

# Computing IV Sec 204: Project Portfolio

Hunter M Hasenfus

Spring 2023

## Contents

<b>1</b>	<b>PS0: Hello World with SFML</b>	<b>3</b>
1.1	Discussion: . . . . .	3
1.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	3
1.3	Images used: . . . . .	3
1.4	What I learned : . . . . .	3
1.5	Acknowledgements: . . . . .	3
1.6	Codebase . . . . .	3
1.7	Output: . . . . .	4
<b>2</b>	<b>PS1a: Linear Feedback Shift Register</b>	<b>6</b>
2.1	Discussion: . . . . .	6
2.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	6
2.3	What I learned : . . . . .	6
2.4	Codebase . . . . .	6
2.5	Output: . . . . .	9
<b>3</b>	<b>PS1b: Application of LFSR with SFML library</b>	<b>10</b>
3.1	Discussion: . . . . .	10
3.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	10
3.3	Images used: . . . . .	10
3.4	What I learned : . . . . .	10
3.5	Acknowledgements : . . . . .	10
3.6	Codebase . . . . .	11
3.7	Output: . . . . .	14
<b>4</b>	<b>PS2a: Sokaban Visual component</b>	<b>16</b>
4.1	Discussion: . . . . .	16
4.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	16
4.3	What I learned : . . . . .	16
4.4	Acknowledgments : . . . . .	16
4.5	Codebase . . . . .	16
4.6	Output: . . . . .	23
<b>5</b>	<b>PS2b: Sokoban Game Mechanics</b>	<b>24</b>
5.1	Discussion: . . . . .	24
5.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	24
5.3	What I learned : . . . . .	24
5.4	Codebase . . . . .	25
5.5	Output: . . . . .	33
<b>6</b>	<b>PS3: Pythagoras Tree</b>	<b>34</b>
6.1	Discussion: . . . . .	34
6.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	34
6.3	What I learned : . . . . .	34
6.4	Codebase . . . . .	34
6.5	Output: . . . . .	38

<b>7</b>	<b>PS4a: Checkers Visual mechanics</b>	<b>39</b>
7.1	Discussion: . . . . .	39
7.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	39
7.3	What I learned : . . . . .	39
7.4	Codebase . . . . .	39
7.5	Output : . . . . .	46
<b>8</b>	<b>PS4b: Checkers Game mechanics</b>	<b>48</b>
8.1	Discussion: . . . . .	48
8.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	48
8.3	What I learned : . . . . .	48
8.4	Codebase . . . . .	48
8.5	Output : . . . . .	59
<b>9</b>	<b>PS5: DNA Sequence Alignment</b>	<b>61</b>
9.1	Discussion: . . . . .	61
9.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	61
9.3	What I learned : . . . . .	61
9.4	Codebase . . . . .	61
9.5	Output: . . . . .	65
<b>10</b>	<b>PS6: Random Writer</b>	<b>66</b>
10.1	Discussion: . . . . .	66
10.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	66
10.3	What I learned : . . . . .	66
10.4	Codebase . . . . .	67
<b>11</b>	<b>PS7: Kronos Log Parsing</b>	<b>73</b>
11.1	Discussion: . . . . .	73
11.2	Key algorithms, Data structures and OO Designs used in this Assignment: .	73
11.3	Explanation of the code: . . . . .	73
11.4	What I learned : . . . . .	73
11.5	Codebase . . . . .	73

# 1 PS0: Hello World with SFML

## 1.1 Discussion:

The first project of CompIV 22 is **Hello World with SFML**. In this assignment, At first we set up our environment by installing the SFML library. We also check for the newest version of the C++. The assignment tasked me with displaying two sprites to the screen, one being a green ball and another being a graphic of my choice from the internet.

## 1.2 Key algorithms, Data structures and OO Designs used in this Assignment:

This assignment did not require any of the key algorithm, Data structures and OO Designs, as the project it self is basic project with the code provided and we just had to add SFML sprite and few Keyboard events to complete the project. In the assignment I implemented a feature that takes input from the keyboard and then moves the sprite around the screen. By formatting conditional statements centered around key pressed I was able to nudge the sprite in the given direction based on the key chosen. This was an interesting project in learning how other libraries are implemented and attempting to use their objects and data structures. I learned how to configure the setup of 3rd party libraries, and I learned how to the libraries and paths are setup on linux machines.

## 1.3 Images used:

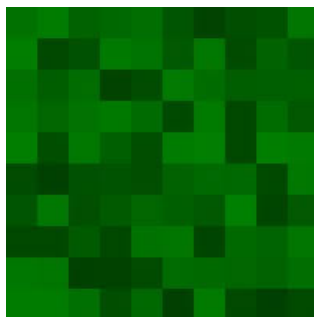


Figure 1: Sprite Image

## 1.4 What I learned :

I learned to use SFML for the first time and also how to add events and manipulate them in the SFML field. Overall, It was fun to do this assignment, as everything for me in this assignment was new an amazing.

## 1.5 Acknowledgements:

- <https://www.sfm1-dev.org/tutorials/2.5/>
- <https://youtu.be/axIgxBQVBg0>

## 1.6 Codebase

### main.cpp:

The main file where the code runs and provides the valid output as shown in figure 2.

```
1 // Hunter M Hasenfus
2 // Computing IV
3 // PS0
4 // Dr. Rykalova
5 // Jan 30, 2023
6 // Demonstrate ability to compile program using SFML and try out various
  functions.
```

```

7
8 #include <SFML/Audio.hpp>
9 #include <SFML/Graphics.hpp>
10 int main()
11 {
12     // Create the main window
13     sf::RenderWindow window(sf::VideoMode(800, 600), "SFML window");
14     // Load a sprite to display
15     sf::CircleShape shape(100.f);
16     shape.setFillColor(sf::Color::Green);
17
18     sf::Texture texture;
19     if (!texture.loadFromFile("sprite.jpg"))
20         return EXIT_FAILURE;
21     sf::Sprite sprite(texture);
22     // Create a graphical text to display
23
24     // Start the game loop
25     while (window.isOpen())
26     {
27         // Process events
28         sf::Event event;
29         while (window.pollEvent(event))
30         {
31             // Close window: exit
32             if (event.type == sf::Event::Closed)
33                 window.close();
34             if ((event.type == sf::Event::KeyPressed) && (event.key.code ==
sf::Keyboard::Left))
35                 sprite.move(-10,0);
36             if ((event.type == sf::Event::KeyPressed) && (event.key.code ==
sf::Keyboard::Right))
37                 sprite.move(10,0);
38             if ((event.type == sf::Event::KeyPressed) && (event.key.code ==
sf::Keyboard::Up))
39                 sprite.move(0,-10);
40             if ((event.type == sf::Event::KeyPressed) && (event.key.code ==
sf::Keyboard::Down))
41                 sprite.move(0,10);
42             if (event.type == sf::Event::MouseButtonPressed)
43                 sprite.rotate(90);
44
45         }
46         // Clear screen
47         window.clear();
48         // Draw the sprite
49         window.draw(shape);
50         window.draw(sprite);
51         // Update the window
52         window.display();
53     }
54     return EXIT_SUCCESS;
55 }

```

## 1.7 Output:

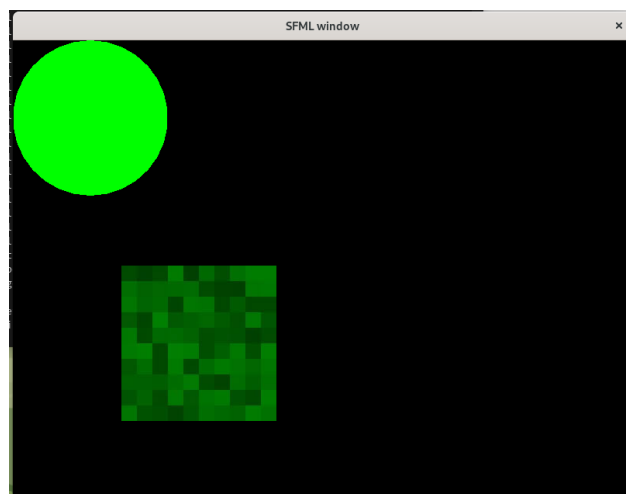


Figure 2: Output of PS0 Assignment

## 2 PS1a: Linear Feedback Shift Register

### 2.1 Discussion:

The assignment focused on constructing a class based around the LFSR, providing opaque object design through a header file, and using the boost framework to link the design to a series of tests that verified the correct implementation of the code. The ps1a assignment is an implementation of **LFSR(Linear Feedback shift Register) which is the Fibonacci LFSR**. This assignment is used for the ps1b i.e PhotoMagic. In this project, there are two main functions i.e, step() and generate(), The step() funtion is used for left shifting the one bit of the given seed, along the lsb is the result of the tap positions. These tap positions use the XOR operations and later it gives the result. The generate() generates the states according to the given k inputs.

**XOR Truth Table:**

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

### 2.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I accomplished all parts of the assignment, utilizing a simple vector of ints for the register bits, which allowed me to easily configure the spots and access all elements with random access memory. **The Tap position algorithm is as follows:**

```
1  int _TAPbitvalue = funXOR(rgs[0], rgs[2]);
2  _TAPbitvalue = funXOR(_TAPbitvalue, funGetBit(rgs[3]));
3  _TAPbitvalue = funXOR(_TAPbitvalue, funGetBit(rgs[5]));
4
```

### 2.3 What I learned :

I learned how to implement again the boost library tests.

### 2.4 Codebase

**Makefile:**

This Makefile is created by the reference of the Version2 Makefile from the notes.

```
1  CC = g++
2  CFLAGS = -c -Wall -Werror -std=c++14
3  LFLAGS = -lboost_unit_test_framework
4  DEP = FibLFSR.h
5  OBJS = FibLFSR.o test.o
6
7  all:ps1a
8
9  ps1a: $(OBJS)
10     $(CC) -o ps1a $(OBJS) $(LFLAGS)
11  FibLFSR.o: FibLFSR.cpp FibLFSR.h
12     $(CC) $(CFLAGS) FibLFSR.cpp
13  test.o: test.cpp FibLFSR.h
14     $(CC) $(CFLAGS) test.cpp $(LFLAGS)
15
16  clean :
17     rm ps1a *.o
```

### FibLFSR.h:

```
1 #ifndef FIBLFSR_H
2 #define FIBLFSR_H
3
4 #include <string>
5 #include <vector>
6 using namespace std;
7
8
9 class FibLFSR {
10 public:
11     FibLFSR(string seed);
12     int step();
13     int generate(int k);
14
15 private:
16     vector<int> registers;
17     vector<int> tapPositions = {2,3,5};
18 };
19
20
21
22 #endif
```

### FibLFSR.cpp:

```
1 // Hunter M Hasenfus
2 // Computing IV
3 // Prof. Rykalova
4 //
5 //
6
7
8 #include <iostream>
9 #include <cstdlib>
10 #include <string>
11 #include <vector>
12 #include <queue>
13 #include <cmath>
14 #include "FibLFSR.h"
15
16 using namespace std;
17
18
19 // int main(int arg, char* argv[])
20 // {
21 //     string test = "1011011000110110";
22 //     FibLFSR tes(test);
23 //     tes.generate(5);
24 //     return 0;
25 // }
26
27 #define PRINT 0
28 FibLFSR::FibLFSR(string seed)
29 {
30     this->registers = vector<int>(seed.size());
31     for(long unsigned int i = 0; i < seed.size(); i++)
32     {
33         this->registers[i] = seed[i]-48;
```

```

34     }
35 }
36
37 int FibLFSR::step()
38 {
39     int x = this->registers[0];
40     for(long unsigned int i = 0; i < this->tapPositions.size(); i++)
41     {
42         x = this->registers[this->tapPositions[i]] ^ x;
43     }
44     for(long unsigned int i = 0; i < this->registers.size()-1; i++)
45     {
46         this->registers[i] = this->registers[i+1];
47     }
48     this->registers[this->registers.size()-1] = x;
49
50     return x;
51 }
52
53 int FibLFSR::generate(int k)
54 {
55     int x= 0, y;
56     for(int i = 0; i < k; i++)
57     {
58         y = this->step();
59         x+= y * pow(2,k-1-i);
60     }
61     if(PRINT)
62     {
63         for(long unsigned int j = 0; j < this->registers.size(); j++)
64         {
65             cout << this->registers[j];
66         }
67         cout << " " << x << endl;
68     }
69
70     return x;
71 }

```

#### test.cpp:

```

1 // Dr. Rykalova
2 // test.cpp for PS1a
3 // updated 1/31/2020
4
5 #include <iostream>
6 #include <string>
7
8 #include "FibLFSR.h"
9
10 #define BOOST_TEST_DYN_LINK
11 #define BOOST_TEST_MODULE Main
12 #include <boost/test/unit_test.hpp>
13
14 BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
15
16     FibLFSR l("1011011000110110");
17     BOOST_REQUIRE(l.step() == 0);
18     BOOST_REQUIRE(l.step() == 0);
19     BOOST_REQUIRE(l.step() == 0);
20     BOOST_REQUIRE(l.step() == 1);

```



```
21 BOOST_REQUIRE(l.step() == 1);
22 BOOST_REQUIRE(l.step() == 0);
23 BOOST_REQUIRE(l.step() == 0);
24 BOOST_REQUIRE(l.step() == 1);
25
26 FibLFSR l2("1011011000110110");
27 BOOST_REQUIRE(l2.generate(9) == 51);
28
29 FibLFSR l3("0000000000000000");
30 BOOST_REQUIRE(l3.step() != 1);
31 BOOST_REQUIRE(l3.generate(100) == 0);
32
33 BOOST_REQUIRE(sizeof(l3.generate(100)) == sizeof(int));
34 BOOST_REQUIRE(sizeof(l3.step()) == sizeof(int));
35 BOOST_REQUIRE(sizeof(l3.generate(100)) == sizeof(l3.step()));
36 BOOST_REQUIRE(l2.step() == 1 || l2.step() == 0);
37 }
```

## 2.5 Output:

```
● [hunting@fedora PS1a]$ ./ps1a
Running 1 test case...

*** No errors detected
```

## 3 PS1b: Application of LFSR with SFML library

### 3.1 Discussion:

This assignment branched off of the work finished in part (a). The linear feedback shift register was implemented in a way to encode images. The LFSR provides the perception of randomness and almost the trapdoor mechanism of encryption. The program steps through each pixel in the respective image and uses the object oriented nature of LFSR to reconfigure each component of the RGB based on the seed provided in the command line argument. As shown in the images below, the image can be encoded and decoded without any loss of information.

### 3.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I used The Vector STL in the LFSR as it is much easier to use and I felt flexible in it. I used Two windows and two sprites for showing the difference between the normal cat image and the encoded and decoded image.

### 3.3 Images used:



Figure 3: Cat Image

### 3.4 What I learned :

In this assignment it was very eye opening and enlightening. First we spent the time to develop our own library for LFSR and then promptly created an application that puts the technology into work. The assignment both exercised my understanding of the SFML library and also how to utilize opaque object design.

### 3.5 Acknowledgements :

- <https://www.sfm1-dev.org/tutorials/2.5/>

### 3.6 Codebase

#### Makefile:

This Makefile contains no lint but it includes the flags as well as it is extension of the ps1a Makefile.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
4 DEP = FibLFSR.h
5 OBJS = FibLFSR.o PhotoMagic.o
6
7 all:ps1b
8
9 ps1b: $(OBJS)
10     $(CC) -o ps1b $(OBJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12     $(CC) -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
14 cleanall : cleanobj
15     rm ps1b
16
17 cleanobj :
18     rm *.o
```

#### PhotoMagic.cpp:

This file is the main file where the reading and writing also the encoding and decoding of the image takes place. This file gives the output in two windows. Input window and output window of the file.

```
1 // Hunter M Hasenfus
2 // Computing IV
3 // Prof. Rykalova
4 //
5 //
6
7 #include <iostream>
8 #include <cstdlib>
9 #include <SFML/System.hpp>
10 #include <SFML/Window.hpp>
11 #include <SFML/Graphics.hpp>
12 #include "FibLFSR.h"
13
14 #define DISPLAY 1
15
16 using namespace std;
17
18 void transform(sf::Image&, FibLFSR*);
19
20 int main(int argc, char* argv[])
21 {
22     if (argc != 4){
23         cout << argc << endl;
24         return -1;}
25
26     FibLFSR seed(argv[3]);
27
28     sf::Image input;
29     input.loadFromFile(argv[1]);
30     sf::Image output;
31     output.loadFromFile(argv[1]);;
32
```

```

33     transform(output, &seed);
34
35     sf::Texture texture1;
36     texture1.loadFromImage(input);
37     sf::Texture texture2;
38     texture2.loadFromImage(output);
39     sf::Sprite sprite1(texture1);
40     sf::Sprite sprite2(texture2);
41
42     sf::RenderWindow window1(sf::VideoMode(sprite1.getTexture()->getSize().x
    * sprite1.getScale().x, sprite1.getTexture()->getSize().y * sprite1.
    getScale().y ), "Before");
43     sf::RenderWindow window2(sf::VideoMode(sprite1.getTexture()->getSize().x
    * sprite1.getScale().x, sprite1.getTexture()->getSize().y * sprite1.
    getScale().y ), "After");
44     if(DISPLAY)
45     {
46         while (window1.isOpen() && window2.isOpen())
47         {
48             sf::Event event;
49             while (window1.pollEvent(event))
50             {
51                 if (event.type == sf::Event::Closed)
52                     window1.close();
53             }
54             while (window2.pollEvent(event))
55             {
56                 if (event.type == sf::Event::Closed)
57                     window2.close();
58             }
59             window1.clear();
60             window1.draw(sprite1);
61             window1.display();
62             window2.clear();
63             window2.draw(sprite2);
64             window2.display();
65         }
66     }
67
68     output.saveToFile(argv[2]);
69
70
71
72     return 0;
73 }
74
75 void transform(sf::Image& input, FibLFSR* seed)
76 {
77     sf::Color p;
78     cout << input.getSize().x << " " << input.getSize().y << endl;
79     for(int i = 0; i < input.getSize().x; i++)
80     {
81         for(int j = 0; j < input.getSize().y; j++)
82         {
83             p = input.getPixel(i,j);
84             p.r = p.r ^ seed->generate(8);
85             p.g = p.g ^ seed->generate(8);
86             p.b = p.b ^ seed->generate(8);
87             input.setPixel(i,j,p);

```

```
88     }
89 }
90 }
```

#### FibLFSR.h:

```
1  #ifndef FIBLFSR_H
2  #define FIBLFSR_H
3
4  #include <string>
5  #include <vector>
6  using namespace std;
7
8
9  class FibLFSR {
10 public:
11     FibLFSR(string seed);
12     int step();
13     int generate(int k);
14
15 private:
16     vector<int> registers;
17     vector<int> tapPositions = {2,3,5};
18 };
19
20
21
22 #endif
```

#### FibLFSR.cpp:

```
1  // Hunter M Hasenfus
2  // Computing IV
3  // Prof. Rykalova
4  //
5  //
6
7
8  #include <iostream>
9  #include <cstdlib>
10 #include <string>
11 #include <vector>
12 #include <queue>
13 #include <cmath>
14 #include "FibLFSR.h"
15
16 using namespace std;
17
18
19 // int main(int arg, char* argv[])
20 // {
21 //     string test = "1011011000110110";
22 //     FibLFSR tes(test);
23 //     tes.generate(5);
24 //     return 0;
25 // }
26
27 #define PRINT 0
28 FibLFSR::FibLFSR(string seed)
29 {
30     this->registers = vector<int>(seed.size());
31     for(long unsigned int i = 0; i < seed.size(); i++)
```

```

32     {
33         this->registers[i] = seed[i]-48;
34     }
35 }
36
37 int FibLFSR::step()
38 {
39     int x = this->registers[0];
40     for(long unsigned int i = 0; i < this->tapPositions.size(); i++)
41     {
42         x = this->registers[this->tapPositions[i]] ^ x;
43     }
44     for(long unsigned int i = 0; i < this->registers.size()-1; i++)
45     {
46         this->registers[i] = this->registers[i+1];
47     }
48     this->registers[this->registers.size()-1] = x;
49
50     return x;
51 }
52
53 int FibLFSR::generate(int k)
54 {
55     int x= 0, y;
56     for(int i = 0; i < k; i++)
57     {
58         y = this->step();
59         x+= y * pow(2,k-1-i);
60     }
61     if(PRINT)
62     {
63         for(long unsigned int j = 0; j < this->registers.size(); j++)
64         {
65             cout << this->registers[j];
66         }
67         cout << " " << x << endl;
68     }
69
70     return x;
71 }

```

### 3.7 Output:





Figure 4: Encoded Image



Figure 5: Decode Image



## 4 PS2a: Sokaban Visual component

### 4.1 Discussion:

The task for this project was in recreating the age-old game of Sokaban. For the first part of the assignment(a), I had to create the visual aspect. Beginning with an opaque object design, I created an object for the game and had the rest of the functions within it. I did this by inheriting the public Drawable class from the SFML library, and by overloading the draw function in a virtual format. This allowed myself the ability to draw the entire class to the screen and utilize members and data structures in the process.

### 4.2 Key algorithms, Data structures and OO Designs used in this Assignment:

The Sokaban game is inherently grid like, moving boxes around the screen to loading locations. I designed the game using an array of characters that instantiated at the start of each level, the grid included the movable locations as well as the wall locations. For the other important data points like player location, box location, and dropoff location I created different data structures to hold them including a pair of integers and two vectors consisting of pairs of integers. The first time I implemented this, I had all of the information embedded into the background grid, however I soon realized that tht it would have been difficult to model multiple figures at the same location. These pieces of data were not only dynamic, but they were a layer above the background. By removing them from the grid and giving them their own data structure I was able to always draw the background, and be able to search for their locations very easily.

### 4.3 What I learned :

In this project I learned a lot about foreground and background visual effects and the utilization of different data structures to achieve the effect. I learned how to operate objected oriented design into existing libraries with the overloading of the draw function. I learned about the relationship between textures and sprites.

### 4.4 Acknowledgments :

Links :

- <https://www.sfm1-dev.org/tutorials/2.5/graphics-sprite.php>
- [https://icarus.cs.weber.edu/~dab/cs1410/textbook/11.Operators/io\\_overload.html](https://icarus.cs.weber.edu/~dab/cs1410/textbook/11.Operators/io_overload.html)
- <https://www.cplusplus.com/reference/vector/vector/>
- [https://www.sfm1-dev.org/documentation/2.5.1/classsf\\_1\\_1Drawable.php](https://www.sfm1-dev.org/documentation/2.5.1/classsf_1_1Drawable.php)
- <https://stackoverflow.com/questions/34458791/making-custom-types-drawable-with-sfm1>

### 4.5 Codebase

#### Makefile

This Makefile has no Linting as the program does not have any lints.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lsfm1-graphics -lsfm1-window -lsfm1-system
4 DEP = Sokaban.h
5 OBJS = main.o Sokaban.o
6
7 all: Sokaban
8
9 Sokaban: $(OBJS)
```



```

10 $(CC) -o KSGuitarSim $(OBJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12 $(CC) -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
14 lint:
15     cpplint.py --filter=-runtime/references,-build/c++11 --root=. *
16
17
18 cleanall : cleanobj
19     rm KSGuitarSim
20
21 cleanobj :
22     rm *.o

```

### main.cpp

```

1  #include <cstdlib>
2  #include <ctime>
3  #include <iostream>
4  #include <fstream>
5  #include <vector>
6  #include <SFML/System.hpp>
7  #include <SFML/Window.hpp>
8  #include <SFML/Graphics.hpp>
9  #include "Sokaban.h"
10
11
12 using namespace std;
13
14
15
16 int main(int arg, char* argc[])
17 {
18     Sokaban s;
19     for(int i = 1; i < arg; i++)
20     {
21         fstream f(argc[i], ios::in);
22         f >> s;
23         sf::RenderWindow window(sf::VideoMode(s.getWidth() * 64, s.getHeight() * 64), argc[i]);
24
25         while (window.isOpen())
26         {
27             sf::Event event;
28             while (window.pollEvent(event))
29             {
30                 if (event.type == sf::Event::Closed)
31                     window.close();
32             }
33
34             window.clear();
35             window.draw(s);
36             window.display();
37         }
38     }
39
40     return 0;
41 }

```

### Sokaban.h

```

1  #ifndef SOKABAN_H
2  #define SOKABAN_H
3
4
5  #include <cstdlib>
6  #include <ctime>
7  #include <iostream>
8  #include <vector>
9  #include <SFML/System.hpp>
10 #include <SFML/Window.hpp>
11 #include <SFML/Graphics.hpp>
12
13 using namespace std;
14
15 class Sokaban: public sf::Drawable{
16     public:
17         Sokaban();
18         friend istream& operator>>(istream& in, Sokaban& s);
19         ~Sokaban();
20         int getHeight() const;
21         int getWidth() const;
22         void setHeight(int h);
23         void setWidth(int w);
24         char operator[] (pair<int,int> p) const;
25     private:
26         virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
27         const;
28         int height;
29         int width;
30         char** map;
31         sf::Texture Wall;
32         sf::Texture Box;
33         sf::Texture Empty;
34         sf::Texture Storage;
35         sf::Texture Player[4];
36         pair<int,int> playerLocation;
37         vector<pair<int,int>> boxLocation;
38         vector<pair<int,int>> storageLocation;
39 };
40 #endif

```

## Sokoban.cpp

```
1 #include "Sokaban.h"
2 #include <cstdlib>
3 #include <ctime>
4 #include <iostream>
5 #include <fstream>
6 #include <utility>
7 #include <vector>
8 #include <SFML/System.hpp>
9 #include <SFML/Window.hpp>
10 #include <SFML/Graphics.hpp>
11
12
13
14 using namespace std;
15
16 Sokaban::Sokaban(){
17     Wall.loadFromFile("sokoban/block_06.png");
18     Box.loadFromFile("sokoban/crate_03.png");
19     Empty.loadFromFile("sokoban/ground_01.png");
20     Storage.loadFromFile("sokoban/ground_04.png");
21     Player[0].loadFromFile("sokoban/player_05.png");
22     Player[1].loadFromFile("sokoban/player_08.png");
23     Player[2].loadFromFile("sokoban/player_17.png");
24     Player[3].loadFromFile("sokoban/player_20.png");
25 }
26
27 Sokaban::~Sokaban(){
28     for(int i = 0; i < height; i++){
29         delete[] map[i];
30     }
31     delete[] map;
32 }
33 int Sokaban::getHeight() const {return height;}
34 int Sokaban::getWidth() const {return width;}
35 void Sokaban::setHeight(int h) {height = h;}
36 void Sokaban::setWidth(int w) {width = w;}
37 char Sokaban::operator[] (pair<int,int> p) const {return map[p.first][p.
    second];}
38
39 istream& operator>>(istream& in, Sokaban& s){
40     in >> s.height >> s.width;
41     s.map = new char*[s.height];
42     for(int i = 0; i < s.height; i++){
43         s.map[i] = new char[s.width];
44         for(int j = 0; j < s.width; j++){
45             in >> s.map[i][j];
46             if(s.map[i][j] == '@'){
47                 s.playerLocation.first = i;
48                 s.playerLocation.second = j;
49             }
50             if(s.map[i][j] == 'A'){
51                 s.boxLocation.push_back(make_pair(i,j));}
52             if(s.map[i][j] == 'a'){
53                 s.storageLocation.push_back(make_pair(i,j));}
54         }
55     }
56 }
```

```

57     return in;
58 }
59
60 void Sokaban::draw(sf::RenderTarget& target, sf::RenderStates states) const
61 {
62     for(int i = 0; i < height; i++){
63         for(int j = 0; j < width; j++){
64             sf::Sprite sprite;
65             switch(map[i][j]){
66                 case '#':
67                     sprite.setTexture(Wall);
68                     break;
69                 case 'A':
70                     sprite.setTexture(Box);
71                     break;
72                 case '.':
73                     sprite.setTexture(Empty);
74                     break;
75                 case 'a':
76                     sprite.setTexture(Storage);
77                     break;
78                 case '@':
79                     sprite.setTexture(Player[0]);
80                     break;
81                 case 'S':
82                     sprite.setTexture(Player[0]);
83                     break;
84                 case 'N':
85                     sprite.setTexture(Player[1]);
86                     break;
87                 case 'E':
88                     sprite.setTexture(Player[2]);
89                     break;
90                 case 'W':
91                     sprite.setTexture(Player[3]);
92                     break;
93             }
94             sprite.setPosition(j * 64, i * 64);
95             target.draw(sprite);
96         }
97     }
98 }

```

```

1  #include "Sokaban.h"
2  #include <cstdlib>
3  #include <ctime>
4  #include <iostream>
5  #include <fstream>
6  #include <utility>
7  #include <vector>
8  #include <SFML/System.hpp>
9  #include <SFML/Window.hpp>
10 #include <SFML/Graphics.hpp>
11
12
13
14 using namespace std;
15
16 Sokaban::Sokaban(){
17     Wall.loadFromFile("sokoban/block_06.png");

```

```

18     Box.loadFromFile("sokoban/crate_03.png");
19     Empty.loadFromFile("sokoban/ground_01.png");
20     Storage.loadFromFile("sokoban/ground_04.png");
21     Player[0].loadFromFile("sokoban/player_05.png");
22     Player[1].loadFromFile("sokoban/player_08.png");
23     Player[2].loadFromFile("sokoban/player_17.png");
24     Player[3].loadFromFile("sokoban/player_20.png");
25 }
26
27 Sokaban::~Sokaban(){
28     for(int i = 0; i < height; i++){
29         delete[] map[i];
30     }
31     delete[] map;
32 }
33 int Sokaban::getHeight() const {return height;}
34 int Sokaban::getWidth() const {return width;}
35 void Sokaban::setHeight(int h) {height = h;}
36 void Sokaban::setWidth(int w) {width = w;}
37 char Sokaban::operator[] (pair<int,int> p) const {return map[p.first][p.
    second];}
38
39 istream& operator>>(istream& in, Sokaban& s){
40     in >> s.height >> s.width;
41     s.map = new char*[s.height];
42     for(int i = 0; i < s.height; i++){
43         s.map[i] = new char[s.width];
44         for(int j = 0; j < s.width; j++){
45             in >> s.map[i][j];
46             if(s.map[i][j] == '@'){
47                 s.playerLocation.first = i;
48                 s.playerLocation.second = j;
49             }
50             if(s.map[i][j] == 'A'){
51                 s.boxLocation.push_back(make_pair(i,j));}
52             if(s.map[i][j] == 'a'){
53                 s.storageLocation.push_back(make_pair(i,j));
54             }
55         }
56     }
57     return in;
58 }
59
60 void Sokaban::draw(sf::RenderTarget& target, sf::RenderStates states) const
61 {
62     for(int i = 0; i < height; i++){
63         for(int j = 0; j < width; j++){
64             sf::Sprite sprite;
65             switch(map[i][j]){
66                 case '#':
67                     sprite.setTexture(Wall);
68                     break;
69                 case 'A':
70                     sprite.setTexture(Box);
71                     break;
72                 case '.':
73                     sprite.setTexture(Empty);
74                     break;
75                 case 'a':

```

```
76         sprite.setTexture(Storage);
77         break;
78     case '@':
79         sprite.setTexture(Player[0]);
80         break;
81     case 'S':
82         sprite.setTexture(Player[0]);
83         break;
84     case 'N':
85         sprite.setTexture(Player[1]);
86         break;
87     case 'E':
88         sprite.setTexture(Player[2]);
89         break;
90     case 'W':
91         sprite.setTexture(Player[3]);
92         break;
93     }
94     sprite.setPosition(j * 64, i * 64);
95     target.draw(sprite);
96 }
97 }
98 }
```

4.6 Output:



## 5 PS2b: Sokoban Game Mechanics

### 5.1 Discussion:

This assignment was implementing the actions, mechanics, and dynamism into the game. By transferring the player, the boxes, and the locations into vectors or pairs, it made it very easy to search and validate locations. It also made it very easy to move pieces to new location. The search and move features were very important and helpful for establishing the edge cases in the game.

### 5.2 Key algorithms, Data structures and OO Designs used in this Assignment:

In this assignment as mentioned prior, I put the pieces and the locations into vectors. I coupled this choice with an easily integrated function from the standard library `find if`, this takes iterators as well as functors. The functor I applied would check certain locations and return the iterator or the end depending on the input. This was very helpful for the not only checking the wall locations, but also checking if there were multiple boxes in front of each other.

Here is an excerpt from the code, I implemented a while loop with the `find if` function to simulate moving the line of boxes.

code:

```
1      while(x != boxLocation.end())
2      {
3          if(map[playerLocation.first - (i+1)][playerLocation.
second] == '.' || map[playerLocation.first - (i+1)][playerLocation.
second] == 'a')
4          {
5              boxes.push_back(*x);
6          }
7          i++;
8          x = find_if(boxLocation.begin(), boxLocation.end(), [
this,i](pair<int,int> p){return p.first == playerLocation.first - i && p
.second == playerLocation.second;});
9      }
10     if(map[playerLocation.first - (i)][playerLocation.second] ==
'.' || map[playerLocation.first - (i)][playerLocation.second] == 'a')
11     {
12         for(auto y : boxes)
13         {
14             vector<pair<int,int>>::iterator x = find_if(
boxLocation.begin(), boxLocation.end(), [y](pair<int,int> p){return p.
first == y.first && p.second == y.second;});
15             x->first--;
16         }
17         playerLocation.first--;
18     }
19
20
```

### 5.3 What I learned :

I learned how to implement features that have higher levels of complexity. Adding the feature of multiple boxes took a certain level of implementation. I also learned how to use lambda functions and to take advantage of the already implemented algorithms in the standard library. I also learned how to lint my code, and what proper programming looks like.



## 5.4 Codebase

### Makefile:

This Makefile contains the linting too.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
4 DEP = Sokaban.h
5 OBJS = main.o Sokaban.o
6
7 all: Sokaban
8
9 Sokaban: $(OBJS)
10     $(CC) -o KSGuitarSim $(OBJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12     $(CC) -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
14 lint:
15     cpplint.py --filter=-runtime/references,-build/c++11 --root=. *
16
17
18 cleanall : cleanobj
19     rm KSGuitarSim
20
21 cleanobj :
22     rm *.o
```

### main.cpp

```
1 #include <cstdlib>
2 #include <ctime>
3 #include <iostream>
4 #include <fstream>
5 #include <vector>
6 #include <SFML/System.hpp>
7 #include <SFML/Window.hpp>
8 #include <SFML/Graphics.hpp>
9 #include "Sokaban.h"
10
11
12 using namespace std;
13
14 int main(int arg, char* argc[])
15 {
16
17     if(arg == 1)
18         return -1;
19     for(int i = 1; i < arg; i++)
20     {
21         Sokaban s;
22         fstream f(argc[i], ios::in);
23         f >> s;
24
25         sf::RenderWindow window(sf::VideoMode(s.getWidth() * 64, s.getHeight() * 64), argc[i]);
26         while (window.isOpen())
27         {
28             sf::Event event;
29             while (window.pollEvent(event))
30             {
31                 if (event.type == sf::Event::Closed)
```

```

32         window.close();
33         if (event.type == sf::Event::KeyPressed)
34         {
35             if(event.key.code == sf::Keyboard::R)
36             {
37                 f.clear();
38                 f.seekg(0, ios::beg);
39                 f >> s;
40             }
41             else
42                 s.movePlayer(event.key.code);
43         }
44
45     }
46     window.clear();
47     window.draw(s);
48     window.display();
49 }
50 }
51
52 return 0;
53 }

```

### Sokaban.h

```

1  #ifndef SOKABAN_H
2  #define SOKABAN_H
3
4
5  #include <cstdlib>
6  #include <ctime>
7  #include <iostream>
8  #include <vector>
9  #include <SFML/System.hpp>
10 #include <SFML/Window.hpp>
11 #include <SFML/Graphics.hpp>
12
13 using namespace std;
14
15 class Sokaban: public sf::Drawable{
16     public:
17         Sokaban();
18         friend istream& operator>>(istream& in, Sokaban& s);
19         ~Sokaban();
20         int getHeight() const;
21         int getWidth() const;
22         void setHeight(int h);
23         void setWidth(int w);
24         char operator[] (pair<int,int> p) const;
25         void movePlayer(sf::Keyboard::Key k);
26         void isWon();
27     private:
28         virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
29         const;
30         int height;
31         int width;
32         char** map;
33         sf::Texture Wall;
34         sf::Texture Box;
35         sf::Texture GBox;
36         sf::Texture Empty;

```

```

36     sf::Texture Storage;
37     sf::Texture Player[4];
38     sf::Texture Win;
39     bool won;
40     pair<int,int> playerLocation;
41     vector<pair<int,int>> boxLocation;
42     vector<pair<int,int>> storageLocation;
43     char direction;
44 };
45
46 #endif

```

### Sokoban.cpp

```

1  #include "Sokaban.h"
2  #include <cstdlib>
3  #include <ctime>
4  #include <iostream>
5  #include <fstream>
6  #include <utility>
7  #include <vector>
8  #include <SFML/System.hpp>
9  #include <SFML/Window.hpp>
10 #include <SFML/Graphics.hpp>
11
12
13
14 using namespace std;
15
16 Sokaban::Sokaban(){
17     Wall.loadFromFile("sokoban/block_06.png");
18     GBox.loadFromFile("sokoban/crate_03G.png");
19     Box.loadFromFile("sokoban/crate_03.png");
20     Empty.loadFromFile("sokoban/ground_01.png");
21     Storage.loadFromFile("sokoban/ground_04.png");
22     Player[1].loadFromFile("sokoban/player_05.png");
23     Player[0].loadFromFile("sokoban/player_08.png");
24     Player[2].loadFromFile("sokoban/player_17.png");
25     Player[3].loadFromFile("sokoban/player_20.png");
26     Win.loadFromFile("sokoban/win.png");
27
28 }
29 void Sokaban::isWon()
30 {
31     if (storageLocation.size() < boxLocation.size()){
32         for(auto y : storageLocation)
33         {
34             vector<pair<int,int>>::iterator x = find_if(boxLocation.begin(),
35 boxLocation.end(), [y](pair<int,int> p){return p.first == y.first && p.
36 second == y.second;});
37             if(x == boxLocation.end()){
38                 won = false;
39                 return;}
40         }}
41     else{
42         for(auto y : boxLocation)
43         {
44             vector<pair<int,int>>::iterator x = find_if(storageLocation.
45 begin(), storageLocation.end(), [y](pair<int,int> p){return p.first == y
46 .first && p.second == y.second;});
47             if(x == storageLocation.end()){

```

```

44         won = false;
45         return;}
46     }}
47
48     won = true;
49 }
50 void Sokaban::movePlayer(sf::Keyboard::Key k)
51 {
52     vector<pair<int,int>> boxes;
53     int i = 1;
54     if(k == sf::Keyboard::Up || k == sf::Keyboard::W)
55     {
56         if(map[playerLocation.first - 1][playerLocation.second] == '.' ||
57 map[playerLocation.first - 1][playerLocation.second] == 'a')
58     {
59         vector<pair<int,int>>::iterator x = find_if(boxLocation.begin(),
60 boxLocation.end(), [this,i](pair<int,int> p){return p.first ==
61 playerLocation.first - i && p.second == playerLocation.second;});
62         if(x != boxLocation.end())
63         {
64             while(x != boxLocation.end())
65             {
66                 if(map[playerLocation.first - (i+1)][playerLocation.
67 second] == '.' || map[playerLocation.first - (i+1)][playerLocation.
68 second] == 'a')
69                 {
70                     boxes.push_back(*x);
71                 }
72                 i++;
73                 x = find_if(boxLocation.begin(), boxLocation.end(), [
74 this,i](pair<int,int> p){return p.first == playerLocation.first - i && p
75 .second == playerLocation.second;});
76             }
77             if(map[playerLocation.first - (i)][playerLocation.second] ==
78 '.' || map[playerLocation.first - (i)][playerLocation.second] == 'a')
79             {
80                 for(auto y : boxes)
81                 {
82                     vector<pair<int,int>>::iterator x = find_if(
83 boxLocation.begin(), boxLocation.end(), [y](pair<int,int> p){return p.
84 first == y.first && p.second == y.second;});
85                     x->first--;
86                 }
87                 playerLocation.first--;
88             }
89         }
90         else
91         {
92             playerLocation.first--;
93         }
94     }
95     direction = 'N';
96 }
97 if(k == sf::Keyboard::Down || k == sf::Keyboard::S)
98 {
99     if(map[playerLocation.first + 1][playerLocation.second] == '.' ||
100 map[playerLocation.first + 1][playerLocation.second] == 'a')
101     {
102         vector<pair<int,int>>::iterator x = find_if(boxLocation.begin(),

```

```

    boxLocation.end(), [this](pair<int,int> p){return (p.first == (
playerLocation.first + 1)) && p.second == playerLocation.second;});
92     if(x != boxLocation.end())
93     {
94         while(x != boxLocation.end())
95         {
96             if(map[playerLocation.first + (i+1)][playerLocation.
second] == '.' || map[playerLocation.first + (i+1)][playerLocation.
second] == 'a')
97             {
98                 boxes.push_back(*x);
99             }
100             i++;
101             x = find_if(boxLocation.begin(), boxLocation.end(), [
this,i](pair<int,int> p){return (p.first == (playerLocation.first + i))
&& p.second == playerLocation.second;});
102         }
103         if(map[playerLocation.first + (i)][playerLocation.second] ==
'. ' || map[playerLocation.first + (i)][playerLocation.second] == 'a')
104         {
105             for(auto y : boxes)
106             {
107                 vector<pair<int,int>>::iterator x = find_if(
boxLocation.begin(), boxLocation.end(), [y](pair<int,int> p){return p.
first == y.first && p.second == y.second;});
108                 x->first++;
109             }
110             playerLocation.first++;
111         }
112     }
113     else
114     {
115         playerLocation.first++;
116     }
117 }
118 direction = 'S';
119 }
120 if(k == sf::Keyboard::Left || k == sf::Keyboard::A)
121 {
122     if(map[playerLocation.first][playerLocation.second - 1] == '.' ||
map[playerLocation.first][playerLocation.second - 1] == 'a')
123     {
124         vector<pair<int,int>>::iterator x = find_if(boxLocation.begin(),
boxLocation.end(), [this](pair<int,int> p){return p.first ==
playerLocation.first && p.second == playerLocation.second - 1;});
125         if(x != boxLocation.end())
126         {
127             while(x != boxLocation.end())
128             {
129                 if(map[playerLocation.first][playerLocation.second - (i
+1)] == '.' || map[playerLocation.first][playerLocation.second - (i+1)]
== 'a')
130                 {
131                     boxes.push_back(*x);
132                 }
133                 i++;
134                 x = find_if(boxLocation.begin(), boxLocation.end(), [
this,i](pair<int,int> p){return p.first == playerLocation.first && p.

```

```

second == (playerLocation.second - i));});
136     }
137     if(map[playerLocation.first][playerLocation.second- (i)] ==
'. .' || map[playerLocation.first][playerLocation.second - (i)] == 'a')
138     {
139         for(auto y : boxes)
140         {
141             vector<pair<int,int>>::iterator x = find_if(
boxLocation.begin(), boxLocation.end(), [y](pair<int,int> p){return p.
first == y.first && p.second == y.second;});
142             x->second--;
143         }
144         playerLocation.second--;
145     }
146 }
147 else
148 {
149     playerLocation.second--;
150 }
151 }
152 direction = 'W';
153 }
154 if(k == sf::Keyboard::Right || k == sf::Keyboard::D)
155 {
156     if(map[playerLocation.first][playerLocation.second + 1] == '. .' ||
map[playerLocation.first][playerLocation.second + 1] == 'a')
157     {
158         vector<pair<int,int>>::iterator x = find_if(boxLocation.begin(),
boxLocation.end(), [this](pair<int,int> p){return p.first ==
playerLocation.first && (p.second == (playerLocation.second + 1));});
159         if(x != boxLocation.end())
160         {
161             while(x != boxLocation.end())
162             {
163                 if(map[playerLocation.first][playerLocation.second + (i
+1)] == '. .' || map[playerLocation.first][playerLocation.second + (i+1)]
== 'a')
164                 {
165                     boxes.push_back(*x);
166                 }
167                 i++;
168                 x = find_if(boxLocation.begin(), boxLocation.end(), [
this,i](pair<int,int> p){return (p.first == playerLocation.first) && (p.
second == (playerLocation.second + i));});
169             }
170             if(map[playerLocation.first][playerLocation.second + i] == '.
.' || map[playerLocation.first][playerLocation.second + i] == 'a')
171             {
172                 for(auto y : boxes)
173                 {
174                     vector<pair<int,int>>::iterator x = find_if(
boxLocation.begin(), boxLocation.end(), [y](pair<int,int> p){return (p.
first == y.first) && (p.second == y.second);});
175                     x->second++;
176                 }
177                 playerLocation.second++;
178             }
179         }
180     else

```

```

181         {
182             playerLocation.second++;
183         }
184     }
185     direction = 'E';
186 }
187 isWon();
188 }
189 Sokaban::~Sokaban(){
190     for(int i = 0; i < height; i++){
191         delete[] map[i];
192     }
193     delete[] map;
194 }
195 int Sokaban::getHeight() const {return height;}
196 int Sokaban::getWidth() const {return width;}
197 void Sokaban::setHeight(int h) {height = h;}
198 void Sokaban::setWidth(int w) {width = w;}
199 char Sokaban::operator[] (pair<int,int> p) const {return map[p.first][p.
    second];}
200
201 istream& operator>>(istream& in, Sokaban& s){
202     in >> s.height >> s.width;
203     s.won = false;
204     s.map = new char*[s.height];
205     s.boxLocation.clear();
206     s.storageLocation.clear();
207     for(int i = 0; i < s.height; i++){
208         s.map[i] = new char[s.width];
209         for(int j = 0; j < s.width; j++){
210             char x;
211             in >> x;
212             if(x == '@'){
213                 s.playerLocation.first = i;
214                 s.playerLocation.second = j;
215                 s.map[i][j] = '.';
216                 s.direction = '@';
217             }
218             if(x == 'A'){
219                 s.boxLocation.push_back(make_pair(i,j));}
220                 s.map[i][j] = '.';
221             if(x == 'a'){
222                 s.storageLocation.push_back(make_pair(i,j));
223                 s.map[i][j] = x;
224             }
225             if(x == '#'){
226                 s.map[i][j] = x;
227             }
228             if(x == '.'){
229                 s.map[i][j] = x;
230             }
231         }
232     }
233 }
234 }
235 return in;
236 }
237
238 void Sokaban::draw(sf::RenderTarget& target, sf::RenderStates states) const

```

```

239 {
240     for(int i = 0; i < height; i++){
241         for(int j = 0; j < width; j++){
242             sf::Sprite sprite;
243             switch(map[i][j]){
244                 case '#':
245                     sprite.setTexture(Wall);
246                     break;
247                 case '.':
248                     sprite.setTexture(Empty);
249                     break;
250                 case 'a':
251                     sprite.setTexture(Storage);
252                     break;
253             }
254             sprite.setPosition(j * 64, i * 64);
255             target.draw(sprite);
256         }
257     }
258
259     for(auto x: boxLocation){
260         sf::Sprite sprite;
261         auto y = find_if(storageLocation.begin(), storageLocation.end(), [x
262 ](pair<int,int> p){return (p.first == x.first) && (p.second == x.second)
263 ;});
264         if(y != storageLocation.end())
265             sprite.setTexture(GBox);
266         else
267             sprite.setTexture(Box);
268         sprite.setPosition(x.second * 64, x.first * 64);
269         target.draw(sprite);
270     }
271     sf::Sprite player;
272     switch(direction){
273         case '@':
274             player.setTexture(Player[0]);
275             break;
276         case 'N':
277             player.setTexture(Player[0]);
278             break;
279         case 'S':
280             player.setTexture(Player[1]);
281             break;
282         case 'E':
283             player.setTexture(Player[2]);
284             break;
285         case 'W':
286             player.setTexture(Player[3]);
287             break;
288     }
289     player.setPosition(playerLocation.second * 64, playerLocation.first *
290 64);
291     target.draw(player);
292
293     if(won)
294     {
295         sf::Sprite sprite;
296         sprite.setTexture(Win);

```



```

295     sprite.scale(64.0/Win.getSize().x, 64.0/Win.getSize().y);
296     sprite.setPosition(playerLocation.second * 64, playerLocation.first
    * 64);
297     target.draw(sprite);
298 }
299 }

```

## 5.5 Output:

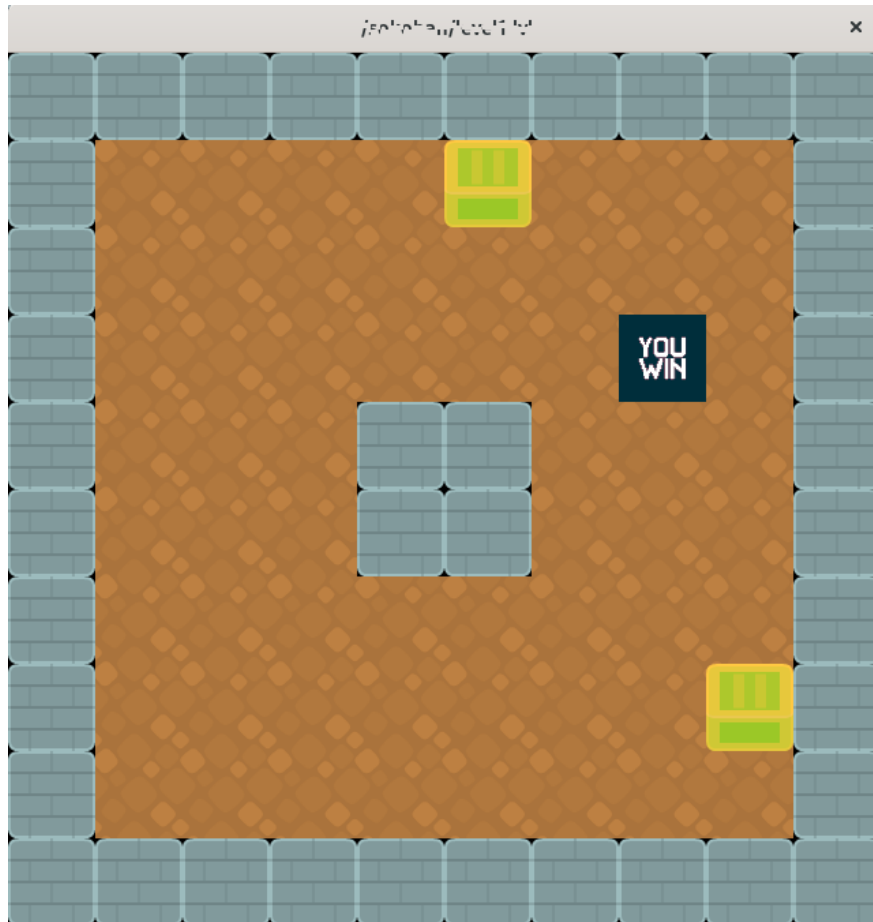


Figure 6: Player wins game

## 6 PS3: Pythagoras Tree

### 6.1 Discussion:

The assignment's objective was to create a primary function that recursively drew a fractal like image. Fractals are dictated by their self-similarity, and in this drawing, at the top of each square the triangle is drawn. Whereby each triangle can make two more squares. The recursive function has a base case, which allows for setting a finite limit on the number of calls. In our assignment, we were able to first create a function that worked for the 45 degree triangle, then afterwards we implemented the function for all angles between 1-89. We then added a coloring scheme that spanned the amount of function calls.

The extra credit feature we implemented was slightly changing the function so that it could operate with any starting angle, not just 45 degrees. This was done by fixing some aspects of the first algorithm so that they were more abstract and not specifically catered towards 45 degrees. One feature I implemented was forming a mapping from the cartesian coordinates into the video window coordinates. The dx and dy were conceptually born from the understanding of cosine and sin in the cartesian plane, so the mapping made the entire problem much easier. Lastly the coloring was implemented with two different methods, one of logarithmic and one linear.

### 6.2 Key algorithms, Data structures and OO Designs used in this Assignment:

A key data structure in the assignment was a vector of rectangles. This was essential because it allowed us to create rectangles for each function call and store them all in a central data structure. Afterwards the draw feature could very easily display all the rectangles that were pushed into the vector.

### 6.3 What I learned :

I learned the difficulty in using parametric functions that are catered to euclidean space which treats the 2 dimensional plane of real numbers. In the SFML library the top left is 0 and everything else is positive. As mentioned prior, this was overcome with a mapping into euclidean space instead of trying to reconfigure the trigonometric functions for the strange orientation.

### 6.4 Codebase

#### Makefile

This Makefile has no Linting as the program does not have any lints.

```
1
2 CXX = g++
3 CXXFLAGS = -std=c++11 -Wall -Wextra -Wpedantic
4 LDFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
5
6 all: ptree
7
8 ptree: main.o PTree.o
9     $(CXX) $(CXXFLAGS) -o $@ $^ $(LDFLAGS)
10
11 main.o: main.cpp PTree.h
12     $(CXX) $(CXXFLAGS) -c -o $@ $<
13
14 PTree.o: PTree.cpp PTree.h
15     $(CXX) $(CXXFLAGS) -c -o $@ $<
16
17 lint:
18     cpplint.py --filter=-runtime/references,-build/c++11 --root=. *
19 clean:
20     rm -f *.o ptree
```

```
21
22 .PHONY: all clean
```

### main.cpp

```
1 // Copyright 2023 Hunter Hasenfus and Daniel Olen
2 #include <iostream>
3 #include <SFML/Graphics.hpp>
4 #include "PTree.h"
5
6 int main(int argc, char* argv[]) {
7     if (argc < 4) {
8         std::cerr << "Usage: " << argv[0] << " L N Theta" << std::endl;
9         return 1;
10    }
11
12    // Parse command-line arguments for the
13    // base square size (L) and recursion depth (N)
14    double L = std::stod(argv[1]);
15    int N = std::stoi(argv[2]);
16    double startingAngle = std::stoi(argv[3]);
17
18    // Create a window with dimensions 6L x 4L
19    sf::RenderWindow window1(sf::VideoMode(6 * L, 4 * L), "Pythagoras Tree")
20    ;
21    window1.setFramerateLimit(60);
22
23    PTree tree1(L, N, 45);
24    PTree tree2(L, N, startingAngle);
25
26    while (window1.isOpen()) {
27        sf::Event event;
28        while (window1.pollEvent(event)) {
29            if (event.type == sf::Event::Closed) {
30                window1.close();
31            }
32        }
33
34        window1.clear();
35        window1.draw(tree1);
36        window1.display();
37    }
38    sf::RenderWindow window2(sf::VideoMode(10 * L, 4 * L), "Pythagoras Tree"
39    );
40    window2.setFramerateLimit(60);
41
42    while (window2.isOpen()) {
43        sf::Event event;
44        while (window2.pollEvent(event)) {
45            if (event.type == sf::Event::Closed) {
46                window2.close();
47            }
48        }
49
50        window2.clear();
51        window2.draw(tree2);
52        window2.display();
53    }
54
55    return 0;
56 }
```

## PTree.h

```
1 // Copyright 2023 Hunter Hasenfus and Daniel Olen
2 #ifndef PTREE_H_
3 #define PTREE_H_
4
5 #include <vector>
6 #include <SFML/Graphics.hpp>
7
8
9 class PTree : public sf::Drawable {
10 public:
11     // Constructor with two arguments, takes in
12     // the length of a rectangle and the number of iterations
13     PTree(double L, int N, double startingAngle);
14
15 private:
16     // Override the draw function from
17     // sf::Drawable to draw the rectangles in the window
18     virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
19         const;
20
21     // Recursive function to generate the Pythagoras Tree
22     void pTree(double L, int N, double x, double y,
23         double angle, double startingAngle);
24     // Vector to store the rectangles that make up the tree
25     std::vector<sf::RectangleShape> m_rectangles;
26
27     double logColor;
28     double linColor;
29 };
30 #endif // PTREE_H_
```

## PTree.cpp

```
1 // Copyright 2023 Hunter Hasenfus and Daniel Olen
2 #include "PTree.h"
3 #include <cmath>
4
5
6
7 // Constant value to convert degrees to radians
8 const double DEG_TO_RAD = M_PI / 180.0;
9
10 PTree::PTree(double L, int N, double startingAngle) {
11     logColor = pow(255.0, (1.0/N));
12     linColor = 255.0/N;
13     pTree(L, N, 3 * L - L/2, 3 * L, 0, startingAngle);
14 }
15
16 void PTree::draw(sf::RenderTarget& target, sf::RenderStates states) const {
17     // Iterates through the vector of rectangles
18     // and draws each rectangle on the target
19     for (const auto& rect : m_rectangles) {
20         target.draw(rect, states);
21     }
22 }
23
24 // Recursive function that generates
```

```

25 // the Pythagoras Tree
26 void PTree::pTree(double L, int N, double x,
27 double y, double angle, double startingAngle) {
28     // Base case: if N is 0, return from the function
29     if (N == 0) {
30         return;
31     }
32     // Create a rectangle with size L x L
33     sf::RectangleShape rect(sf::Vector2f(L, L));
34     rect.setPosition(x, y);
35     rect.setRotation(-1 * angle);
36     rect.setFillColor(sf::Color(255 - pow(logColor,
37 N), pow(logColor, N), linColor * N));
38     rect.setOutlineColor(sf::Color::Black);
39     rect.setOutlineThickness(1.0f);
40
41     // Add the rectangle to the vector of rectangles
42     m_rectangles.push_back(rect);
43
44     // Convert the angle from degrees to radians
45     double angleRad = startingAngle * DEG_TO_RAD;
46
47     // Calculate the new length of the rectangles for the next iteration
48     double newL1 = L * sin(angleRad);
49     double newL2 = L * cos(angleRad);
50
51     // Convert the angle for the second rectangle from degrees to radians
52     double angle2Rad = (angle + startingAngle) * DEG_TO_RAD;
53
54     // Calculate the new x and y coordinates for
55     // the first rectangle in the next iteration
56     double dx = (newL1 + newL2) * cos(angle2Rad);
57     double dy = (newL1 + newL2) * sin(angle2Rad);
58
59     double newX1, newY1, newX2, newY2;
60     if (dx >= 0 && dy >= 0) {
61         newX1 = x + abs(dx);
62         newY1 = y - abs(dy);
63     } else { if (dx <= 0 && dy <= 0) {
64         newX1 = x - abs(dx);
65         newY1 = y + abs(dy);
66     } else { if (dx <= 0 && dy >= 0) {
67         newX1 = x - abs(dx);
68         newY1 = y - abs(dy);
69     } else { if (dx >= 0 && dy <= 0) {
70         newX1 = x + abs(dx);
71         newY1 = y + abs(dy);
72     }}}
73
74     angle2Rad = (angle + startingAngle + 90) * DEG_TO_RAD;
75
76     // Calculate the new x and y coordinates
77     // for the second rectangle in the next iteration
78     dx = newL2 * cos(angle2Rad);
79     dy = newL2 * sin(angle2Rad);
80     if (dx >= 0 && dy >= 0) {
81         newX2 = x + abs(dx);
82         newY2 = y - abs(dy);
83     } else {if (dx <= 0 && dy <= 0) {

```

```

84     newX2 = x - abs(dx);
85     newY2 = y + abs(dy);
86 } else {if (dx <= 0 && dy >= 0) {
87     newX2 = x - abs(dx);
88     newY2 = y - abs(dy);
89 } else {if (dx >= 0 && dy <= 0) {
90     newX2 = x + abs(dx);
91     newY2 = y + abs(dy);
92 }}}
93
94 // Recursively call the pTree function for
95 // the first rectangle with the new parameters
96 pTree(newL1, N - 1, newX1, newY1,
97 angle - (90 - startingAngle), startingAngle);
98 pTree(newL2, N - 1, newX2, newY2, angle + startingAngle, startingAngle);
99 }

```

## 6.5 Output:

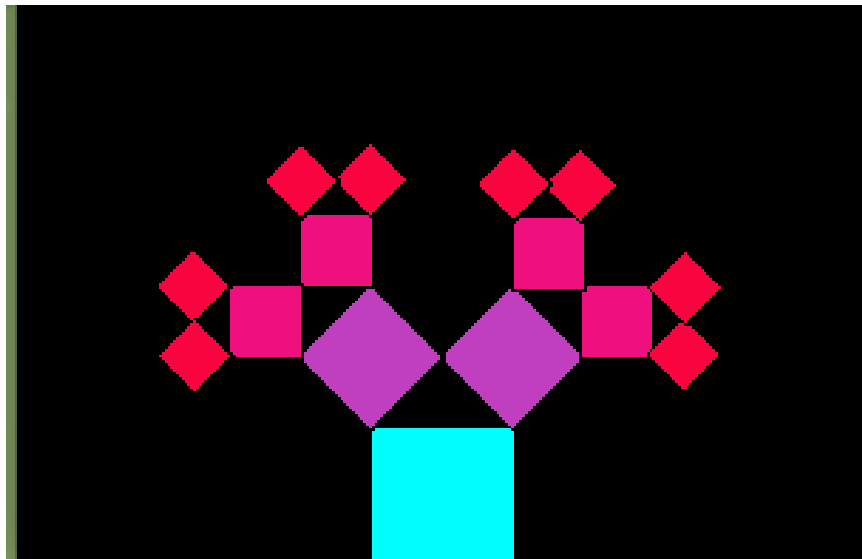


Figure 7: PS3 Output Window

## 7 PS4a: Checkers Visual mechanics

### 7.1 Discussion:

The assignment is tasked with creating a game using the SFML library that mimics the game of checkers. The game of checkers is inherently designed in the form of an array, which makes the functionality of it very easy to design and implement. By using indices as locations the area of clicking as well the entire drawing process can be circumvented into a simple fashion.

### 7.2 Key algorithms, Data structures and OO Designs used in this Assignment:

The key data structures that I implemented was a map of pairs. This allowed me to have all the pieces designated to a given user in a single data structure, and made it very easy to decipher what locations were valid when a given user was up. It also allowed me to figure out where a piece was then easily decipher its type.

I did not use smart pointers. I created an array of the game board, and then 2 map containers for the pieces relative to each user. For selecting the piece based on the button pressed and mouse location, I made a variable that was a member of that class. This allowed a dynamic nature to the 'selected variable'.

### 7.3 What I learned :

I learned how to decipher spatial locations within the SFML window. This was an interesting process and one that I can completely understand can take up a great deal of time for game creators.

### 7.4 Codebase

#### Makefile:

This Makefile has linting included.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
4 DEP = Checkers.h
5 OBJS = main.o Checkers.o
6
7 all: Checkers
8
9 Checkers: $(OBJS)
10    $(CC) -o Checkers $(OBJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12    $(CC) -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
14 lint:
15    cpplint.py --filter=--runtime/references,-build/c++11 --root=. *
16
17
18 cleanall : cleanobj
19    rm Checkers
20
21 cleanobj :
22    rm *.o
```

#### main.cpp:

```
1 // Copyright 2023 Hunter Hasenfus
2 #include <cstdlib>
3 #include <ctime>
4 #include <iostream>
```

```

5  #include <fstream>
6  #include <vector>
7  #include <SFML/System.hpp>
8  #include <SFML/Window.hpp>
9  #include <SFML/Graphics.hpp>
10 #include "Checkers.h"
11
12 int main(int arg, char* argc[]) {
13     bool replay = false;
14     do {
15         Checkers s;
16
17         sf::RenderWindow window(sf::VideoMode(8 * 64, 8 * 64), argc[0]);
18         while (window.isOpen()) {
19             sf::Event event;
20             while (window.pollEvent(event)) {
21                 if (event.type == sf::Event::Closed)
22                     window.close();
23                 if (event.type == sf::Event::MouseButtonPressed
24                     && sf::Mouse::Left == event.mouseButton.button) {
25                     // sf::Vector2i localPosition =
26                     // sf::Mouse::getPosition(window);
27                     // s.recognizePiece(localPosition.x, localPosition.y);
28                     s.recognizePiece(event.mouseButton.x, event.mouseButton.
29                                     y);
30                     // cout << event.mouseButton.x << "," <<
31                     // event.mouseButton.y << endl;
32                 }
33                 window.clear();
34                 window.draw(s);
35                 window.display();
36             }
37             std::cout << "Would you like to play again? (y/n)";
38             std::cin >> replay;
39             if (replay == 'y')
40                 replay = true;
41             else
42                 replay = false;
43         } while (replay);
44     }
45     return 0;
46 }

```

#### Checkers.h:

```

1  // Copyright 2023 Hunter Hasenfus
2  #ifndef CHECKERS_H_
3  #define CHECKERS_H_
4  #include <cstdlib>
5  #include <ctime>
6  #include <iostream>
7  #include <vector>
8  #include <map>
9  #include <string>
10 #include <utility>
11 #include <SFML/System.hpp>
12 #include <SFML/Window.hpp>
13 #include <SFML/Graphics.hpp>
14
15 class Checkers : public sf::Drawable {
16 public:

```



```

17     Checkers();
18     ~Checkers();
19     void recognizePiece(int x, int y);
20 private:
21
22     virtual void draw(sf::RenderTarget& target,
23     sf::RenderStates states) const;
24     char **board;
25
26     bool turn;
27
28     std::vector<std::string> colors = {"red", "black", "white"};
29     std::string p1color;
30     std::string p2color;
31
32     sf::Texture blackPawn;
33     sf::Texture redPawn;
34     sf::Texture blackKing;
35     sf::Texture redKing;
36     sf::Texture whitePawn;
37     sf::Texture whiteKing;
38     sf::Texture blackBoard;
39     sf::Texture redBoard;
40
41     sf::Texture arrow;
42
43     sf::Texture p1pawn;
44     sf::Texture p1king;
45     sf::Texture p2pawn;
46     sf::Texture p2king;
47
48     std::pair<std::pair<int, int>, std::pair<char, char>> selectedPiece;
49
50     std::vector<std::pair<int, int>> p1pawns;
51     std::vector<std::pair<int, int>> p1kings;
52
53     std::vector<std::pair<int, int>> p2pawns;
54     std::vector<std::pair<int, int>> p2kings;
55
56     std::map<std::pair<int, int>, char> p1pieces;
57     std::map<std::pair<int, int>, char> p2pieces;
58 };
59
60 #endif // CHECKERS_H_

```

#### Checkers.cpp:

```

1 // Copyright 2023 Hunter Hasenfus
2 #include "Checkers.h"
3 #include <cstdlib>
4 #include <ctime>
5 #include <iostream>
6 #include <fstream>
7 #include <utility>
8 #include <string>
9 #include <algorithm>
10 #include <vector>
11 #include <SFML/System.hpp>
12 #include <SFML/Window.hpp>
13 #include <SFML/Graphics.hpp>
14

```

```

15
16 void Checkers::recognizePiece(int x, int y) {
17     double n, m;
18     if (x > 36 && x < 36 + 55 * 8 && y > 36 && y < 36 + 55 * 8) {
19         m = (x-36) / 55;
20         n = (y-36) / 55;
21         if (turn) {
22             auto Piece = find_if(p1pieces.begin(), p1pieces.end(),
23                 [n, m](std::pair<std::pair<int, int>
24                     , char> x){return x.first.first == n && x.first.second == m;});
25             if (Piece != p1pieces.end())
26                 selectedPiece = make_pair(Piece->first,
27                     std::make_pair(Piece->second, '1'));
28             else
29                 selectedPiece = std::make_pair(std::make_pair(-1, -1),
30                     std::make_pair('0', '0'));
31         } else {
32             auto Piece = find_if(p2pieces.begin(), p2pieces.end(),
33                 [n, m](std::pair<std::pair<int, int>
34                     , char> x){return x.first.first == n && x.first.second == m;});
35             if (Piece != p2pieces.end())
36                 selectedPiece = make_pair(Piece->first,
37                     std::make_pair(Piece->second, '2'));
38             else
39                 selectedPiece = std::make_pair(std::make_pair(-1, -1),
40                     std::make_pair('0', '0'));
41         } else {
42             selectedPiece = std::make_pair(std::make_pair(-1, -1),
43                 std::make_pair('0', '0'));
44         }
45         // cout << n << "," << m << endl;
46         // cout << selectedPiece.first.first << "," <<
47         // selectedPiece.first.second << " "
48         // << selectedPiece.second.first << " " <<
49         // selectedPiece.second.second << endl;
50     }
51 }
52 Checkers::~~Checkers() {
53     for (int i = 0; i < 8; i++) {
54         delete[] board[i];
55     }
56     delete[] board;
57 }
58
59
60 Checkers::Checkers() {
61     blackPawn.loadFromFile("checkers/blackpawn.png");
62     redPawn.loadFromFile("checkers/redpawn.png");
63     blackKing.loadFromFile("checkers/blackking.png");
64     redKing.loadFromFile("checkers/redking.png");
65     whitePawn.loadFromFile("checkers/whitepawn.png");
66     whiteKing.loadFromFile("checkers/whiteking.png");
67     arrow.loadFromFile("checkers/download.png");
68
69
70     turn = true;
71     selectedPiece = std::make_pair(std::make_pair(-1, -1),
72         std::make_pair('0', '0'));
73

```

```

74     std::vector<std::string>::iterator y;
75     std::cout << "Choose a color for player 1 (red, black, or white)";
76     do {
77         std::cin >> p1color;
78         y = find_if(colors.begin(), colors.end(), [this](std::string x)
79             {return x == this->p1color;});
80     } while (y == colors.end());
81
82     if (p1color == "black") {
83         p1pawn = blackPawn;
84         p1king = blackKing;
85     } else { if (p1color == "red") {
86         p1pawn = redPawn;
87         p1king = redKing;
88     } else {if (p1color == "white") {
89         p1pawn = whitePawn;
90         p1king = whiteKing;
91     }}}
92     colors.erase(y);
93
94     std::cout << "Choose a color for player 2 (" << colors[0] << " or "
95         << colors[1] << ")";
96     do {
97         std::cin >> p2color;
98     } while (find_if(colors.begin(), colors.end(), [this](std::string x)
99         {return x == this->p2color;}) == colors.end());
100
101     if (p2color == "black") {
102         p2pawn = blackPawn;
103         p2king = blackKing;
104     } else {if (p2color == "red") {
105         p2pawn = redPawn;
106         p2king = redKing;
107     } else {if (p2color == "white") {
108         p2pawn = whitePawn;
109         p2king = whiteKing;
110     }}}
111
112     board = new char*[8];
113     for (int i = 0; i < 8; i++) {
114         board[i] = new char[8];
115         for (int j = 0; j < 8; j++) {
116             if (i % 2) {
117                 if (j % 2) {
118                     board[i][j] = 'r';
119                 } else {
120                     board[i][j] = 'b';
121                     if (i < 3) {
122                         p1pieces.insert(std::make_pair(
123                             std::make_pair(i, j), 'p'));
124                     } else if (i > 4) {
125                         p2pieces.insert(std::make_pair(
126                             std::make_pair(i, j), 'p'));
127                     } else {
128                         if (j % 2) {
129                             board[i][j] = 'b';
130                             if (i < 3) {
131                                 p1pieces.insert(std::make_pair(
132                                     std::make_pair(i, j), 'p'));

```

```

133         } else {if (i > 4) {
134             p2pieces.insert(std::make_pair(
135                 std::make_pair(i, j), 'p'));
136         }} else {
137             board[i][j] = 'r';
138         }}}
139 // cout << "p1pawns:" << endl;
140 // for(auto x: p1pieces) {
141 //     std::cout << x.first.first << " " << x.first.second << " - ";
142 // }
143 // cout << endl << "p2pawns:" << endl;
144 // for(auto x: p2pieces) {
145 //     std::cout << x.first.first << " " << x.first.second << " - ";
146 // }
147 }
148
149 void Checkers::draw(sf::RenderTarget& target, sf::RenderStates states) const
150 {
151     for (int i = 0; i < 3; i++) {
152         sf::RectangleShape border(sf::Vector2f(8 * (64 - i * 4),
153             8 * (64 - i * 4)));
154         border.setPosition(i * 16, i * 16);
155         border.setFillColor(sf::Color(175 + i * 25, 175 + i * 25,
156             100 + i * 25));
157         border.setOutlineColor(sf::Color::Black);
158         border.setOutlineThickness(0.5f);
159         target.draw(border);
160     }
161     for (int i = 0; i < 8; i++) {
162         for (int j = 0; j < 8; j++) {
163             sf::RectangleShape rect(sf::Vector2f(55, 55));
164             rect.setPosition(36 + i * 55, 36 + j * 55);
165
166             rect.setOutlineColor(sf::Color::Black);
167             rect.setOutlineThickness(0.1f);
168             sf::Sprite sprite;
169             switch (board[i][j]) {
170                 case 'b':
171                     rect.setFillColor(sf::Color::Black);
172                     break;
173                 case 'r':
174                     rect.setFillColor(sf::Color::Red);
175                     break;
176             }
177             target.draw(rect);
178         }
179     }
180
181     for (auto x : p1pieces) {
182         sf::Sprite sprite;
183         if (x.second == 'p')
184             sprite.setTexture(p1pawn);
185         else
186             sprite.setTexture(p1king);
187         sprite.scale(55.0/p1pawn.getSize().x, 55.0/p1pawn.getSize().y);
188         sprite.setPosition(36 + x.first.second * 55, 36 + x.first.first *
189             55);
190         target.draw(sprite);

```

```

190     }
191
192     for (auto x : p2pieces) {
193         sf::Sprite sprite;
194         if (x.second == 'p')
195             sprite.setTexture(p2pawn);
196         else
197             sprite.setTexture(p2king);
198         sprite.scale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y);
199         sprite.setPosition(36 + x.first.second * 55, 36 + x.first.first *
200 55);
201         target.draw(sprite);
202     }
203
204     sf::Sprite sprite;
205     if (selectedPiece.second.second != '0') {
206         if (selectedPiece.second.second == '1') {
207             if (selectedPiece.second.first == 'p')
208                 sprite.setTexture(p1pawn);
209             else
210                 sprite.setTexture(p1king);
211         } else { if (selectedPiece.second.second == '2') {
212             if (selectedPiece.second.first == 'p')
213                 sprite.setTexture(p2pawn);
214             else
215                 sprite.setTexture(p2king);
216         }}
217
218         // sprite.setScale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y)
219         ;
220         // cout << "SCALE: " << sprite.getScale().x << ", "
221         // << sprite.getScale().y << endl;
222         // sprite.setColor(sf::Color::Yellow);
223         // cout << "COLOR: " << sprite.getScale().x << ", "
224         // << sprite.getScale().y << endl;
225
226         // cout << "texture size: " << p2pawn.getSize().x << ", "
227         // << p2pawn.getSize().y << endl;
228         // cout << "intendend scale: " << 55.0/p2pawn.getSize().x
229         // << ", " << 55.0/p2pawn.getSize().y << endl;
230         sprite.setColor(sf::Color(0, 255, 0));
231         // cout << "COLOR: " << sprite.getScale().x << ", " <<
232         // sprite.getScale().y << endl;
233         sprite.setScale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y);
234         // cout << "SCALE: " << sprite.getScale().x << ", " <<
235         // sprite.getScale().y << endl;
236
237         // sprite.scale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y);
238         // cout << sprite.getScale().x << ", " << sprite.getScale().y << endl
239         ;
240         sprite.setPosition(36 + selectedPiece.first.second * 55,
241 36 + selectedPiece.first.first * 55);
242         target.draw(sprite);
243     }
244
245     if (turn) {
246         sf::Sprite sprite;
247         sprite.setTexture(arrow);
248         sprite.setOrigin(arrow.getSize().x/2, arrow.getSize().y/2);

```

```

246     sprite.scale(36.0/arrow.getSize().x, 36.0/arrow.getSize().y);
247     sprite.setPosition(18, 4 * 64);
248     sprite.setRotation(0);
249     target.draw(sprite);
250 } else {
251     sf::Sprite sprite;
252     sprite.setTexture(arrow);
253     sprite.setOrigin(arrow.getSize().x/2, arrow.getSize().y/2);
254     sprite.scale(36.0/arrow.getSize().x, 36.0/arrow.getSize().y);
255     sprite.setPosition(18, 4 * 64);
256     sprite.setRotation(180);
257     target.draw(sprite);
258 }
259
260 // if(won)
261 // {
262 //     sf::Sprite sprite;
263 //     sprite.setTexture(Win);
264 //     sprite.scale(64.0/Win.getSize().x, 64.0/Win.getSize().y);
265 //     sprite.setPosition(playerLocation.second * 64,
266 //     playerLocation.first * 64);
267 //     target.draw(sprite);
268 // }
269 }

```

## 7.5 Output :

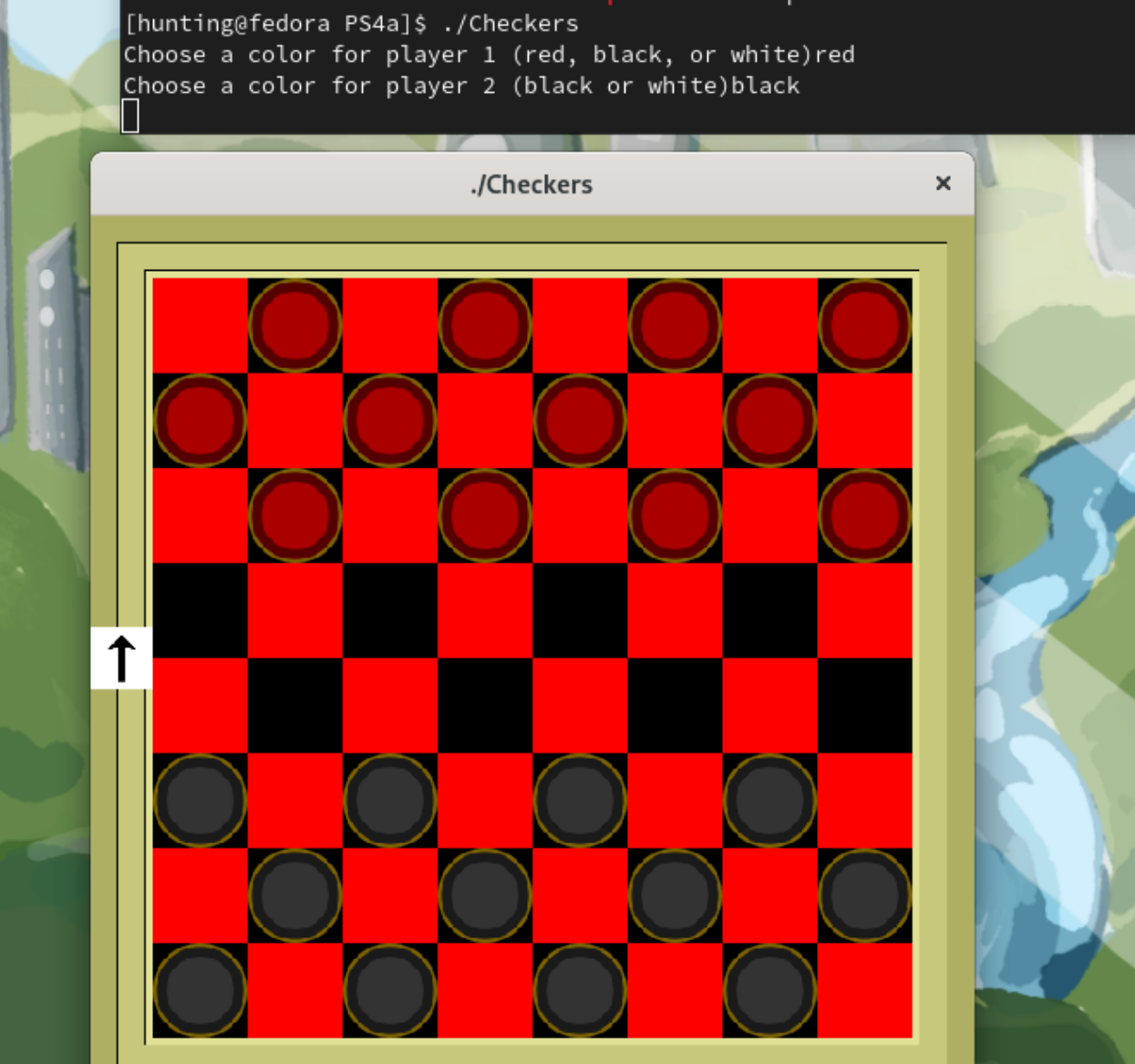


Figure 8: PS4a Output in Terminal

## 8 PS4b: Checkers Game mechanics

### 8.1 Discussion:

The assignment is the task of applying functionality to the UI that was developed in the previous assignment. This assignment is where the movement of the players is conditioned as well as the crowning of pawns and the decision of victory.

### 8.2 Key algorithms, Data structures and OO Designs used in this Assignment:

The key algorithms I used were maps, this allowed myself the ability to track various pieces based on certain mappings. It made it very easy and transferable to coordinate functions based on the mappings of different different groups. The object oriented nature allowed myself to store various data structures that made it indispensable when coordinating all the functions together.

### 8.3 What I learned :

I created a few features and actually made and remade aspects as I went on. I noticed smoother and more versatile methods by which I could implement the game and noticed that traveling onward with legacy structures would have brought along handicaps that could have made the development much harder. The key points of this were abstracting away the nextMove functionality. This allowed myself the capacity to check if any piece had any next moves which aided in the implementation of both multijumps as well as endgame scenarios.

### 8.4 Codebase

#### Makefile:

This Makefile has linting included.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
4 DEP = Checkers.h
5 OBJS = main.o Checkers.o
6
7 all: Checkers
8
9 Checkers: $(OBJS)
10     $(CC) -o Checkers $(OBJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12     $(CC) -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
14 lint:
15     cpplint.py --filter=-runtime/references,-build/c++11 --root=. *
16
17
18 cleanall : cleanobj
19     rm Checkers
20
21 cleanobj :
22     rm *.o
```

#### main.cpp:

```
1 // Copyright 2023 Hunter Hasenfus
2 #include <cstdlib>
3 #include <ctime>
4 #include <iostream>
5 #include <fstream>
```



```

6  #include <vector>
7  #include <SFML/System.hpp>
8  #include <SFML/Window.hpp>
9  #include <SFML/Graphics.hpp>
10 #include "Checkers.h"
11
12 int main(int arg, char* argc[]) {
13     bool replay = false;
14     do {
15         Checkers s;
16
17         sf::RenderWindow window(sf::VideoMode(8 * 64, 8 * 64), argc[0]);
18         while (window.isOpen()) {
19             sf::Event event;
20             while (window.pollEvent(event)) {
21                 if (event.type == sf::Event::Closed)
22                     window.close();
23                 if (event.type == sf::Event::MouseButtonPressed
24                     && sf::Mouse::Left == event.mouseButton.button) {
25                     if (s.pieceSelected())
26                         s.checkMove(event.mouseButton.x, event.mouseButton.y);
27
28                     s.recognizePiece(event.mouseButton.x, event.mouseButton.y);
29
30                     window.clear();
31                     window.draw(s);
32                     window.display();
33                 }
34                 std::cout << "Would you like to play again? (y/n)";
35                 std::cin >> replay;
36                 if (replay == 'y')
37                     replay = true;
38                 else
39                     replay = false;
40             } while (replay);
41
42             return 0;
43     }

```

#### Checkers.h:

```

1  // Copyright 2023 Hunter Hasenfus
2  #ifndef CHECKERS_H_
3  #define CHECKERS_H_
4  #include <cstdlib>
5  #include <ctime>
6  #include <iostream>
7  #include <vector>
8  #include <map>
9  #include <string>
10 #include <utility>
11 #include <SFML/System.hpp>
12 #include <SFML/Window.hpp>
13 #include <SFML/Graphics.hpp>
14
15 class Checkers : public sf::Drawable {
16 public:
17     Checkers();
18     ~Checkers();
19     void pawn2King(std::map<std::pair<int, int>, char> &pieces,

```

```

20     std::pair<int, int> pos);
21 void recognizePiece(int x, int y);
22 bool pieceSelected();
23 void checkMove(int x, int y);
24 void nextMoves(std::pair<std::pair<int, int>, std::pair<char,
25     char>> &piece, std::map<std::pair<int, int>, char> &Moves);
26
27 private:
28     void movePiece(std::map<std::pair<int, int>, char> &pieces,
29         std::pair<int, int> newPos);
30     virtual void draw(sf::RenderTarget& target,
31         sf::RenderStates states) const;
32     char **board;
33
34     bool turn;
35
36     std::vector<std::string> colors = {"red", "black", "white"};
37     std::string p1color;
38     std::string p2color;
39
40     sf::Texture blackPawn;
41     sf::Texture redPawn;
42     sf::Texture blackKing;
43     sf::Texture redKing;
44     sf::Texture whitePawn;
45     sf::Texture whiteKing;
46     sf::Texture blackBoard;
47     sf::Texture redBoard;
48
49     sf::Texture arrow;
50
51     sf::Texture p1pawn;
52     sf::Texture p1king;
53     sf::Texture p2pawn;
54     sf::Texture p2king;
55     sf::Font loserFont;
56
57     std::map<std::pair<int, int>, char> potMoves;
58
59     std::pair<std::pair<int, int>, std::pair<char, char>> selectedPiece;
60
61     // std::vector<std::pair<int, int>> p1pawns;
62     // std::vector<std::pair<int, int>> p1kings;
63
64     // std::vector<std::pair<int, int>> p2pawns;
65     // std::vector<std::pair<int, int>> p2kings;
66
67     std::map<std::pair<int, int>, char> p1pieces;
68     std::map<std::pair<int, int>, char> p2pieces;
69
70     bool p1Win;
71     bool p2Win;
72 };
73
74 #endif // CHECKERS_H_

```

## Checkers.cpp:

```
1 // Copyright 2023 Hunter Hasenfus
2 #include "Checkers.h"
3 #include <cstdlib>
4 #include <ctime>
5 #include <iostream>
6 #include <fstream>
7 #include <utility>
8 #include <string>
9 #include <algorithm>
10 #include <vector>
11 #include <SFML/System.hpp>
12 #include <SFML/Window.hpp>
13 #include <SFML/Graphics.hpp>
14
15 void Checkers::pawn2King(std::map<std::pair<int, int>, char> &pieces,
16     std::pair<int, int> pos) {
17     auto x = pieces.find(pos);
18     pieces.erase(x);
19     pieces.insert(std::make_pair(pos, 'k'));
20 }
21
22 void Checkers::movePiece(std::map<std::pair<int, int>, char> &pieces,
23     std::pair<int, int> newPos) {
24     pieces.erase(selectedPiece.first);
25     pieces.insert(std::make_pair(std::make_pair(newPos.first, newPos.second)
26     ,
27     selectedPiece.second.first));
28     selectedPiece = std::make_pair(std::make_pair(-1, -1),
29     std::make_pair('0', '0'));
30 }
31
32 void Checkers::nextMoves(std::pair<std::pair<int, int>, std::pair<char,
33     char>> &piece, std::map<std::pair<int, int>, char> &Moves) {
34     int m, n;
35     Moves.clear();
36     if (piece.second.second == '1') {
37         if (piece.second.first == 'p') {
38             // if click is in a valid close square
39             for (int j = -1; j < 2; j += 2) {
40                 n = piece.first.first + 1;
41                 m = piece.first.second + j;
42                 if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7) && (
43                 p2pieces.find(std::make_pair(n, m)) ==
44                 p2pieces.end()) && (p1pieces.find(std::make_pair(n, m))
45                 == p1pieces.end())) {
46                     Moves.insert(std::make_pair(std::make_pair(n, m), 'r
47                     '));
48                 }
49                 auto x = p2pieces.find(std::make_pair(n, m));
50                 if (x != p2pieces.end()) {
51                     n = piece.first.first + 2;
52                     m = piece.first.second + j * 2;
53                     if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7) && (
54                     p2pieces.find(std::make_pair(n, m)) ==
55                     p2pieces.end()) && (p1pieces.find(std::make_pair(n,
56                     m))
57                     == p1pieces.end())) {
58                         Moves.insert(std::make_pair(std::make_pair(n,
```

```

54         m), 'j'));
55     }}}}
56     if (piece.second.first == 'k') {
57         for (int i = -1; i < 2; i += 2) {
58             for (int j = -1; j < 2; j += 2) {
59                 n = piece.first.first + i;
60                 m = piece.first.second + j;
61                 if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7) && (
p2pieces.find(std::make_pair(n, m)) ==
62                     p2pieces.end()) && (p1pieces.find(std::make_pair(n,
m))
63                         == p1pieces.end())) {
64                     Moves.insert(std::make_pair(std::make_pair(n,
65 m), 'r'));
66                 }
67                 auto x = p2pieces.find(std::make_pair(n, m));
68                 if (x != p2pieces.end()) {
69                     n = piece.first.first + i * 2;
70                     m = piece.first.second + j * 2;
71                     if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7) &&
(p2pieces.find(std::make_pair(n, m)) ==
72                         p2pieces.end()) && (p1pieces.find(
73                             std::make_pair(n, m))
74                             == p1pieces.end())) {
75                         Moves.insert(std::make_pair(
76                             std::make_pair(n, m), 'j'));
77 }}}}}} else { if (piece.second.second == '2') {
78     if (piece.second.first == 'p') {
79         // if click is in a valid close square
80         for (int j = -1; j < 2; j += 2) {
81             n = piece.first.first - 1;
82             m = piece.first.second + j;
83             if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7) && (
p2pieces.find(std::make_pair(n, m)) ==
84                 p2pieces.end()) && (p1pieces.find(std::make_pair(n,
m))
85                     == p1pieces.end())) {
86                 Moves.insert(std::make_pair(
87                     std::make_pair(n, m), 'r'));
88             }
89             auto x = p1pieces.find(std::make_pair(n, m));
90             if (x != p1pieces.end()) {
91                 n = piece.first.first - 2;
92                 m = piece.first.second + j * 2;
93                 if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7) &&
(p2pieces.find(std::make_pair(n, m))
94                     == p2pieces.end()) && (p1pieces.find(
95                         std::make_pair(n, m))
96                         == p1pieces.end())) {
97                     Moves.insert(std::make_pair(
98                         std::make_pair(n, m), 'j'));
99                 }}}}
100     if (piece.second.first == 'k') {
101         for (int i = -1; i < 2; i += 2) {
102             for (int j = -1; j < 2; j += 2) {
103                 n = piece.first.first + i;
104                 m = piece.first.second + j;
105                 if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7) &&
(p2pieces.find(std::make_pair(n, m)) ==

```

```

106         p2pieces.end()) && (p1pieces.find(
107             std::make_pair(n, m))
108             == p1pieces.end())) {
109             Moves.insert(std::make_pair(
110                 std::make_pair(n, m), 'r'));
111         }
112         std::cout << "here1";
113         auto x = p1pieces.find(std::make_pair(n, m));
114         if (x != p1pieces.end()) {
115             std::cout << "here2";
116             n = piece.first.first + i * 2;
117             m = piece.first.second + j * 2;
118             if ((n >= 0 && n <= 7) && (m >= 0 && m <= 7)
119                 && (p2pieces.find(std::make_pair(n, m))
120                     == p2pieces.end()) && (p1pieces.find(
121                         std::make_pair(n, m)) == p1pieces.end())
122                 ) {
123                 std::cout << "here3";
124                 Moves.insert(std::make_pair(
125                     std::make_pair(n, m), 'j'));
126             }
127         }
128     }
129 }
130
131 void Checkers::checkMove(int x, int y) {
132     double n, m;
133     if (x > 36 && x < 36 + 55 * 8 && y > 36 && y < 36 + 55 * 8) {
134         m = (x-36) / 55;
135         n = (y-36) / 55;
136         // if p1s turn
137         for (auto moves : potMoves) {
138             if (moves.first.first == n && moves.first.second == m) {
139                 if (moves.second == 'r') {
140                     if (turn) {
141                         movePiece(p1pieces, std::make_pair(n, m));
142                     } else {
143                         movePiece(p2pieces, std::make_pair(n, m));
144                     }
145                     turn = !turn;
146                     break;
147                 }
148             }
149             if (moves.second == 'j') {
150                 if (turn) {
151                     auto temp = std::make_pair(selectedPiece.first.first
152                         - (selectedPiece.first.first - n)/2,
153                         selectedPiece.first.second - (
154                             selectedPiece.first.second - m)/2);
155                     auto newPiece = std::make_pair(std::make_pair(n, m),
156                         std::make_pair(selectedPiece.second.first, '1'));
157                     movePiece(p1pieces, std::make_pair(n, m));
158                     p2pieces.erase(temp);
159                     selectedPiece = newPiece;
160                     nextMoves(selectedPiece, potMoves);
161                     auto temp2 = find_if(potMoves.begin(), potMoves.end

```

```

162         selectedPiece = std::make_pair(
163             std::make_pair(-1, -1), std::make_pair(
164                 '0', '0'));
165     }
166     if (n == 7) {
167         pawn2King(p1pieces, std::make_pair(n, m));
168     } else {
169         auto temp = std::make_pair(selectedPiece.first.first
170             - (selectedPiece.first.first - n)/2,
171             selectedPiece.first.second - (
172                 selectedPiece.first.second - m)/2);
173         auto newPiece = std::make_pair(std::make_pair(n, m),
174             std::make_pair(selectedPiece.second.first, '2'));
175         movePiece(p2pieces, std::make_pair(n, m));
176         p1pieces.erase(temp);
177         selectedPiece = newPiece;
178         nextMoves(selectedPiece, potMoves);
179         auto temp2 = find_if(potMoves.begin(),
180             potMoves.end(), [](std::pair<std::pair<int, int>,
181                 char> piece) {
182                 return piece.second == 'j';
183             });
184         if (temp2 == potMoves.end()) {
185             turn = !turn;
186             selectedPiece = std::make_pair(std::make_pair(
187                 -1, -1), std::make_pair('0', '0'));
188         }
189         if (n == 0) {
190             pawn2King(p2pieces, std::make_pair(n, m));
191         }
192     }
193     break;
194 }}}
195     if (p1pieces.empty()) {
196         p1Win = true;
197     }
198     if (p2pieces.empty()) {
199         p2Win = true;
200     }
201     std::map<std::pair<int, int>, char> noMoves;
202     std::vector<bool> noMovesVector;
203     for (auto x : p1pieces) {
204         auto temp = std::make_pair(x.first, std::make_pair(x.second, '1'));
205         nextMoves(temp, noMoves);
206         if (noMoves.empty())
207             noMovesVector.push_back(true);
208     }
209     if (noMovesVector.empty())
210         p1Win = true;
211     noMovesVector.clear();
212     for (auto x : p2pieces) {
213         auto temp = std::make_pair(x.first, std::make_pair(x.second, '2'));
214         nextMoves(temp, noMoves);
215         if (noMoves.empty())
216             noMovesVector.push_back(true);
217     }
218     if (noMovesVector.empty())
219         p2Win = true;
220     noMovesVector.clear();

```

```

221 }}
222
223
224
225
226
227 bool Checkers::pieceSelected() {
228     return selectedPiece.second.second != '0';
229 }
230 void Checkers::recognizePiece(int x, int y) {
231     double n, m;
232     if (x > 36 && x < 36 + 55 * 8 && y > 36 && y < 36 + 55 * 8) {
233         m = (x-36) / 55;
234         n = (y-36) / 55;
235         if (turn) {
236             auto Piece = find_if(p1pieces.begin(), p1pieces.end(),
237                 [n, m](std::pair<std::pair<int, int>
238                     , char> x){return x.first.first == n && x.first.second == m;});
239             if (Piece != p1pieces.end())
240                 selectedPiece = make_pair(Piece->first,
241                     std::make_pair(Piece->second, '1'));
242             else
243                 selectedPiece = std::make_pair(std::make_pair(-1, -1),
244                     std::make_pair('0', '0'));
245         } else {
246             auto Piece = find_if(p2pieces.begin(), p2pieces.end(),
247                 [n, m](std::pair<std::pair<int, int>
248                     , char> x){return x.first.first == n && x.first.second == m;});
249             if (Piece != p2pieces.end())
250                 selectedPiece = make_pair(Piece->first,
251                     std::make_pair(Piece->second, '2'));
252             else
253                 selectedPiece = std::make_pair(std::make_pair(-1, -1),
254                     std::make_pair('0', '0'));
255         } else {
256             selectedPiece = std::make_pair(std::make_pair(-1, -1),
257                 std::make_pair('0', '0'));
258         }
259
260         potMoves.clear();
261         nextMoves(selectedPiece, potMoves);
262     }
263
264     Checkers::~~Checkers() {
265         for (int i = 0; i < 8; i++) {
266             delete[] board[i];
267         }
268         delete[] board;
269     }
270
271     Checkers::Checkers() {
272         blackPawn.loadFromFile("checkers/blackpawn.png");
273         redPawn.loadFromFile("checkers/redpawn.png");
274         blackKing.loadFromFile("checkers/blackking.png");
275         redKing.loadFromFile("checkers/redking.png");
276         whitePawn.loadFromFile("checkers/whitepawn.png");
277         whiteKing.loadFromFile("checkers/whiteking.png");
278         arrow.loadFromFile("checkers/download.png");
279

```



```

280     loserFont.loadFromFile("checkers/UbuntuMono-R.ttf");
281
282     p1Win = false;
283     p2Win = false;
284     turn = true;
285     selectedPiece = std::make_pair(std::make_pair(-1, -1),
286         std::make_pair('0', '0'));
287
288     std::vector<std::string>::iterator y;
289     std::cout << "Choose a color for player 1 (red, black, or white)";
290     do {
291         std::cin >> p1color;
292         y = find_if(colors.begin(), colors.end(), [this](std::string x)
293             {return x == this->p1color;});
294     } while (y == colors.end());
295
296     if (p1color == "black") {
297         p1pawn = blackPawn;
298         p1king = blackKing;
299     } else { if (p1color == "red") {
300         p1pawn = redPawn;
301         p1king = redKing;
302     } else {if (p1color == "white") {
303         p1pawn = whitePawn;
304         p1king = whiteKing;
305     }}}
306     colors.erase(y);
307
308     std::cout << "Choose a color for player 2 (" << colors[0] << " or "
309         << colors[1] << ")";
310     do {
311         std::cin >> p2color;
312     } while (find_if(colors.begin(), colors.end(), [this](std::string x)
313         {return x == this->p2color;}) == colors.end());
314
315     if (p2color == "black") {
316         p2pawn = blackPawn;
317         p2king = blackKing;
318     } else {if (p2color == "red") {
319         p2pawn = redPawn;
320         p2king = redKing;
321     } else {if (p2color == "white") {
322         p2pawn = whitePawn;
323         p2king = whiteKing;
324     }}}
325
326     board = new char*[8];
327     for (int i = 0; i < 8; i++) {
328         board[i] = new char[8];
329         for (int j = 0; j < 8; j++) {
330             if (i % 2) {
331                 if (j % 2) {
332                     board[i][j] = 'r';
333                 } else {
334                     board[i][j] = 'b';
335                     if (i < 3) {
336                         p1pieces.insert(std::make_pair(
337                             std::make_pair(i, j), 'p'));
338                     } else if (i > 4) {

```



```

339         p2pieces.insert(std::make_pair(
340             std::make_pair(i, j), 'p'));
341     }} else {
342     if (j % 2) {
343         board[i][j] = 'b';
344         if (i < 3) {
345             p1pieces.insert(std::make_pair(
346                 std::make_pair(i, j), 'p'));
347         } else {if (i > 4) {
348             p2pieces.insert(std::make_pair(
349                 std::make_pair(i, j), 'p'));
350         }} else {
351             board[i][j] = 'r';
352         }}}}}
353
354 void Checkers::draw(sf::RenderTarget& target, sf::RenderStates states) const
355 {
356     for (int i = 0; i < 3; i++) {
357         sf::RectangleShape border(sf::Vector2f(8 * (64 - i * 4),
358             8 * (64 - i * 4)));
359         border.setPosition(i * 16, i * 16);
360         border.setFillColor(sf::Color(175 + i * 25, 175 + i * 25,
361             100 + i * 25));
362         border.setOutlineColor(sf::Color::Black);
363         border.setOutlineThickness(0.5f);
364         target.draw(border);
365     }
366     for (int i = 0; i < 8; i++) {
367         for (int j = 0; j < 8; j++) {
368             sf::RectangleShape rect(sf::Vector2f(55, 55));
369             rect.setPosition(36 + i * 55, 36 + j * 55);
370
371             rect.setOutlineColor(sf::Color::Black);
372             rect.setOutlineThickness(0.1f);
373             sf::Sprite sprite;
374             switch (board[i][j]) {
375                 case 'b':
376                     rect.setFillColor(sf::Color::Black);
377                     break;
378                 case 'r':
379                     rect.setFillColor(sf::Color::Red);
380                     break;
381             }
382             target.draw(rect);
383         }
384     }
385
386     for (auto x : p1pieces) {
387         sf::Sprite sprite;
388         if (x.second == 'p')
389             sprite.setTexture(p1pawn);
390         else
391             sprite.setTexture(p1king);
392         sprite.scale(55.0/p1pawn.getSize().x, 55.0/p1pawn.getSize().y);
393         sprite.setPosition(36 + x.first.second * 55, 36 + x.first.first *
394             55);
395         target.draw(sprite);
396     }

```

```

396
397     for (auto x : p2pieces) {
398         sf::Sprite sprite;
399         if (x.second == 'p')
400             sprite.setTexture(p2pawn);
401         else
402             sprite.setTexture(p2king);
403         sprite.scale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y);
404         sprite.setPosition(36 + x.first.second * 55, 36 + x.first.first *
55);
405         target.draw(sprite);
406     }
407
408     sf::Sprite sprite;
409     if (selectedPiece.second.second != '0') {
410         if (selectedPiece.second.second == '1') {
411             if (selectedPiece.second.first == 'p')
412                 sprite.setTexture(p1pawn);
413             else
414                 sprite.setTexture(p1king);
415         } else { if (selectedPiece.second.second == '2') {
416             if (selectedPiece.second.first == 'p')
417                 sprite.setTexture(p2pawn);
418             else
419                 sprite.setTexture(p2king);
420         }}
421
422
423
424         // sprite.setScale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y)
;
425         // cout << "SCALE: " << sprite.getScale().x << ", "
426         // << sprite.getScale().y << endl;
427         // sprite.setColor(sf::Color::Yellow);
428         // cout << "COLOR: " << sprite.getScale().x << ", "
429         // << sprite.getScale().y << endl;
430
431         // cout << "texture size: " << p2pawn.getSize().x << ", "
432         // << p2pawn.getSize().y << endl;
433         // cout << "intendend scale: " << 55.0/p2pawn.getSize().x
434         // << ", " << 55.0/p2pawn.getSize().y << endl;
435         sprite.setColor(sf::Color(0, 255, 0));
436         // cout << "COLOR: " << sprite.getScale().x << ", " <<
437         // sprite.getScale().y << endl;
438         sprite.setScale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y);
439         // cout << "SCALE: " << sprite.getScale().x << ", " <<
440         // sprite.getScale().y << endl;
441
442         // sprite.scale(55.0/p2pawn.getSize().x, 55.0/p2pawn.getSize().y);
443         // cout << sprite.getScale().x << ", " << sprite.getScale().y << endl
;
444         sprite.setPosition(36 + selectedPiece.first.second * 55,
445         36 + selectedPiece.first.first * 55);
446         target.draw(sprite);
447     }
448
449     if (turn) {
450         sf::Sprite sprite;
451         sprite.setTexture(arrow);

```

```

452     sprite.setOrigin(arrow.getSize().x/2, arrow.getSize().y/2);
453     sprite.scale(36.0/arrow.getSize().x, 36.0/arrow.getSize().y);
454     sprite.setPosition(18, 4 * 64);
455     sprite.setRotation(0);
456     target.draw(sprite);
457 } else {
458     sf::Sprite sprite;
459     sprite.setTexture(arrow);
460     sprite.setOrigin(arrow.getSize().x/2, arrow.getSize().y/2);
461     sprite.scale(36.0/arrow.getSize().x, 36.0/arrow.getSize().y);
462     sprite.setPosition(18, 4 * 64);
463     sprite.setRotation(180);
464     target.draw(sprite);
465 }
466
467
468 if (p1Win) {
469     sf::Text text;
470     text.setFont(loserFont);
471     text.setPosition(0, 64 * 4);
472     text.setColor(sf::Color::White);
473     text.setStyle(sf::Text::Bold | sf::Text::Underlined);
474     text.setCharacterSize(64);
475     text.setString("Player 2 Wins!");
476     target.draw(text);
477 }
478 if (p2Win) {
479     sf::Text text;
480     text.setFont(loserFont);
481     text.setPosition(0, 64 * 4);
482     text.setColor(sf::Color::White);
483     text.setStyle(sf::Text::Bold | sf::Text::Underlined);
484     text.setCharacterSize(64);
485     text.setString("Player 1 Wins!");
486     target.draw(text);
487 }
488 }

```

## 8.5 Output :

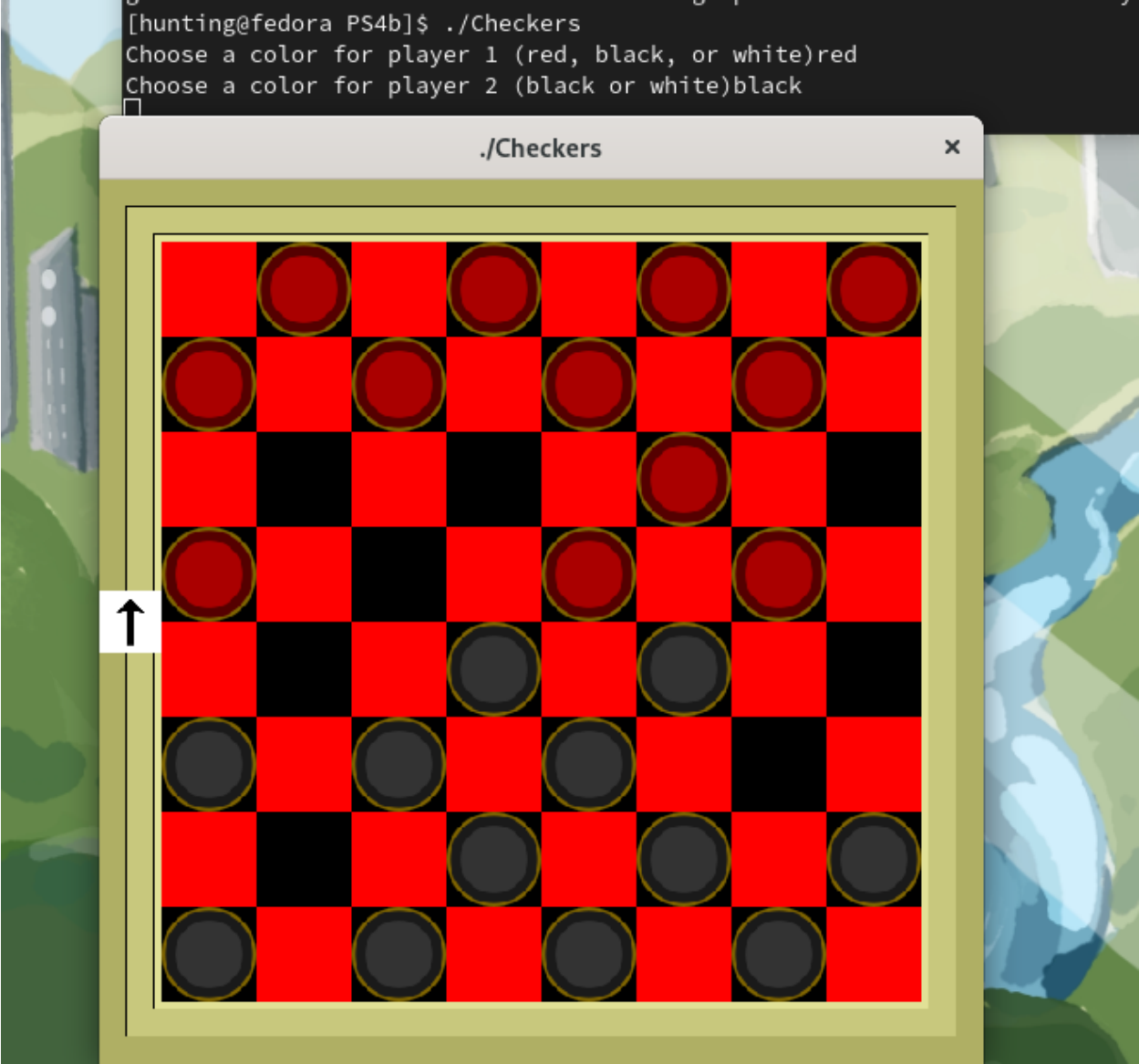


Figure 9: PS4b Output in Terminal

## 9 PS5: DNA Sequence Alignment

### 9.1 Discussion:

The main objective in this assignment was to construct a systematic way of identifying differences in strings and calculating the string that minimizes distance between the two strings. For the runtime and space complexity scale pretty drastically depending on the implementation of this program. It is very important to consider many different methods and how each influences the program.

### 9.2 Key algorithms, Data structures and OO Designs used in this Assignment:

The dynamic programming approach is used to calculate the edit distance between two strings. The choice of this approach is based on its efficiency and relative ease of implementation. The method stores intermediate results in a two-dimensional vector, thus avoiding redundant computations. This approach is more efficient than the recursive method without memoization, which can have exponential time complexity. Although recursive with memoization and Hirschberg's algorithm can also be efficient, dynamic programming is often simpler to implement and understand. The main disadvantage of dynamic programming is that it can consume more memory than other methods, particularly when working with very long strings.

### 9.3 What I learned :

I learned how to calculate the exponential growth and thus the runtime of the program. I did so by both utilizing the time command in C and also using valgrind to identify the amount of heap space allocated during runtime.

### 9.4 Codebase

#### Makefile :

This Makefile contains the lint.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
4 DEP = EDistance.h
5 OBJS = main.o EDistance.o
6
7 all: EDistance
8
9 EDistance: $(OBJS)
10     $(CC) -g -o EDistance $(OBJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12     $(CC) -g -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
14 lint:
15     cpplint.py --filter=-runtime/references,-build/c++11 --root=. *
16
17
18 cleanall : cleanobj
19     rm EDistance
20
21 cleanobj :
22     rm *.o
```

#### main.cpp :

```
1 #include <iostream>
2 #include <fstream>
```

```

3  #include <SFML/System.hpp>
4  #include "EDistance.h"
5
6  // Copyright [2023] Daniel Olen & Hunter Hasenfus
7
8  int main(int argc, char *argv[]) {
9      sf::Clock clock;
10     sf::Time t;
11
12     if (argc != 2) {
13         std::cerr << "Usage: " << argv[0] << " <input_file>" << std::endl;
14         return 1;
15     }
16
17     std::string input_file = argv[1];
18     std::ifstream infile(input_file);
19
20     if (!infile) {
21         std::cerr << "Error: Unable to open input file" << std::endl;
22         return 1;
23     }
24
25     std::string x, y;
26     std::getline(infile, x);
27     std::getline(infile, y);
28     infile.close();
29
30     EDistance eDistance(x, y);
31     int distance = eDistance.optDistance();
32     std::string alignment_str = eDistance.alignment();
33
34     std::cout << alignment_str;
35     std::cout << "Edit distance = " << distance << std::endl;
36
37
38     t = clock.getElapsedTime();
39     std::cout << "Execution time is " << t.asSeconds() << " seconds \n";
40
41     return 0;
42 }

```

#### EDistance.h :

```

1  #pragma once
2
3  #ifndef EDISTANCE_H
4  #define EDISTANCE_H
5
6  // Copyright [2023] Daniel Olen & Hunter Hasenfus
7
8  #include <string>
9  #include <vector>
10 #include <algorithm>
11
12 class EDistance {
13 public:
14     EDistance(const std::string &x, const std::string &y);
15
16     static int penalty(char a, char b);
17     static int min(int a, int b, int c);
18     int optDistance();

```

```
19     std::string alignment();
20     int n;
21 private:
22     std::string x, y;
23     std::vector<std::vector<int>> opt;
24 };
25
26 #endif
```

### EDistance.cpp :

```
1 #include "EDistance.h"
2 #include <algorithm>
3
4 // Copyright [2023] Daniel Olen & Hunter Hasenfus
5
6 EDistance::EDistance(const std::string &x, const std::string &y) :
7 x(x), y(y), opt(x.length() + 1, std::vector<int>(y.length() + 1)) {}
8
9 int EDistance::penalty(char a, char b) {
10     return a == b ? 0 : 1;
11 }
12
13 int EDistance::min(int a, int b, int c) {
14     return std::min(std::min(a, b), c);
15 }
16
17 int EDistance::optDistance() {
18     int n = x.length();
19     int m = y.length();
20
21     for (int i = 0; i <= n; ++i) {
22         opt[i][0] = i * 2;
23     }
24
25     for (int j = 0; j <= m; ++j) {
26         opt[0][j] = j * 2;
27     }
28
29     for (int i = 1; i <= n; ++i) {
30         for (int j = 1; j <= m; ++j) {
31             int match = opt[i - 1][j - 1] + penalty(x[i - 1], y[j - 1]);
32             int insert = opt[i - 1][j] + 2;
33             int delete_ = opt[i][j - 1] + 2;
34             opt[i][j] = min(match, insert, delete_);
35         }
36     }
37
38     return opt[n][m];
39 }
40
41 std::string EDistance::alignment() {
42     std::string result;
43     // int n = 0, temp;
44     int i = x.length(), j = y.length();
45
46     while (i > 0 || j > 0) {
47         if (i > 0 && opt[i][j] == opt[i - 1][j] + 2) {
48             result = std::string(1, x[i - 1]) +
49                 " - " + std::to_string(2) + "\n" + result;
50             // n+=2;
51             --i;
52         } else if (j > 0 && opt[i][j] == opt[i][j - 1] + 2) {
53             result = "- " + std::string(1, y[j - 1]) +
54                 " " + std::to_string(2) + "\n" + result;
55             // n+=2;
56             --j;
57         } else {
58             // temp = penalty(x[i - 1], y[j - 1]);
```



```

59     result = std::string(1, x[i - 1]) + " " + std::string(1, y[j -
1]) +
60     " " + std::to_string(penalty(x[i - 1], y[j - 1])) + "\n" +
result;
61     // n+=temp;
62     --i;
63     --j;
64 }
65 }
66
67
68 return result;
69 }

```

## 9.5 Output:

```

[hunting@fedora PS5]$ ./EDistance ./sequence/bothgaps20.txt
a a 0
- z 2
- z 2
b b 0
c c 0
d d 0
e e 0
f f 0
g g 0
h h 0
i i 0
z - 2
z - 2
z - 2
z - 2
j j 0
k k 0
l l 0
m m 0
n n 0
o o 0
p p 0
Edit distance = 12
Execution time is 0.000417 seconds
[hunting@fedora PS5]$

```

Figure 10: PS5 Output Window

## 10 PS6: Random Writer

### 10.1 Discussion:

The assignment delves into the topic of markov models and markov chains. A markov probability is one where the cumulative sum is 1.0, this is done by using frequency probabilities. In the assignment the frequencies of kgrams are found and then the frequencies of the subsequent letters are found. This enables the ability to create probability distributions for each kgram and then randomly generate streams of characters based off the training data sets acquired probabilities.

### 10.2 Key algorithms, Data structures and OO Designs used in this Assignment:

In this implementation I constructed such markov model using three separate data types. The first was a simple mapping from strings to ints, this was used for the frequencies of the kgrams. I then made a separate data structure of a mapping into a vector, this allowed me to create a vector for each kgram and denote probabilities for the following characters based on the frequency in which they were found in the training data. I chose to implement this with a vector because the last data structure I implemented was a mapping from strings into discrete distributions. By nature, the discrete distribution requires an input that returns unlabeled integers. By using the sparse 127 element vector I was able to very seamlessly return the integer equivalent of the ascii letters and convert to characters then and there.

The mapping to discrete distributions is pretty interesting in that, it seamlessly takes the vector of character probabilities and converts it into a probability distribution based on the discrete distribution. This allows the ability to pass a random generator into the probability distribution and pass out a value. Provides a great deal of simplification for the latter half of the program. line 63-67 RandWriter.cpp

```
1 for (auto it: kgramCharProb) {  
2     std::discrete_distribution<int> dist(it.second.begin(), it.second.  
    end());  
3     kgramCharDist[it.first] = dist;  
4 }  
5
```

### 10.3 What I learned :

I have learnt about the Markov model and how it works. Also learnt about the Hash Map. I used exceptions on all the points listed in the pdf. Mainly invalid argument exceptions that made sure the length of the kgrams were accurate. line 78-81 RandWriter.cpp

```
1 if (kgram.length() != order) {  
2     throw std::invalid_argument("kgram is not the same length as order"  
3     + kgram);  
4 }  
5
```

This was an exception to catch invalid kgrams by length == order. line 126-130 RandWriter.cpp

```
1 if (n == 0) {  
2     throw std::invalid_argument("kgram does not exist: " + kgram  
3     + ", " + static_cast<char>(c));  
4 }  
5
```

This was an exception to catch kgrams that were not apart of the original text.

## 10.4 Codebase

### Makefile :

This Makefile contains the lint.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lboost_unit_test_framework
4 DEP = RandWriter.h
5 OBJS = RandWriter.o TextWriter.o
6
7 all: TextWriter test
8
9 TextWriter: $(OBJS)
10     $(CC) -g -o TextWriter $(OBJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12     $(CC) -g -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
14 test: test.o $(DEP)
15     $(CC) -g -o test test.o RandWriter.o $(LFLAGS)
16 test.o:test.cpp
17     $(CC) -g -c $(CFLAGS) test.cpp $(LFLAGS)
18
19
20 lint:
21     cpplint.py --filter=--runtime/references,-build/c++11 --root=. *
22
23
24 cleanall : cleanobj
25     rm TextWriter
26
27 cleanobj :
28     rm *.o
```

### TextWriter.cpp :

```
1 // Copyright [2023] Hunter Hasenfus
2
3 #include <string>
4 #include <iostream>
5 #include <cstdlib>
6 #include "RandWriter.h"
7
8 int main(int argc, char* argv[]) {
9     if (argc != 3) {
10         std::cerr << "Usage: " << argv[0] << " k L" << std::endl;
11         return 1;
12     }
13     int k = atoi(argv[1]);
14     int L = atoi(argv[2]);
15     std::string text;
16     std::string line;
17     while (std::getline(std::cin, line)) {
18         text += line;
19     }
20     RandWriter rw(text, k);
21
22     //std::cout << rw << rw.generate(text.substr(0, k), L) << std::endl;
23     return 0;
24 }
```

### RandWriter.h :

```

1  // Copyright [2023] Hunter Hasenfus
2
3  #ifndef RANDWRITER_H_
4  #define RANDWRITER_H_
5
6  #include <string>
7  #include <vector>
8  #include <iostream>
9  #include <cstdlib>
10 #include <map>
11 #include <random>
12 #include <chrono>
13
14
15 class RandWriter{
16 public:
17     RandWriter(std::string text, int k);
18     int orderK() const;
19     int freq(std::string kgram) const;
20     int freq(std::string kgram, int c) const;
21     char kRand(std::string kgram);
22     std::string generate(std::string kgram, int L);
23     friend std::ostream& operator<<(std::ostream &out, RandWriter &rw);
24 private:
25     std::string BaseText;
26     std::map<std::string, int> kgrams;
27     std::map<std::string, std::vector<float>> kgramCharProb;
28     std::map<std::string, std::discrete_distribution<int>> kgramCharDist;
29     int order;
30     std::vector<char> alphabet;
31     std::default_random_engine generator;
32 };
33
34 #endif // RANDWRITER_H_

```

## RandWriter.cpp :

```
1 // Copyright [2023] Hunter Hasenfus
2
3 #include <string>
4 #include <algorithm>
5 #include <cstdlib>
6 #include <iostream>
7 #include <vector>
8 #include <map>
9 #include <random>
10 #include <chrono>
11 #include "RandWriter.h"
12
13
14 RandWriter::RandWriter(std::string text, int k) {
15     // unsigned nseed = std::chrono::system_clock::now().
16     // time_since_epoch().count();
17     // generator.seed(nseed);
18     BaseText = text;
19     BaseText.erase(std::remove(BaseText.begin(), BaseText.end(), '\n'),
20         BaseText.cend());
21     // std::cout << BaseText << std::endl;
22     order = k;
23     int x = orderK();
24     for (int i = 0; i < 127; i++) {
25         if (BaseText.find(static_cast<char>(i)) != std::string::npos) {
26             alphabet.push_back(static_cast<char>(i));
27         }
28     }
29     qsort(&alphabet[0], alphabet.size(), sizeof(char), [](const void *a,
30         const void *b) { return (*(char *)a - *(char *)b); });
31     for (int i = 0; i < BaseText.length(); i++) {
32         std::string kgram = BaseText.substr(i, order);
33         if (kgram.size() < order) {
34             int x = (i + order) - BaseText.length();
35             kgram += BaseText.substr(0, x);
36
37             if (kgrams.find(kgram) == kgrams.end()) {
38                 kgrams[kgram] = freq(kgram);
39                 kgramCharProb[kgram] = std::vector<float>(127, 0);
40                 kgramCharProb[kgram][BaseText[x]] = static_cast<float>
41                     (freq(kgram,
42                         BaseText[x])) / kgrams[kgram];
43                 // std::cout << kgramCharProb[kgram][BaseText[x]]
44                 // << std::endl;
45             } else { if (kgramCharProb[kgram][BaseText[x]] == 0) {
46                 kgramCharProb[kgram][BaseText[x]] = static_cast<float>
47                     (freq(kgram,
48                         BaseText[x])) / kgrams[kgram];
49                 // std::cout << kgramCharProb[kgram]
50                 // [BaseText[x]] << std::endl;
51             } }
52
53         } else {
54             int x = (i + order) == BaseText.length() ? 0 : i + order;
55
56             if (kgrams.find(kgram) == kgrams.end()) {
57                 kgrams[kgram] = freq(kgram);
58                 kgramCharProb[kgram] = std::vector<float>(127, 0);
```

```

59         kgramCharProb[kgram][BaseText[x]] = static_cast<float>
60         (freq(kgram, BaseText[x]))
61         / kgrams[kgram];
62     } else { if (kgramCharProb[kgram][BaseText[x]] == 0) {
63         kgramCharProb[kgram][BaseText[x]] = static_cast<float>(freq(
kgram,
64         BaseText[x])) / kgrams[kgram];
65     }}}}
66     // std::cout << "HERE" << std::endl;
67     for (auto it : kgramCharProb) {
68         std::discrete_distribution<int> dist(it.second.begin(),
69         it.second.end());
70         kgramCharDist[it.first] = dist;
71     }
72 }
73
74 int RandWriter::orderK() const {
75     return order;
76 }
77 int RandWriter::freq(std::string kgram) const {
78     if (kgram.length() != order) {
79         throw std::invalid_argument("kgram is not the same length as order"
80         + kgram);
81     }
82     if (kgrams.find(kgram) == kgrams.end()) {
83         throw std::invalid_argument("kgram is not valid");
84     }
85     int n = 0;
86     for (int i = 0; i < BaseText.length(); i++) {
87         std::string kgram2 = BaseText.substr(i, order);
88         if (kgram2.size() < order) {
89             int x = (i + order) - BaseText.length();
90             kgram2 += BaseText.substr(0, x);
91             if (kgram2 == kgram) {
92                 n++;
93             }
94         } else {
95             if (kgram2 == kgram) {
96                 n++;
97             }
98         }
99     }
100     return n;
101 }
102 int RandWriter::freq(std::string kgram, int c) const {
103     if (kgram.length() != order) {
104         throw std::invalid_argument("kgram is not the same length as order"
+
105         kgram);
106     }
107     int n = 0;
108     for (int i = 0; i < BaseText.length(); i++) {
109         std::string kgram2 = BaseText.substr(i, order);
110
111         if (kgram2.size() < order) {
112             int x = (i + order) - BaseText.length();
113             kgram2 += BaseText.substr(0, x);
114             // std::cout << "(1) " << i << " - ";
115             // std::cout << BaseText[x] << ": " << x << std::endl;

```

```

116         if (kgram2 == kgram && BaseText[x] == c) {
117             n++;
118         }
119
120     } else {
121         int x = (i + order) == BaseText.length() ? 0 : i + order;
122         // std::cout << "(2) " << i << " - ";
123         // std::cout << BaseText[x] << ": " << x << std::endl;
124         if (kgram2 == kgram && BaseText[x] == c) {
125             n++;
126         }}
127     }
128
129     if (n == 0) {
130         throw std::invalid_argument("kgram does not exist: " + kgram
131             + ", " + static_cast<char>(c));
132     }
133
134     return n;
135 }
136
137 char RandWriter::kRand(std::string kgram) {
138     int x = kgramCharDist[kgram](generator);
139     // int i = 0;
140     // std::cout << kgram << ":" << std::endl;
141     // for (double x: kgramCharDist[kgram].probabilities()) {
142     //     if (x > 0)
143     //         std::cout << char(i) << "- " << x << std::endl;
144     //     i++;
145     // }
146
147     return x;
148 }
149 std::string RandWriter::generate(std::string kgram, int L) {
150     if (kgram.length() != order) {
151         throw std::invalid_argument("kgram is not the same length as order")
152     };
153     if (kgrams.find(kgram) == kgrams.end()) {
154         throw std::invalid_argument("kgram does not exist: " + kgram);
155     }
156     std::string kgram2 = kgram;
157     std::string output = kgram2;
158     for (int i = 0; i < L - kgram.length(); i++) {
159         output += static_cast<char>(kRand(kgram2));
160         // std::cout << kRand(kgram2) << std::endl;
161         // std::cout << "kgram2: " << kgram2 << std::endl;
162         kgram2 = output.substr(output.length() - order, order);
163     }
164     return output;
165 }
166 std::ostream& operator<<(std::ostream &out, RandWriter &rw) {
167     std::cout << "Order: " << rw.orderK() << std::endl;
168     std::cout << "Alphabet: " << std::endl;
169     for (char c : rw.alphabet) {
170         std::cout << c << ", ";
171     }
172     std::cout << std::endl << "kgrams frequencies: " << std::endl;
173     for (auto x : rw.kgrams) {

```

```

174     std::cout << x.first << ": " << x.second << std::endl;
175 }
176 std::cout << "kgrams + 1 frequencies: " << std::endl;
177
178 return out;
179 }

```

#### test.cpp :

```

1  // Copyright [2023] Hunter Hasenfus
2
3  #define BOOST_TEST_MODULE RandWriterTest
4  #include <boost/test/included/unit_test.hpp>
5  #include "./RandWriter.h"
6
7  BOOST_AUTO_TEST_SUITE(RandWriterTest)
8  BOOST_AUTO_TEST_CASE(test1) {
9      RandWriter rw("abc", 1);
10     BOOST_REQUIRE_THROW(rw.freq("ab"), std::invalid_argument);
11 }
12 BOOST_AUTO_TEST_CASE(test2) {
13     RandWriter rw("abc", 1);
14     BOOST_REQUIRE_THROW(rw.freq("ab", 'c'), std::invalid_argument);
15 }
16 BOOST_AUTO_TEST_CASE(test3) {
17     RandWriter rw("abc", 2);
18     BOOST_REQUIRE_THROW(rw.freq("xy"), std::invalid_argument);
19 }
20 BOOST_AUTO_TEST_CASE(test4) {
21     RandWriter rw("abc", 2);
22     BOOST_REQUIRE_THROW(rw.freq("xy", 'c'), std::invalid_argument);
23 }
24 BOOST_AUTO_TEST_CASE(test5) {
25     RandWriter rw("abc", 1);
26     BOOST_REQUIRE_THROW(rw.generate("x", 1), std::invalid_argument);
27 }
28 BOOST_AUTO_TEST_CASE(test6) {
29     RandWriter rw("abc", 1);
30     BOOST_REQUIRE_THROW(rw.generate("ab", 2), std::invalid_argument);
31 }
32 BOOST_AUTO_TEST_CASE(test7) {
33     RandWriter rw("abc", 0);
34     BOOST_REQUIRE_EQUAL(rw.orderK(), 0);
35 }
36 BOOST_AUTO_TEST_CASE(test8) {
37     RandWriter rw("abc", 1);
38     BOOST_REQUIRE_EQUAL(rw.kRand("a"), 'b');
39 }
40 BOOST_AUTO_TEST_CASE(test9) {
41     RandWriter rw("abc", 0);
42     BOOST_REQUIRE_EQUAL(rw.freq(""), 3);
43 }
44 BOOST_AUTO_TEST_SUITE_END()

```



## 11 PS7: Kronos Log Parsing

### 11.1 Discussion:

We analyze the Kronos Intouch time clock log by using regular expressions to parse the file, in addition, we verified device boot up timing. We take the given device[1-6]\*underscore\*intouch.log files and give a resultant file which contains the time, date, status and line number of the boot to the .rpt file.

### 11.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I used the lambda expression for getting the time from first 19 characters i.e substring(0,19). I prefer using lambda expression than ordinary method as its more efficient. Just utilized the three functions from the regex library to finish my project.

### 11.3 Explanation of the code:

Firstly, I initialize regular expressions to the starting and ending log entries so it can be used to compare the regex to find. if there is a match in every line of the file(.log). I create an -outputfile where the data of the result is stored as filename.log.rpt .There is a use of bool exp to keep tracking of the incomplete booting. I utilize the regex-search function to search a match for the starting regular expression(log.c.166). if there is no incomplete booting, I write the date, time, line number and status of the boot to our -outputfile. The else if condn is used to search the line for finding the match for the ending regular expression. if that is found, The calculation of the duration of the time is begun by using posix. Again I write the line-number, date , time and duration it took for the completion of sequence.

### 11.4 What I learned :

I learnt implementing of the regex library <regex> More efficiently.  
I have implemented following regex functions into my code.

- boost::regex startMessage("(log.c.166) server started");
- boost::regex endMessage("(oejs.AbstractConnector:Started SelectChannelConnector)");
- regex-search(s, startMessage)
- regex-search(s, endMessage)

### 11.5 Codebase

#### Makefile:

This Makefile has lint.

```
1 CC = g++
2 CFLAGS = -c -Wall -Werror -std=c++14
3 LFLAGS = -lboost_regex
4 DEP =
5 OBJJS = stdinboost.o
6
7 all: ps7
8
9 ps7: $(OBJJS)
10     $(CC) -g -o ps7 $(OBJJS) $(LFLAGS)
11 $(objects): %.o: %.cpp $(DEP)
12     $(CC) -g -c $(CFLAGS) $< -o $@ $(LFLAGS)
13
```

```

14 lint:
15     cpplint.py --filter=-runtime/references,-build/c++11 --root=. *
16
17
18 cleanall : cleanobj
19     rm ps7
20
21 cleanobj :
22     rm *.o

```

#### stdinboost.cpp:

```

1  // // Copyright [2023] Hunter M Hasenfus
2
3  #include <iostream>
4  #include <fstream>
5  #include <string>
6  #include <regex>
7  #include <boost/date_time/posix_time/posix_time.hpp>
8
9  int main(int argc, char* argv[]) {
10     std::ifstream lf(argv[1]);
11     if (!lf) {
12         std::cerr << "Error with log file: " << argv[1] << std::endl;
13         return 1;
14     }
15
16     std::ofstream of(std::string(argv[1]) + ".rpt");
17     if (!of) {
18         std::cerr << "Error with output file: "
19             << argv[1] << ".rpt" << std::endl;
20         return 1;
21     }
22
23     std::string line, rs;
24     std::regex timePattern(R"((\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}))");
25     std::regex startPattern(R"(((log\.c\.166)\) server started)");
26     std::regex finishPattern(R"(oejs\.AbstractConnector:Started
27         SelectChannelConnector@\d+\.\d+\.\d+\.\d+:\d+)");
28     boost::posix_time::ptime start_t, finish_t;
29     int n;
30     bool start = false;
31
32     while (getline(lf, line)) {
33         n++;
34
35         std::smatch match;
36         if (std::regex_search(line, match, timePattern)
37             && match.size() > 1) {
38             std::string timestamp_str = match[1];
39             boost::posix_time::ptime current_time =
40                 boost::posix_time::time_from_string(timestamp_str);
41
42             if (std::regex_search(line, match, startPattern)) {
43                 if (start) {
44                     of << "\t\t**** Incomplete boot **** \n\n";
45                     start = false;
46                 } else {
47                     of << "=== Device boot ===\n";
48                     of << n << " (" << argv[1] << "): " <<

```

```

49         start_t << " Boot Start" << std::endl;
50         start_t = current_time;
51         start = true;
52     }
53 } else if (std::regex_search(line, match, finishPattern)) {
54     if (start) {
55         finish_t = current_time;
56         boost::posix_time::time_duration elapsed =
57             finish_t - start_t;
58         of << n << " (" << argv[1] << "): " <<
59             finish_t << " Boot Completed" << std::endl;
60         of << "\t\tBoot Time: " <<
61             elapsed.total_milliseconds() << "ms\n" << std::endl;
62         start = false;
63     }}}}
64
65
66
67 if (start) {
68     of << "=== Device boot ===\n";
69     of << n << " (" << argv[1] << "): " <<
70         start_t << " Boot Start" << std::endl;
71 }
72
73 lf.close();
74 of.close();
75 return 0;
76 }

```