options in main.cpp
2020-03-07	Implamented Player::move in player.cpp
2020-03-07	Implemented Player::act in main.cpp
2020-03-07	Implemented player.shoot in main.cpp
2020-03-07	Implamented waking / moving wumpus in player::shoot

3.3 main

```
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus
#include "wumpus.hpp"
#include "cave.hpp"
#include "player.hpp"
#include "hazards.hpp"

/* CAVE SYSTEM LAYOUT
1 -- 6,9,13
2 -- 4,8,19
3 -- 6,16,17
4 -- 2,5,7
5 -- 4,8,9
6 -- 1,3,15
7 -- 4,11,18
8 -- 2,5,12
9 -- 1,5,10
10 -- 9,13,16
11 -- 7,14,18
12 -- 8,17,19
13 -- 1,10,15
14 -- 7,11,18
15 -- 6,13,16
16 -- 3,10,15
17 -- 3,12,20
18 -- 7,14,20
19 -- 2,12,20
20 -- 17,18,19
*/

int randInt(int low, int high)
{
  random_device rd;
  mt19937 gen1(rd());
  uniform_int_distribution<int> dist(low,high);
  return dist(gen1);
}

int main()
{
  //Cave Layout
  vector<Cave> rooms;
  for (size_t i = 0; i < 21; i++)
  {
    Cave room(i);
    rooms.push_back(room);
  }
  rooms[1].setRoomOne(rooms[6]);
  rooms[1].setRoomTwo(rooms[9]);
  rooms[1].setRoomThree(rooms[13]);
  rooms[2].setRoomOne(rooms[4]);
  rooms[2].setRoomTwo(rooms[8]);
  rooms[2].setRoomThree(rooms[19]);
  rooms[3].setRoomOne(rooms[6]);
  rooms[3].setRoomTwo(rooms[16]);
  rooms[3].setRoomThree(rooms[17]);
```

```
59  rooms[4].setRoomOne(rooms[2]);
60  rooms[4].setRoomTwo(rooms[5]);
61  rooms[4].setRoomThree(rooms[7]);
62  rooms[5].setRoomOne(rooms[4]);
63  rooms[5].setRoomTwo(rooms[8]);
64  rooms[5].setRoomThree(rooms[9]);
65  rooms[6].setRoomOne(rooms[1]);
66  rooms[6].setRoomTwo(rooms[3]);
67  rooms[6].setRoomThree(rooms[15]);
68  rooms[7].setRoomOne(rooms[4]);
69  rooms[7].setRoomTwo(rooms[11]);
70  rooms[7].setRoomThree(rooms[18]);
71  rooms[8].setRoomOne(rooms[2]);
72  rooms[8].setRoomTwo(rooms[5]);
73  rooms[8].setRoomThree(rooms[12]);
74  rooms[9].setRoomOne(rooms[1]);
75  rooms[9].setRoomTwo(rooms[5]);
76  rooms[9].setRoomThree(rooms[10]);
77  rooms[10].setRoomOne(rooms[9]);
78  rooms[10].setRoomTwo(rooms[13]);
79  rooms[10].setRoomThree(rooms[16]);
80  rooms[11].setRoomOne(rooms[7]);
81  rooms[11].setRoomTwo(rooms[14]);
82  rooms[11].setRoomThree(rooms[18]);
83  rooms[12].setRoomOne(rooms[8]);
84  rooms[12].setRoomTwo(rooms[17]);
85  rooms[12].setRoomThree(rooms[19]);
86  rooms[13].setRoomOne(rooms[1]);
87  rooms[13].setRoomTwo(rooms[10]);
88  rooms[13].setRoomThree(rooms[15]);
89  rooms[14].setRoomOne(rooms[7]);
90  rooms[14].setRoomTwo(rooms[11]);
91  rooms[14].setRoomThree(rooms[18]);
92  rooms[15].setRoomOne(rooms[6]);
93  rooms[15].setRoomTwo(rooms[13]);
94  rooms[15].setRoomThree(rooms[16]);
95  rooms[16].setRoomOne(rooms[3]);
96  rooms[16].setRoomTwo(rooms[10]);
97  rooms[16].setRoomThree(rooms[15]);
98  rooms[17].setRoomOne(rooms[3]);
99  rooms[17].setRoomTwo(rooms[12]);
100 rooms[17].setRoomThree(rooms[20]);
101 rooms[18].setRoomOne(rooms[7]);
102 rooms[18].setRoomTwo(rooms[14]);
103 rooms[18].setRoomThree(rooms[20]);
104 rooms[19].setRoomOne(rooms[2]);
105 rooms[19].setRoomTwo(rooms[12]);
106 rooms[19].setRoomThree(rooms[20]);
107 rooms[20].setRoomOne(rooms[17]);
108 rooms[20].setRoomTwo(rooms[18]);
109 rooms[20].setRoomThree(rooms[19]);
110
111 cout << "Hunt the Wumpus" << endl;
112 //Instructions
113 cout << "Would you like instructions (Y/N)? ";
114 string instructions = "";
115 getline(cin, instructions);
116
```

```
117   //Yes instructions
118   if (instructions == "Y" || instructions == "y" || instructions == "yes")
119   {
120     cout << "Welcome to Hunt the Wumpus (originally developed by Gregory Yob)."
121          << " The Wumpus lives in a cave of 20 rooms. Each room is adjoined "
122          << "by three tunnels connecting to other rooms." << endl << endl
123          << "Hazards:" << endl << "Bottemless pits -- Two rooms have pits. "
124          << "If you go in one of these rooms you will fall and die." << endl
125          << "Super bats -- Two other rooms have bats. If you go in one of "
126          << "these rooms a bat will grab you and carry you to another room "
127          << "at random. Which could be deadly." << endl << "Wumpus:" << endl
128          << "The Wumpus is not harmed by hazards. He is usally sleeping. "
129          << "He is woken by you shooting an arrow or you entering his room. "
130          << "If the Wumpus wakes he moves one room. If he walks into your "
131          << "room he eats you." << endl << "You:" << endl << "Each turn you "
132          << "may move or shoot an arrow that can go to 3 rooms." << endl
133          << "Moving: You can move one room at a time." << endl
134          << "Arrows: You have 5 arrows. You lose if you run out of arrows. "
135          << "Each arrow can go to 1 room. You aim by telling the "
136          << "computer the room # you want the arrow to go through."
137          << endl << "If the arrow hits the Wumpus, you win." << endl
138          << "If the arrow hits you, you lose." << endl << "Good luck." << endl;
139
140   }
141
142   //Declare player, wumpus, bats, and pits
143   Player player;
144   Wumpus wumpus;
145   Hazard bat1(1);
146   Hazard bat2(1);
147   Hazard pit1(2);
148   Hazard pit2(2);
149   //Set player in random room (1-20)
150   player.setCurrentRoom(rooms[randInt(1,20)]);
151   //Set wumpus in random room (1-20)
152   while (wumpus.getCurrentRoom() == 0 ||
153          wumpus.getCurrentRoom() == player.getCurrentRoom())
154   {
155     wumpus.setCurrentRoom(rooms[randInt(1,20)]);
156   }
157   //Set bat1 in random room (1-20)
158   while (bat1.getCurrentRoom() == 0 ||
159          bat1.getCurrentRoom() == player.getCurrentRoom() ||
160          bat1.getCurrentRoom() == wumpus.getCurrentRoom())
161   {
162     bat1.setCurrentRoom(rooms[randInt(1,20)]);
163   }
164   //Set bat2 in random room (1-20)
165   while (bat2.getCurrentRoom() == 0 ||
166          bat2.getCurrentRoom() == player.getCurrentRoom() ||
167          bat2.getCurrentRoom() == wumpus.getCurrentRoom() ||
168          bat2.getCurrentRoom() == bat1.getCurrentRoom())
169   {
170     bat2.setCurrentRoom(rooms[randInt(1,20)]);
171   }
172   //Set pit1 in random room (1-20)
173   while (pit1.getCurrentRoom() == 0 ||
174          pit1.getCurrentRoom() == player.getCurrentRoom() ||
175          pit1.getCurrentRoom() == wumpus.getCurrentRoom() ||
```

```
176           pit1.getCurrentRoom() == bat1.getCurrentRoom() ||
177           pit1.getCurrentRoom() == bat2.getCurrentRoom())
178   {
179     pit1.setCurrentRoom(rooms[randInt(1,20)]);
180   }
181   //Set pit2 in random room (1-20)
182   while (pit2.getCurrentRoom() == 0 ||
183           pit2.getCurrentRoom() == player.getCurrentRoom() ||
184           pit2.getCurrentRoom() == wumpus.getCurrentRoom() ||
185           pit2.getCurrentRoom() == bat1.getCurrentRoom() ||
186           pit2.getCurrentRoom() == bat2.getCurrentRoom() ||
187           pit2.getCurrentRoom() == pit1.getCurrentRoom())
188   {
189     pit2.setCurrentRoom(rooms[randInt(1,20)]);
190   }
191
192   string input;
193
194   while (player.getAlive() == 0 && wumpus.getAliveState() == 0 &&
195           player.getArrows() != 0)
196   {
197     //If player is in danger room, act on them
198     player.act(rooms[player.getCurrentRoom()], bat1, bat2, pit1, pit2,
199                         wumpus);
200     //If player happened to die on act, skip.
201     if (player.getAlive() == 0)
202     {
203       //Shows player surrounding tunnels
204       player.surrounding(rooms[player.getCurrentRoom()]);
205       //Shows player surrounding hazards
206       player.seeHazards(rooms[player.getCurrentRoom()], bat1, bat2, pit1, pit2,
207                           wumpus);
208       cout << "Move or shoot (M/S)? ";
209       getline(cin, input);
210       if (input == "m" || input == "M")
211         {
212           //move function
213           player.move(rooms[player.getCurrentRoom()]);
214         }
215       else if (input == "s" || input == "S")
216       {
217         //shoot function
218         player.shoot(rooms[player.getCurrentRoom()],
219                       rooms[wumpus.getCurrentRoom()], wumpus);
220       }
221     }
222   }
223
224   //If player ran out of arrows, explain why they lost.
225   if (player.getArrows() == 0)
226   {
227     cout << "You ran out of arrows. Game over. Better luck next time." << endl;
228   }
229
230  return 0;
231 }
```

3.4 Cave Header

```
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus

#ifndef CAVE_HPP_
#define CAVE_HPP_

// #include "hazards.hpp"
// #include "player.hpp"
// #include "wumpus.hpp"
class Cave
{
  //Room number identification
  int _room = 0;

  //Connected rooms
  char _one = 0;
  char _two = 0;
  char _three = 0;

public:
  Cave();
  Cave(int _room);

  int getRoom() const;
  int getRoomOne() const;
  int getRoomTwo() const;
  int getRoomThree() const;

  void setRoomOne(const Cave& cave);
  void setRoomTwo(const Cave& cave);
  void setRoomThree(const Cave& cave);

};

#endif
```

3.5 Cave Source

```
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus
#include "cave.hpp"
#include <iostream>
Cave::Cave() {}

Cave::Cave(int room)
{
  _room = room;
}

int Cave::getRoom() const
{
  return _room;
}
int Cave::getRoomOne() const
```

```
{
  return _one;
}
int Cave::getRoomTwo() const
{
  return _two;
}
int Cave::getRoomThree() const
{
  return _three;
}
void Cave::setRoomOne(const Cave& room)
{
  _one = room.getRoom();
}
void Cave::setRoomTwo(const Cave& room)
{
  _two = room.getRoom();
}
void Cave::setRoomThree(const Cave& room)
{
  _three = room.getRoom();
}
```

3.6 Player Header

```
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus
#ifndef PLAYER_HPP_
#define PLAYER_HPP_
#include "cave.hpp"
#include "hazards.hpp"
#include "wumpus.hpp"


class Player
{
  //Alive = 0, Win = 1, Died to wumpus = 2, Died to pits = 3, died to arrow = 4
  int _alive = 0;
  int _currentRoom = 0;
  int _arrows = 5;

public:
  Player();
  Player(int alive, const Cave& current_room);
  int getAlive() const;
  int getCurrentRoom() const;
  int getArrows() const;
  void setCurrentRoom(const Cave& room);
  void move(const Cave& room);
  void seeHazards(const Cave& room, const Hazard& bat1,
                  const Hazard& bat2, const Hazard& pit1,
```

```
                const Hazard& pit2, const Wumpus& wumpus);
  void surrounding(const Cave& room);
  void act(const Cave& room, const Hazard& bat1,
           const Hazard& bat2, const Hazard& pit1,
           const Hazard& pit2, const Wumpus& wumpus);
  void shoot(const Cave& playerroom, const Cave& wumpusroom,
                              Wumpus& wumpus);
};

#endif
```

3.7 Player Source

```
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus
#include "player.hpp"
#include <iostream>

Player::Player() {}

void Player::setCurrentRoom(const Cave& room)
{
  _currentRoom = room.getRoom();
}

int Player::getCurrentRoom() const
{
  return _currentRoom;
}

int Player::getArrows() const
{
  return _arrows;
}

int Player::getAlive() const
{
  return _alive;
}

//Shows player hazards in connecting rooms to player
void Player::seeHazards(const Cave& room, const Hazard& bat1,
               const Hazard& bat2, const Hazard& pit1,
               const Hazard& pit2, const Wumpus& wumpus)
{
  //If bat1 is in adjacent room
  if (room.getRoomOne() == bat1.getCurrentRoom() ||
      room.getRoomTwo() == bat1.getCurrentRoom() ||
      room.getRoomThree() == bat1.getCurrentRoom())
    {
      std::cout << "I hear flapping." << std::endl;
    }

    //If bat2 is in adjacent room
    if (room.getRoomOne() == bat2.getCurrentRoom() ||
        room.getRoomTwo() == bat2.getCurrentRoom() ||
        room.getRoomThree() == bat2.getCurrentRoom())
```

```
47        {
48          std::cout << "I hear flapping." << std::endl;
49        }
50
51      //If pit1 is in adjacent room
52      if (room.getRoomOne() == pit1.getCurrentRoom() ||
53          room.getRoomTwo() == pit1.getCurrentRoom() ||
54          room.getRoomThree() == pit1.getCurrentRoom())
55        {
56          std::cout << "I feel a breeze." << std::endl;
57        }
58
59        //If pit2 is in adjacent room
60        if (room.getRoomOne() == pit2.getCurrentRoom() ||
61            room.getRoomTwo() == pit2.getCurrentRoom() ||
62            room.getRoomThree() == pit2.getCurrentRoom())
63          {
64            std::cout << "I feel a breeze." << std::endl;
65          }
66
67        //If wumpus is in an adjacent room
68        if (room.getRoomOne() == wumpus.getCurrentRoom() ||
69            room.getRoomTwo() == wumpus.getCurrentRoom() ||
70            room.getRoomThree() == wumpus.getCurrentRoom())
71          {
72            std::cout << "I smell a Wumpus." << std::endl;
73          }
74  }
75
76  //Shows connecting rooms to player
77  void Player::surrounding(const Cave& room)
78  {
79    std::cout << "You are in room " << _currentRoom << std::endl;
80    std::cout << "Tunnels lead to rooms " << room.getRoomOne() << " "
81              << room.getRoomTwo() << " " << room.getRoomThree() << std::endl;
82  }
83
84  //Allows player to move to connecting room
85  void Player::move(const Cave& room)
86  {
87    string input = "";
88    int choice;
89    bool loop = true;
90    while (loop)
91    {
92      std::cout << "Where to? ";
93      getline(cin, input);
94      istringstream iss(input);
95      iss >> choice;
96      if (iss)
97      {
98        //If player selected room 1
99        if (choice == room.getRoomOne())
100       {
101         _currentRoom = room.getRoomOne();
102         loop = false;
103       }
104       //If player selected room 2
105       else if (choice == room.getRoomTwo())
106       {
107         _currentRoom = room.getRoomTwo();
```

```
108         loop = false;
109       }
110       //If player selected room 3
111       else if (choice == room.getRoomThree())
112       {
113         _currentRoom = room.getRoomThree();
114         loop = false;
115       }
116       //Improper choice. Try again.
117       else
118       {
119         loop = true;
120       }
121     }
122     //Improper choice. Try again.
123     else
124     {
125       loop = true;
126     }
127   }
128 }
129
130 //Hazard or wumpus acts upon player if player is in same room
131 void Player::act(const Cave& room, const Hazard& bat1,
132          const Hazard& bat2, const Hazard& pit1,
133          const Hazard& pit2, const Wumpus& wumpus)
134 {
135   //If player is in same room as wumpus
136   if (room.getRoom() == wumpus.getCurrentRoom())
137   {
138     _alive = 2;
139     std::cout << "Ah! The Wumpus ate you! Better luck next time." << std::endl;
140   }
141   //If player is in same room as a pit
142   else if (room.getRoom() == pit1.getCurrentRoom() ||
143            room.getRoom() == pit2.getCurrentRoom())
144   {
145     _alive = 3;
146     std::cout << "Ahhhhhhhhh! You fell to your death." << std::endl;
147   }
148   //If player is in same room as a bat
149   else if (room.getRoom() == bat1.getCurrentRoom() ||
150            room.getRoom() == bat2.getCurrentRoom())
151   {
152     std::cout << "You got snatched by a bat!" << endl;
153     _currentRoom = randInt(1,20);
154     cout << "You were transported to room " << _currentRoom << std::endl;
155   }
156 }
157
158 //Allows player to try and shoot wumpus
159 void Player::shoot(const Cave& playerroom, const Cave& wumpusroom,
160                    Wumpus& wumpus)
161 {
162   if (_arrows > 0)
163   {
164     bool loop = true;
165     string input = "";
166     int choice;
167     while (loop)
```

```
168     {
169       cout << "What room do you want to shoot the arrow in: ";
170       getline(cin, input);
171       istringstream iss(input);
172       iss >> choice;
173       if (iss)
174       {
175         //If you shoot your own room
176         if (choice == playerroom.getRoom())
177         {
178           _alive = 4;
179           std::cout << "You shot yourself! Game over." << std::endl;
180           loop = false;
181         }
182         //If arrow goes in first room
183         else if (choice == playerroom.getRoomOne())
184         {
185           //If hit the wumpus
186           if (wumpus.getCurrentRoom() == playerroom.getRoomOne())
187           {
188             //Set wumpus to dead
189             wumpus.setAliveState(1);
190             std::cout << "You killed the Wumpus! You won!" << std::endl;
191           }
192           //If miss
193           else
194           {
195             //Minus 1 arrow.
196             _arrows -= 1;
197             std::cout << "You hit nothing. You now have " << _arrows
198                       << " arrow(s).\n";
199
200             //Wakes wumpus if he was in one of the connected rooms
201             if (wumpus.getCurrentRoom() == playerroom.getRoomOne() ||
202                 wumpus.getCurrentRoom() == playerroom.getRoomTwo() ||
203                 wumpus.getCurrentRoom() == playerroom.getRoomThree())
204             {
205               int wumpusmove = randInt(1,3);
206               if (wumpusmove == 1)
207               {
208                 wumpus.setCurrentRoom(wumpusroom.getRoomOne());
209                 if (wumpus.getCurrentRoom() == playerroom.getRoom())
210                 {
211                   std::cout << "The Wumpus walked into your room!\n";
212                 }
213               }
214               else if (wumpusmove == 2)
215               {
216                 wumpus.setCurrentRoom(wumpusroom.getRoomTwo());
217                 if (wumpus.getCurrentRoom() == playerroom.getRoom())
218                 {
219                   std::cout << "The Wumpus walked into your room!\n";
220                 }
221               }
222               else if (wumpusmove == 3)
223               {
224                 wumpus.setCurrentRoom(wumpusroom.getRoomThree());
225                 if (wumpus.getCurrentRoom() == playerroom.getRoom())
226                 {
227                   std::cout << "The Wumpus walked into your room!\n";
```

```
228                 }
229               }
230             }
231           }
232           loop = false;
233         }
234         //If arrow goes in second room
235         else if (choice == playerroom.getRoomTwo())
236         {
237           //If hit the wumpus
238           if (wumpus.getCurrentRoom() == playerroom.getRoomTwo())
239           {
240             //Set wumpus to dead
241             wumpus.setAliveState(1);
242             std::cout << "You killed the Wumpus! You won!" << std::endl;
243           }
244           //If miss
245           else
246           {
247             //Minus 1 arrow.
248             _arrows -= 1;
249             std::cout << "You hit nothing. You now have " << _arrows
250                       << " arrow(s).\n";
251                       //Wakes wumpus if he was in one of the connected rooms
252                       if (wumpus.getCurrentRoom() == playerroom.getRoomOne() ||
253                           wumpus.getCurrentRoom() == playerroom.getRoomTwo() ||
254                           wumpus.getCurrentRoom() == playerroom.getRoomThree())
255                       {
256                         int wumpusmove = randInt(1,3);
257                         if (wumpusmove == 1)
258                         {
259                           wumpus.setCurrentRoom(wumpusroom.getRoomOne());
260                           if (wumpus.getCurrentRoom() == playerroom.getRoom())
261                           {
262                             std::cout << "The Wumpus walked into your room!\n";
263                           }
264                         }
265                         else if (wumpusmove == 2)
266                         {
267                           wumpus.setCurrentRoom(wumpusroom.getRoomTwo());
268                           if (wumpus.getCurrentRoom() == playerroom.getRoom())
269                           {
270                             std::cout << "The Wumpus walked into your room!\n";
271                           }
272                         }
273                         else if (wumpusmove == 3)
274                         {
275                           wumpus.setCurrentRoom(wumpusroom.getRoomThree());
276                           if (wumpus.getCurrentRoom() == playerroom.getRoom())
277                           {
278                             std::cout << "The Wumpus walked into your room!\n";
279                           }
280                         }
281                       }
282           }
283           loop = false;
284         }
285         //If arrow goes in third room
286         else if (choice == playerroom.getRoomThree())
```

```
287         {
288           //If hit the wumpus
289           if (wumpus.getCurrentRoom() == playerroom.getRoomThree())
290           {
291             //Set wumpus to dead
292             wumpus.setAliveState(1);
293             std::cout << "You killed the Wumpus! You won!" << std::endl;
294           }
295           //If miss
296           else
297           {
298             //Minus 1 arrow.
299             _arrows -= 1;
300             std::cout << "You hit nothing. You now have " << _arrows
301                       << " arrow(s).\n";
302                       //Wakes wumpus if he was in one of the connected rooms
303                       if (wumpus.getCurrentRoom() == playerroom.getRoomOne() ||
304                           wumpus.getCurrentRoom() == playerroom.getRoomTwo() ||
305                           wumpus.getCurrentRoom() == playerroom.getRoomThree())
306                       {
307                         int wumpusmove = randInt(1,3);
308                         if (wumpusmove == 1)
309                         {
310                           wumpus.setCurrentRoom(wumpusroom.getRoomOne());
311                           if (wumpus.getCurrentRoom() == playerroom.getRoom())
312                           {
313                             std::cout << "The Wumpus walked into your room!\n";
314                           }
315                         }
316                         else if (wumpusmove == 2)
317                         {
318                           wumpus.setCurrentRoom(wumpusroom.getRoomTwo());
319                           if (wumpus.getCurrentRoom() == playerroom.getRoom())
320                           {
321                             std::cout << "The Wumpus walked into your room!\n";
322                           }
323                         }
324                         else if (wumpusmove == 3)
325                         {
326                           wumpus.setCurrentRoom(wumpusroom.getRoomThree());
327                           if (wumpus.getCurrentRoom() == playerroom.getRoom())
328                           {
329                             std::cout << "The Wumpus walked into your room!\n";
330                           }
331                         }
332                       }
333           }
334           loop = false;
335         }
336         //Improper choice
337         else
338         {
339           loop = true;
340         }
341       }
342       //Improper choice
343       else
344       {
345         loop = true;
346       }
```

```
347
348
349       }
350
351
352     }
353 }
```

3.8 Wumpus Header

```
1  // Kelby Hubbard
2  // CS202
3  // March 8, 2020
4  // Hw004 -- Hunt the Wumpus
5
6  #ifndef WUMPUS_HPP_
7  #define WUMPUS_HPP_
8
9  ////////////////////////////////////////////////////////////////////////////
10 // I N C L U D E S /////////////////////////////////////////////////////////
11 ////////////////////////////////////////////////////////////////////////////
12
13 #include <iostream>
14 using std::cin;
15 using std::cout;
16 using std::endl;
17 #include <string>
18 using std::string;
19 #include <random>
20 using std::mt19937;
21 using std::random_device;
22 using std::uniform_int_distribution;
23 #include <sstream>
24 using std::istringstream;
25 #include <vector>
26 using std::vector;
27 #include "cave.hpp"
28
29 ////////////////////////////////////////////////////////////////////////////
30 // P R O T O T Y P E S /////////////////////////////////////////////////////
31 ////////////////////////////////////////////////////////////////////////////
32
33 class Wumpus
34 {
35   int _currentRoom = 0;
36   int _alive = 0;
37
38 public:
39   Wumpus();
40   int getCurrentRoom() const;
41   int getAliveState() const;
42   void setCurrentRoom(const Cave& room);
43   void move();
44   void setAliveState(int type);
45 };
46
47 int randInt(int low, int high);
48
49
50
51 #endif
```

3.9 Wumpus Source

```cpp
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus
#include "wumpus.hpp"


Wumpus::Wumpus() {}
void Wumpus::setCurrentRoom(const Cave& room)
{
  _currentRoom = room.getRoom();
}
int Wumpus::getCurrentRoom() const
{
  return _currentRoom;
}
int Wumpus::getAliveState() const
{
  return _alive;
}
void Wumpus::setAliveState(int type)
{
  _alive = type;
}
```

3.10 Hazards Header

```cpp
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus
#ifndef HAZARDS_HPP_
#define HAZARDS_HPP_
#include "cave.hpp"

class Hazard
{
  int _room = 0;
  //Bat = 1; Pit = 2;
  int _type = 0;

public:
  Hazard();
  Hazard(int type);
  int getCurrentRoom() const;
  int getType();
  void setType(int type);
  void setCurrentRoom(const Cave& room);
};

#endif
```

3.11 Hazards Source

```
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Hunt the Wumpus
#include "hazards.hpp"

Hazard::Hazard() {}

Hazard::Hazard(int type)
{
  _type = type;
}

int Hazard::getCurrentRoom() const
{
  return _room;
}

int Hazard::getType()
{
  return _type;
}

void Hazard::setType(int type)
{
  _type = type;
}

void Hazard::setCurrentRoom(const Cave& room)
{
  _room = room.getRoom();
}
```

4 Recursion Problems

4.1 Sample Output

```
===============================================================
All tests passed (25 assertions in 1 test case)
```

4.2 Git Commit Messages

Date	Message
2020-03-07	Started basis. Catch will test for correct fib numbers
2020-03-07	Implemented fib(n) with recursion
2020-03-07	Implemented fibonacci non recursive.
2020-03-07	Implemented factorial(n) in main
2020-03-07	Implemented factorialloop (non recursive)
2020-03-07	Implemented ackermann's function

4.3 Source Code

```
// Kelby Hubbard
// CS202
// March 8, 2020
// Hw004 -- Recursion Problems
#define CATCH_CONFIG_MAIN
#include <catch2/catch.hpp>

//Fibonacci with recursion
unsigned int fib(unsigned int n)
 {
   if (n <= 1)
    {
      return n;
    }
  return fib(n-1) + fib(n-2);
}

//Fibonacci without recursion
int fib_loop(int n)
{
  if (n <= 1)
  {
    return n;
  }
  int a = 1;
  int b = 1;
  for (int i = 2; i < n; ++i)
  {
    int temp = a;
    a += b;
    b = temp;
  }
  return a;
}

//Factorial with recursion
int factorial(int n)
{
  if (n > 1)
  {
```

```
41      return n * factorial (n-1);
42    }
43    else
44    {
45      return 1;
46    }
47
48  }
49
50  //Factorial non recursion
51  int factorial_loop(int n)
52  {
53    if (n > 1)
54    {
55      int a = 1, i;
56      for (i = 1; i <= n; i++)
57      {
58        a = a * i;
59      }
60      return a;
61    }
62    else
63    {
64      return 1;
65    }
66  }
67
68  //Ackermann's Function
69  int ack(int m, int n)
70  {
71    if (m == 0)
72    {
73      return n + 1;
74    }
75    else if ((m > 0) && (n == 0))
76    {
77      return ack(m - 1, 1);
78    }
79    else if ((m > 0) && (n > 0))
80    {
81      return ack(m - 1, ack(m, n-1));
82    }
83  }
84
85  TEST_CASE( "Fibonacci Sequence", "[fibonacci]" )
86  {
87    SECTION("FIBONACCI RECURSIVE")
88    {
89      REQUIRE( fib(0) == 0 );
90      REQUIRE( fib(1) == 1 );
91      REQUIRE( fib(9) == 34 );
92      REQUIRE( fib(14) == 377 );
93    }
94
95    SECTION ("FIBONACCI NON RECURSIVE")
96    {
97      REQUIRE( fib_loop(0) == 0);
98      REQUIRE( fib_loop(1) == 1 );
99      REQUIRE( fib_loop(9) == 34 );
100     REQUIRE( fib_loop(14) == 377 );
101   }
102
```

```
103   SECTION ("FACTORIAL RECURSIVE")
104   {
105     REQUIRE (factorial(0) == 1);
106     REQUIRE (factorial(1) == 1);
107     REQUIRE (factorial(2) == 2);
108     REQUIRE (factorial(4) == 24);
109     REQUIRE (factorial(6) == 720);
110     REQUIRE (factorial(12) == 479001600);
111   }
112
113   SECTION ("FACTORIAL NON REDCURSIVE")
114   {
115     REQUIRE (factorial_loop(0) == 1);
116     REQUIRE (factorial_loop(1) == 1);
117     REQUIRE (factorial_loop(2) == 2);
118     REQUIRE (factorial_loop(4) == 24);
119     REQUIRE (factorial_loop(6) == 720);
120     REQUIRE (factorial_loop(12) == 479001600);
121   }
122
123   SECTION ("ACKERMANN'S FUNCTION")
124   {
125     REQUIRE (ack(0,0) == 1);
126     REQUIRE (ack(1,0) == 2);
127     REQUIRE (ack(0,1) == 2);
128     REQUIRE (ack(1,1) == 3);
129     REQUIRE (ack(3,4) == 125);
130
131   }
132
133 }
```