

# Computing IV Sec 201A: Project Portfolio

Brendan Kelleher

Spring 2025

## Contents

<b>1</b>	<b>PS0: Hello SFML</b>	<b>2</b>
<b>2</b>	<b>PS1: Photomagic</b>	<b>5</b>
<b>3</b>	<b>PS2: Triangle Fractal</b>	<b>12</b>
<b>4</b>	<b>PS3: N-Body Simulation</b>	<b>16</b>
<b>5</b>	<b>PS4: Sokoban</b>	<b>24</b>
<b>6</b>	<b>PS5: DNA Alignment</b>	<b>36</b>
<b>7</b>	<b>PS6: RandWriter</b>	<b>42</b>
<b>8</b>	<b>PS7: RandWriter</b>	<b>48</b>

Time to Complete Portfolio: 10 hours

# 1 PS0: Hello SFML

## 1.1 Discussion

The goal of this project was to set up an environment to be able to complete the rest of the projects for the semester. This project served as an introduction to SFML and allowed for me to become more comfortable with SFML by making the tasks extremely simple. The first step of the project was to draw a sprite to the screen, which the starter coded already did. Then I had to give the sprite the ability to move, which was done by modifying its x and y values. Then to make it respond to keystrokes I used event listeners to search for and up, down, left, or right input on the keyboard and then move the sprite accordingly. An additional feature was also required to be added at the programmers discretion and I chose to make the output rotate 90 degrees if the R key is pressed. This was done by using SFML's setRotation and getRotation functions along with an event listener for the R key.

## 1.2 What I learned

While working on this project I got eased into SFML and its capabilities. Figuring out how to move the sprite based on keystrokes made me comfortable with the event handling that would be necessary for the additional feature, and for future projects. Implementing the additional feature required me to dig deeper into the SFML documentation to be able to apply something myself.

## 1.3 Challenges

As this project was more of an introduction I didn't have too many issues while building it, however when I first turned in the assignment the project was looking for a sprite that was not in the project directory. After adjusting what file the project was looking for everything was back to working the way it was supposed to

## 1.4 Codebase

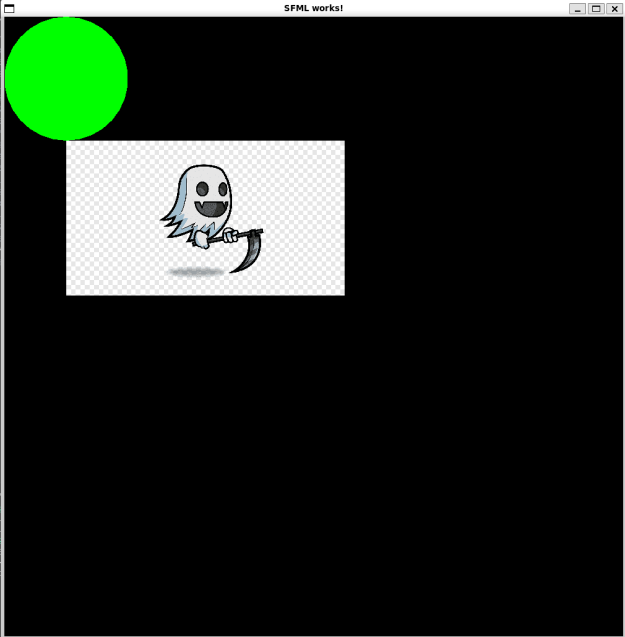
```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
      lboost_unit_test_framework
4 # Your .hpp files
5 DEPS =
6 # Your compiled .o files
7 OBJECTS = main.o
8 # The name of your program
9 PROGRAM = sfml-app
10
11 .PHONY: all clean lint
12
13
14 all: $(PROGRAM)
15
16 # Wildcard recipe to make .o files from corresponding .cpp file
17 %.o: %.cpp $(DEPS)
18     $(CC) $(CFLAGS) -c $<
19
20 $(PROGRAM): main.o $(OBJECTS)
21     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
22
23 clean:
24     rm *.o $(PROGRAM)
25
26 lint:
27     cpplint *.cpp *.hpp
```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <SFML/Graphics.hpp>
3 int main() {
4     int x = 100;
5     int y = 200;
6     sf::RenderWindow window(sf::VideoMode(1000, 1000), "SFML works!");
7     sf::CircleShape shape(100.f);
8     shape.setFillColor(sf::Color::Green);
9     sf::Texture texture;
10    if (!texture.loadFromFile("sprite.png")) {
11        return EXIT_FAILURE;
12    }
13    sf::Sprite sprite(texture);
14    sprite.setScale(0.5f, 0.5f);
15    sprite.setPosition(x, y);
16    while (window.isOpen()) {
17        sf::Event event;
18        while (window.pollEvent(event)) {
19            if (event.type == sf::Event::Closed)
20                window.close();
21            if (event.type == sf::Event::KeyPressed) {
22                if (event.key.code == sf::Keyboard::Left) {
23                    x -= 5;
24                    sprite.setPosition(x, y);
25                } else if (event.key.code == sf::Keyboard::Right) {
26                    x += 5;
27                    sprite.setPosition(x, y);
28                } else if (event.key.code == sf::Keyboard::Up) {
29                    y -= 5;
30                    sprite.setPosition(x, y);
31                } else if (event.key.code == sf::Keyboard::Down) {
32                    y += 5;
33                    sprite.setPosition(x, y);
34                } else if (event.key.code == sf::Keyboard::R) {
35                    sprite.setRotation(sprite.getRotation() + 90);
36                }
37            }
38        }
39
40        window.clear();
41        window.draw(shape);
42        window.draw(sprite);
43        window.display();
44    }
45
46    return 0;
47 }

```

## 1.5 Output



## 2 PS1: Photomagic

### 2.1 Discussion

The goal of this project was to be able to encrypt and decrypt an image with the use of a binary seed and a linear feedback shift register. The LFSR creates semi random bits based on the binary key provided. These generated bits are then used to encrypt an image by changing the RGB values of the pixels throughout the image. In order to decrypt the image the program must be ran again with the same binary key and the encrypted image as input. The result will be the original image.

### 2.2 Key Algorithms Used

The central mechanism to this program is the Linear feedback shift register. The class holds the length of the seed, the seed itself and an array of the taps that will be used in generating the next bits. It is mainly implemented with a step, and generate functions. The step function returns the next rightmost bit as an integer. (0 or 1). Step accomplishes this by finding the current element of the taps array and xoring that with the leftmost bit. Generate returns an integer that simulates k steps of the LFSR. Generate is implemented by calling step k times and each time adding the result to a stored value, and multiplying that stored value by 2 for each step to account for the left shift in bits.

### 2.3 What I learned

Working on this project allowed me to see a way in which images could be encrypted. Even though this seems to be a basic encryption method I have never implemented anything like it myself and I believe that experience to be helpful. This project also served as an introduction to boost tests for me. Learning how to use these tests proved crucial for many of the future projects and I will continue to use them to assist with my programming.

### 2.4 Challenges

The main challenge of this project for me was trying to understand the linear feedback shift register. Once I had an understanding of what was happening the implementation of the function did not prove too difficult.

### 2.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system
4 TEST_LIB = -lboost_unit_test_framework
5 AR = ar
6 ARFLAGS = rcs
7
8 SOURCES = main.cpp FibLFSR.cpp PhotoMagic.cpp
9 DEPS = FibLFSR.hpp PhotoMagic.hpp
10 OBJECTS = $(SOURCES:.cpp=.o)
11
12
13
14 LIBRARY = PhotoMagic.a
15 LIBRARY_OBJECTS = FibLFSR.o PhotoMagic.o
16 PROGRAM = PhotoMagic
17
18 TEST_SOURCES = test.cpp
19 TEST_OBJECTS = $(TEST_SOURCES:.cpp=.o)
20 TEST_PROGRAM = test
21
22 .PHONY: all clean lint test_runner
```

```

23
24 all: $(LIBRARY) $(PROGRAM) $(TEST_PROGRAM)
25
26 %.o: %.cpp $(DEPS)
27     $(CC) $(CFLAGS) -c $< -o $@
28
29 $(LIBRARY): $(LIBRARY_OBJECTS)
30     $(AR) $(ARFLAGS) $@ $^
31
32 $(PROGRAM): main.o $(LIBRARY)
33     $(CC) $(CFLAGS) -o $@ main.o $(LIBRARY_OBJECTS) $(LIB)
34
35 $(TEST_PROGRAM): $(TEST_OBJECTS) $(LIBRARY)
36     $(CC) $(CFLAGS) -o $@ $(TEST_OBJECTS) $(LIBRARY_OBJECTS) $(TEST_LIB) $(
    LIB)
37
38 test_runner: $(TEST_PROGRAM)
39     ./$(TEST_PROGRAM)
40
41 clean:
42     rm -f *.o $(PROGRAM) $(TEST_PROGRAM) $(LIBRARY)
43
44 lint:
45     cpplint *.cpp *.hpp

```

```

1  // Copyright 2025 Brendan Kelleher
2  #include <iostream>
3  #include <algorithm>
4  #include "../FibLFSR.hpp"
5  #include "../PhotoMagic.hpp"
6  #include <SFML/System.hpp>
7  #include <SFML/Window.hpp>
8  #include <SFML/Graphics.hpp>
9
10
11
12
13
14 int main(int argc, char* argv[]) {
15     try {
16         if (argc != 4) {
17             throw std::invalid_argument
18                 (" input an input-file an output-file and an LFSR seed");
19         }
20         sf::Image image;
21         if (!image.loadFromFile(argv[1]))
22             return -1;
23         sf::Image input = image;
24         PhotoMagic::FibLFSR seed(argv[3]);
25         transform(image, &seed);
26         // fredm: saving a PNG segfaults for me, though it does properly
27         // write the file
28         if (!image.saveToFile(argv[2])) {
29             return -1;
30         }
31         sf::Image output = image;
32         sf::Texture inputTexture, outputTexture;
33         inputTexture.loadFromImage(input);
34         outputTexture.loadFromImage(output);
35         sf::Sprite inputSprite(inputTexture);

```

```

36     sf::Sprite outputSprite(outputTexture);
37     sf::Vector2u size = image.getSize();
38     sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "input");
39     sf::RenderWindow window2(sf::VideoMode(size.x, size.y), "output");
40     while (window1.isOpen() && window2.isOpen()) {
41 sf::Event event;
42 while (window1.pollEvent(event)) {
43 if (event.type == sf::Event::Closed)
44 window1.close();
45 }
46 while (window2.pollEvent(event)) {
47 if (event.type == sf::Event::Closed)
48 window2.close();
49 }
50 window1.clear(sf::Color::Black);
51 window1.draw(inputSprite);
52 window1.display();
53 window2.clear(sf::Color::Black);
54 window2.draw(outputSprite);
55 window2.display();
56 }
57 }
58     catch(std::invalid_argument& arg) {
59         std::cout << arg.what() << std::endl;
60     }
61     return 0;
62 }

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <algorithm>
3 #include <SFML/System.hpp>
4 #include <SFML/Window.hpp>
5 #include <SFML/Graphics.hpp>
6 #include "./FibLFSR.hpp"
7
8 #ifndef PHOTOMAGIC_HPP_
9 #define PHOTOMAGIC_HPP_
10 namespace PhotoMagic {
11 void transform(sf::Image& I, PhotoMagic::FibLFSR* fib);
12 } // namespace PhotoMagic
13 #endif // PHOTOMAGIC_HPP_

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include "./PhotoMagic.hpp"
3 #include <algorithm>
4 #include <SFML/System.hpp>
5 #include <SFML/Window.hpp>
6 #include <SFML/Graphics.hpp>
7 namespace PhotoMagic {
8 void transform(sf::Image& I, PhotoMagic::FibLFSR* fib) {
9 int random_num = fib->generate(8);
10 sf::Color p;
11 sf::Vector2u size = I.getSize();
12 for (unsigned int x = 0; x < size.x; x++) {
13     for (unsigned int y = 0; y < size.y; y++) {
14         p = I.getPixel(x, y);
15         p.r = (random_num ^ p.r);
16         random_num = fib->generate(8);
17         p.g = (random_num ^ p.g);
18         random_num = fib->generate(8);

```

```

19         p.b = (random_num ^ p.b);
20         I.setPixel(x, y, p);
21     }
22 }
23 }
24 } // namespace PhotoMagic

```

```

1 // Copyright 2025 Brendan Kelleher
2 #ifndef FIBLFSR_HPP_
3 #define FIBLFSR_HPP_
4
5 #include <iostream>
6 #include <string>
7
8 namespace PhotoMagic {
9
10 class FibLFSR {
11 public:
12     // Constructor to create LFSR with the given initial seed
13     explicit FibLFSR(std::string inputSeed);
14
15     // Simulate one step and return the new bit as 0 or 1
16     int step();
17
18     // Simulate k steps and return a k-bit integer
19     int generate(int k);
20     std::string getSeed() const { return seed; }
21
22     // Overload the output operator
23     friend std::ostream& operator<<(std::ostream& out, const FibLFSR& lfsr);
24
25 private:
26     // Fields
27     int seedLength;
28     std::string seed;
29     int taps[3];
30 };
31
32
33 } // namespace PhotoMagic
34
35 #endif // FIBLFSR_HPP_

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include<iostream>
3 #include<string>
4 #include "../FibLFSR.hpp"
5
6 namespace PhotoMagic {
7 // Constructor to create LFSR with the given initial seed
8 FibLFSR::FibLFSR(std::string inputSeed) {
9     if (inputSeed.size() != 16) {
10         throw std::invalid_argument("seed must be 16 digits long");
11     }
12     for (int i = 0; i < 16; i++) {
13         if (inputSeed[i] != '1' && inputSeed[i] != '0') {
14             throw std::invalid_argument("seed must only contain 1's or 0's")
15         }
16     }
17 }

```



```

17 seed = inputSeed;
18 seedLength = seed.size();
19 taps[0] = 10;
20 taps[1] = 12;
21 taps[2] = 13;
22 }
23 // Simulate one step and return the new bit as 0 or 1
24 int FibLFSR::step() {
25     int heldBit;
26     // store value of leftmost bit
27 if(seed[0] == '0') {
28     heldBit = 0;
29 } else {
30     heldBit = 1;
31 }
32 // initialize l to amount of elements in the tap array
33 for (int i = sizeof(taps) / sizeof(taps[0]) - 1; i >= 0; i--) {
34     int bitValue = (seed[15 - (taps[i])] == '1') ? 1 : 0;
35     if (heldBit == bitValue) {
36         heldBit = 0;
37     } else {
38         heldBit = 1;
39     }
40 }
41 std::string newSeed(seedLength, '0');
42 char stringBit = (heldBit == 1) ? '1' : '0';
43
44 for (int i = 0; i < 15; i++) {
45     newSeed[i] = seed[i+1];
46 }
47 newSeed[seedLength - 1] = stringBit;
48 seed = newSeed;
49 return heldBit;
50 }
51
52 // Simulate k steps and return a k-bit integer
53 int FibLFSR::generate(int k) {
54     if (k < 0) {
55         throw std::invalid_argument("input for generate must be at least 0")
56     ;
57     }
58     int storedValue = 0;
59     while (k > 0) {
60         int stepValue = step();
61         storedValue = (storedValue*2) + stepValue;
62         k--;
63     }
64     return storedValue;
65 }
66 std::ostream& operator<<(std::ostream& out, const FibLFSR& lfsr) {
67     out << lfsr.getSeed();
68     return out;
69 }
70
71
72
73
74 } // namespace PhotoMagic

```

```

1 // Copyright 2022
2 // By Dr. Rykalova
3 // Editted by Dr. Daly
4 // test.cpp for PS1a
5 // updated 1/8/2024
6
7 #include <iostream>
8 #include <string>
9 #include <sstream>
10 #include "../FibLFSR.hpp"
11
12 #define BOOST_TEST_DYN_LINK
13 #define BOOST_TEST_MODULE Main
14 #include <boost/test/included/unit_test.hpp>
15
16 using PhotoMagic::FibLFSR;
17
18 BOOST_AUTO_TEST_CASE(testStepInstr) {
19     FibLFSR l("1011011000110110");
20     BOOST_REQUIRE_EQUAL(l.step(), 0);
21     BOOST_REQUIRE_EQUAL(l.step(), 0);
22     BOOST_REQUIRE_EQUAL(l.step(), 0);
23     BOOST_REQUIRE_EQUAL(l.step(), 1);
24     BOOST_REQUIRE_EQUAL(l.step(), 1);
25     BOOST_REQUIRE_EQUAL(l.step(), 0);
26     BOOST_REQUIRE_EQUAL(l.step(), 0);
27     BOOST_REQUIRE_EQUAL(l.step(), 1);
28 }
29
30 BOOST_AUTO_TEST_CASE(testGenerateInstr) {
31     FibLFSR l("1011011000110110");
32     BOOST_REQUIRE_EQUAL(l.generate(9), 51);
33     BOOST_REQUIRE_EQUAL(l.generate(1), 0);
34     FibLFSR k("1011011000110110");
35     BOOST_REQUIRE_EQUAL(k.generate(10), 102);
36     BOOST_CHECK_THROW(l.generate(-1), std::invalid_argument);
37 }
38
39 BOOST_AUTO_TEST_CASE(testConstructorInstr) {
40     BOOST_CHECK_THROW(FibLFSR k("1011011000110112"), std::invalid_argument);
41 }
42 BOOST_AUTO_TEST_CASE(testOstreamInstr) {
43     FibLFSR l("1011011000110110");
44     std::ostringstream output;
45     output << l;
46     BOOST_CHECK_EQUAL(output.str(), "1011011000110110");
47 }
48 BOOST_AUTO_TEST_CASE(testBufferAfterSteps) {
49     FibLFSR l("1011011000110110");
50     std::ostringstream output;
51     std::ostringstream output2;
52     l.step();
53     output << l;
54     BOOST_CHECK_EQUAL(output.str(), "0110110001101100");
55     l.step();
56     output2 << l;
57     BOOST_CHECK_EQUAL(output2.str(), "1101100011011000");
58 }

```

2.6 Output

Encrypted Image



Decrypted Image



## 3 PS2: Triangle Fractal

### 3.1 Discussion

The goal of this project was to write a program that draws a variation of a sierpinski triangle. When the program is ran the user is prompted for a side length and a depth of recursion for the triangle. Each recursive step results in drawing 3 smaller triangles attached to the vertices of the triangle. These resulting sub-triangles have a side length half of what their parent triangle had. As the depth of recursion increases the time to run also increases exponentially. This happens because the number of triangles needed to be drawn increases exponentially in correlation with the depth of recursion.

### 3.2 Key Algorithms Used

The only calculations made during this project were the calculations necessary to find the positions of new triangles. Because one triangle will come from each vertex and will have one vertex shared with the original triangles only 2 points must be calculated per triangle. For the new top left triangle, the bottom point is the top left of the current triangle. The location of the top left corner is calculated by subtracting the side length divided by two from the top left corner and subtracting the height from the y value. The top right corner of the new triangle is calculated by adding the length to x instead of subtracting, and still subtracting the height from y. The other triangles points are calculated in similar manners. These new points are then passed into 3 separate calls of draw triangle and recursion continues until the desired depth is reached.

### 3.3 What I learned

Working on this project served as a nice refresher on the applications of recursion. I tend to struggle understanding recursion, but the more I practice the more confident I become and this project was a good help. I also learned how to utilize SFMLs draw function. This function is extremely useful and drawing the triangles served as a nice introduction to the capabilities of the function.

### 3.4 Challenges

The main challenge of this project for me was deriving the equation that correctly calculated the position of the triangles. I was able to get the triangles close together, but struggled to get the correct arrangement. After some trial and error I was able to get the triangles to draw in the correct positions

### 3.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = triangle.hpp
6 # Your compiled .o files
7 OBJECTS = triangle.o main.o
8 # The name of your program
9 PROGRAM = Triangle
10
11 .PHONY: all clean lint Triangle
12
13
14 all: $(PROGRAM)
15
16 # Wildcard recipe to make .o files from corresponding .cpp file
17 %.o: %.cpp $(DEPS)
```

```

18 $(CC) $(CFLAGS) -c $<
19
20 $(PROGRAM): main.o $(OBJECTS)
21 $(CC) $(CFLAGS) -o $@ $^ $(LIB)
22
23 clean:
24 rm *.o $(PROGRAM)
25
26 lint:
27 cpplint *.cpp *.hpp

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <iostream>
3 #include <cmath>
4 #include <SFML/Graphics.hpp>
5 #include "./triangle.hpp"
6 int main(int argc, char* argv[]) {
7     try {
8         if (argc != 3) {
9             throw std::invalid_argument
10                ("enter a side length and a depth of recursion");
11         }
12         double length = atof(argv[1]);
13         int depth = atoi(argv[2]);
14         sf::RenderWindow window
15             (sf::VideoMode(length * 3, length * 3), "Fractal Triangle");
16         sf::Vector2f startPoint
17             (window.getSize().x / 2, window.getSize().y / 2 + (sqrt(3) / 12) *
18              length);
19         // Center position
20         while (window.isOpen()) {
21             sf::Event event;
22             while (window.pollEvent(event)) {
23                 if (event.type == sf::Event::Closed)
24                     window.close();
25             }
26
27             // Clear the window
28             window.clear(sf::Color::Black);
29
30             // Draw the fractal triangle
31             fractal(&window, depth, length, startPoint);
32
33             // Display the frame
34             window.display();
35         }
36     }
37     catch(std::invalid_argument& arg) {
38         std::cout << "error: " << arg.what() << std::endl;
39     }
40     return 0;
41 }

```

```

1 // Copyright 2025 Brendan Kelleher
2 #ifndef _TRIANGLE_HPP_
3 #define _TRIANGLE_HPP_
4 #include <SFML/Graphics.hpp>
5 void fractal
6 (sf::RenderTarget* window, int n, double length, sf::Vector2f point);

```

```

7 void drawTriangle
8 (sf::RenderTarget* window, int n,
9 double length, sf::Vector2f top, sf::Vector2f bl, sf::Vector2f br);\
10
11 #endif // _TRIANGLE_HPP_

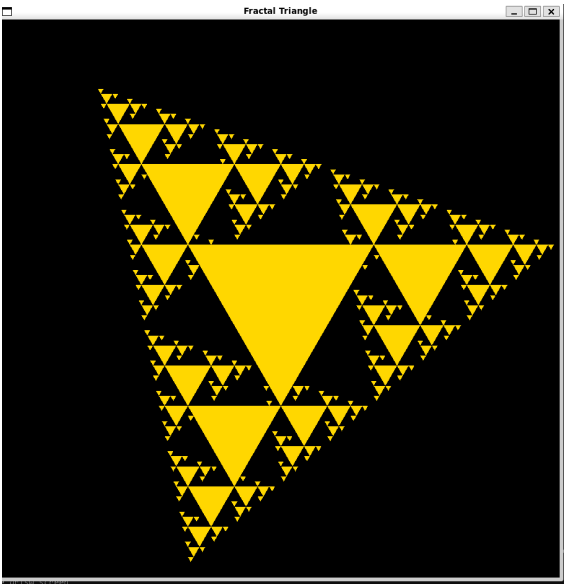
```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <cmath>
3 #include "../triangle.hpp"
4 #include <SFML/Graphics.hpp>
5 void fractal
6 (sf::RenderTarget* window, int n, double length, sf::Vector2f point) {
7     float h = sqrt(3) / 2 * length;
8     sf::Vector2f bottom = {0, h / 2};
9     sf::Vector2f left = {static_cast<float>(-length / 2), -h / 2.f};
10    sf::Vector2f right = {static_cast<float>(length / 2), -h / 2.f};
11    drawTriangle
12    (window, n, length, bottom + point, left + point, right + point);
13 }
14 void drawTriangle
15 (sf::RenderTarget* window, int n, double length,
16 sf::Vector2f bottom, sf::Vector2f tl, sf::Vector2f tr) {
17     sf::ConvexShape triangle(3);
18     triangle.setFillColor(sf::Color(0xFFD700FF));
19     triangle.setPoint(0, bottom);
20     triangle.setPoint(1, tl);
21     triangle.setPoint(2, tr);
22     window->draw(triangle);
23 if (n <= 0) {
24     return;
25 } else {
26     double newLength = length/2;
27     float h = sqrt(3) / 2 * newLength;
28     sf::Vector2f top_left_top_left =
29     {static_cast<float>(tl.x - newLength/2), tl.y - h};
30     // the top left corner of the new top left triangle
31     sf::Vector2f top_left_top_right =
32     {static_cast<float>(tl.x + newLength/2), tl.y - h};
33     // the top right corner of the new top left triangle
34     sf::Vector2f top_right_top_right =
35     {static_cast<float>(tr.x + newLength), tr.y};
36     // the top right corner of the new top right triangle
37     sf::Vector2f top_right_bottom =
38     {static_cast<float>(tr.x + newLength/2), tr.y + h};
39     // the top right corner of the new top right triangle
40     sf::Vector2f bottom_top_left =
41     {static_cast<float>(bottom.x - newLength), bottom.y};
42     // the top left corner of the new bottom triangle
43     sf::Vector2f bottom_bottom =
44     {static_cast<float>(bottom.x - newLength/2), bottom.y + h};
45     // the bottom corner of the new bottom triangle
46     drawTriangle
47     (window, n - 1, newLength, tl, top_left_top_left, top_left_top_right
48 );
49     drawTriangle
50     (window, n - 1, newLength, top_right_bottom, tr, top_right_top_right
51 );
52     drawTriangle
53     (window, n - 1, newLength, bottom_bottom, bottom_top_left, bottom);
54 }

```

3.6 Output



## 4 PS3: N-Body Simulation

### 4.1 Discussion

The goal of this project was to create a program that can draw and then simulate the physics of a universe of N bodies when given the necessary information about the universe. The information about the universe and the time that the program should run for is provided by the user through standard input. Drawing the universe is done by utilizing the SFML draw function along with sprites to be able to draw the planets and update their position as time passes. The physics calculations and updates to the sprites positioning are done at an interval chosen by the user. The larger the interval the faster the simulation will be. At the end of the simulation each bodies position, velocity, and mass are printed.

### 4.2 Key Algorithms Used

The most difficult calculations for this project were the calculations necessary to derive the positions of the planets after a certain period of time had passed. The first step was to identify the largest body in the universe. Then for each body in the Universe many values are calculated. The distance between them is found using the distance formula. The total force is calculated using newtons law of gravitation. The horizontal and vertical forces are calculated using force components. The directional accelerations are found by dividing directional force by mass. The new positions of the planet are found by adding velocity multiplied by the step time and one half the acceleration times the step time squared to the original positions. The updated velocity is found by adding the acceleration times the step time to the current velocity.

### 4.3 What I learned

This project required a more advanced utilization of SFML's draw function. Getting the planets to update in real time was something that I had not even considered using SFML for. I also learned how to create a static library through the makefile.

### 4.4 Challenges

The most difficult parts of this project for me were smart pointers and updating the planets locations correctly. I was not able to implement smart pointers in this project. Instead I just loaded the textures as part of the constructor stored them as sprites. The issues with updating the planets positions were due to mistakes in the formulas. The large amount of variables that must be calculated led to there being a couple bugs in the calculations.

### 4.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
      lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = Universe.hpp CelestialBody.hpp
6 # Your compiled .o files
7 OBJECTS = Universe.o CelestialBody.o
8 # The name of your program
9 PROGRAM = NBody
10 STATIC_LIB = NBody.a
11 # Test files and test executable
12 TEST_FILE = test.cpp
13 TEST_EXEC = test
14
15 .PHONY: all clean lint test
16
17 # Default target builds the main program and the test executable
```



```

18 all: $(PROGRAM) $(TEST_EXEC)
19
20 # Wildcard recipe to make .o files from corresponding .cpp file
21 %.o: %.cpp $(DEPS)
22     $(CC) $(CFLAGS) -c $<
23
24 $(STATIC_LIB): $(OBJECTS)
25     ar rcs $@ $^
26
27 $(PROGRAM): main.o $(STATIC_LIB)
28     $(CC) $(CFLAGS) -o $@ main.o $(STATIC_LIB) $(LIB)
29
30 $(TEST_EXEC): $(TEST_FILE) $(STATIC_LIB)
31     $(CC) $(CFLAGS) -o $@ $(TEST_FILE) $(STATIC_LIB) $(LIB)
32
33 clean:
34     rm -f *.o $(PROGRAM) $(STATIC_LIB) $(TEST_EXEC)
35
36 lint:
37     cpplint *.cpp *.hpp

```

```

1  // Copyright 2025 Brendan Kelleher
2  #include <iostream>
3  #include <fstream>
4  #include <vector>
5  #include <sstream>
6  #include <SFML/Graphics.hpp>
7  #include "../Universe.hpp"
8  #include "../CelestialBody.hpp"
9
10 int main(int argc, char* argv[]) {
11     double T = std::stod(argv[1]);
12     double delta_t = std::stod(argv[2]);
13     double t = 0;
14     NB::Universe universe;
15
16     std::cin >> universe;
17     sf::RenderWindow window(sf::VideoMode(1200, 1000), "Solar System");
18
19     // Move the window to position (0, 0) after creation
20     window.setPosition(sf::Vector2i(0, 0));
21     sf::View view(sf::FloatRect(0, 0, 1200, 1000));
22     // Center the view on the first celestial body
23     view.setCenter(0, 0);
24     window.setView(view);
25     window.setPosition({0, 0});
26     while (window.isOpen() && t < T) {
27         sf::Event event;
28         while (window.pollEvent(event)) {
29             if (event.type == sf::Event::Closed)
30                 window.close();
31         }
32
33         universe.step(delta_t);
34         t += delta_t;
35
36         window.clear();
37         window.draw(universe);
38         window.display();
39     }

```

```

40     std::cout << "Universe Data:\n" << universe;
41     return 0;
42 }

```

```

1  // Copyright 2025 Brendan Kelleher
2  #pragma once
3
4  #include <iostream>
5  #include <string>
6  #include <SFML/Graphics.hpp>
7
8
9  namespace NB {
10 class Universe;
11
12 class CelestialBody: public sf::Drawable {
13 public:
14     explicit CelestialBody();
15
16     sf::Vector2f position() const { return sf::Vector2f(x, y); }
17     sf::Vector2f velocity() const {return sf::Vector2f(x_velo, y_velo);}
18     float mass() const {return _mass;}
19     void setUniverse(Universe* uni) {universe = uni; }
20     void setPosition(sf::Vector2f V) {x = V.x; y = V.y;}
21     void setVelocity(sf::Vector2f V) {x_velo = V.x; y_velo = V.y;}
22     virtual ~CelestialBody() = default;
23
24 protected:
25     void draw(sf::RenderTarget& window,
26               sf::RenderStates states) const override;
27
28 private:
29     float x;
30     float y;
31     float x_velo;
32     float y_velo;
33     float _mass;
34     std::string image;
35     Universe* universe;
36     friend std::istream& operator>>(std::istream& is, CelestialBody& uni);
37     friend std::ostream& operator<<(std::ostream& os, const CelestialBody&
38 uni);
39 } // namespace NB

```

```

1  // Copyright 2025 Brendan Kelleher
2  #include<iostream>
3  #include<cmath>
4  #include<unordered_map>
5  #include "../Universe.hpp"
6  #include "CelestialBody.hpp"
7
8  namespace NB {
9  void CelestialBody::draw
10 (sf::RenderTarget& window, sf::RenderStates states) const {
11     if (!universe) {
12         std::cerr << "Universe pointer is null!" << std::endl;
13         return;
14     }
15     sf::Texture texture;

```

```

16     if (!texture.loadFromFile("nbody/nbody/" + image)) {
17         std::cerr << "Error loading texture: "
18         << "nbody/nbody/" + image << std::endl;
19         return;
20     }
21     sf::Sprite sprite(texture);
22     // scale x and y values to window size
23     float scaled_x = (x/universe->radius()) * 600;
24     float scaled_y = -(y/universe->radius()) * 600;
25     // Set the sprite's position to the celestial body's position (x, y)
26     sprite.setPosition(scaled_x, scaled_y);
27     window.draw(sprite, states);
28 }
29 CelestialBody::CelestialBody() {
30     x = 0;
31     y = 0;
32     x_velo = 0;
33     y_velo = 0;
34     _mass = 0.0f;
35     image = "";
36     universe = nullptr;
37 }
38
39
40 std::istream& operator>>(std::istream& is, CelestialBody& uni) {
41     is >> uni.x >> uni.y >> uni.x_velo >> uni.y_velo >> uni._mass >> uni.
42     image;
43     return is;
44 }
45
46 std::ostream& operator<<(std::ostream& os, const CelestialBody& uni) {
47     os << uni.x << " " << uni.y << " "
48     << uni.x_velo << " " << uni.y_velo << " "
49     << uni._mass << " " << uni.image;
50     return os;
51 }
52 // namespace NB

```

```

1 // Copyright 2025 Brendan Kelleher
2 #pragma once
3
4 #include <iostream>
5 #include<string>
6 #include<vector>
7 #include <SFML/Graphics.hpp>
8
9 #include "CelestialBody.hpp"
10
11 namespace NB {
12 class Universe: public sf::Drawable {
13 public:
14     Universe(); // Required
15     explicit Universe(const std::string& filename); // Optional
16
17     size_t size() const {return _size;} // Optional
18     double radius() const {return _radius;} // Optional
19     void push_back(CelestialBody body) {bodies.push_back(body);}
20     const CelestialBody& operator[](size_t i)
21     const {return bodies[i];} // Optional
22 }

```

```

23 // Implemented in part b, behavior for part a is undefined
24 void step(double dt);
25
26 virtual ~Universe() = default;
27
28 protected:
29     void draw(sf::RenderTarget& window, sf::RenderStates states)
30     const override; // From sf::Drawable
31
32 private:
33     // Fields and helper functions go here
34     size_t _size;
35     double _radius;
36     std::vector<CelestialBody> bodies;
37
38     friend std::istream& operator>>(std::istream& is, Universe& uni);
39     friend std::ostream& operator<<(std::ostream& os, const Universe& uni);
40 };
41
42
43
44 } // namespace NB

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <iostream>
3 #include <fstream>
4 #include <vector>
5 #include <sstream>
6 #include <cmath>
7 #include "Universe.hpp"
8 namespace NB {
9     Universe::Universe() {
10         _size = 0;
11         _radius = 0;
12         bodies = {};
13     }
14     std::istream& operator>>(std::istream& is, Universe& uni) {
15         CelestialBody body;
16         is >> uni._size >> uni._radius;
17         while (is >> body) {
18             uni.bodies.push_back(body);
19             uni.bodies.back().setUniverse(&uni);
20         }
21         return is;
22     }
23     std::ostream& operator<<(std::ostream& os, const Universe& uni) {
24         os << uni._size << " " << uni._radius << std::endl;
25         for (const auto& body : uni.bodies) {
26             os << body << "\n";
27         }
28         return os;
29     }
30     void Universe::draw(sf::RenderTarget& window, sf::RenderStates states) const
31     {
32         for (int i = 0; i < static_cast<int>(bodies.size()); i++) {
33             window.draw(bodies[i]);
34         }
35     }
36     void Universe::step(double dt) {
37         double total_force, force_x, force_y, distance_between;

```

```

37     float new_x, new_y, largest_new_y, largest_new_x,
38     accel_x, accel_y, largest_accel_y, largest_accel_x;
39     double G = 6.6674e-11;
40     CelestialBody* largest = &bodies[0];
41     for (CelestialBody& body : bodies) {
42         if (body.mass() > largest->mass())
43             largest = &body;
44     }
45     for (CelestialBody& body : bodies) {
46         if (&body == largest) continue;
47         distance_between =
48             sqrt(pow(largest->position().x - body.position().x, 2)
49                 + pow(largest->position().y - body.position().y, 2));
50         if (distance_between < 1e-5) continue;
51         total_force = (G * body.mass() *
52             largest->mass()) / pow(distance_between, 2);
53         force_x = total_force *
54             (largest->position().x - body.position().x) / distance_between;
55         force_y = total_force *
56             (largest->position().y - body.position().y) / distance_between;
57         accel_x = force_x / body.mass();
58         accel_y = force_y / body.mass();
59         largest_accel_x = -(force_x / largest->mass());
60         largest_accel_y = -(force_y / largest->mass());
61         new_x = body.position().x + body.velocity().x
62             * dt + 0.5* accel_x * pow(dt, 2);
63         new_y = body.position().y + body.velocity().y
64             * dt + 0.5* accel_y * pow(dt, 2);
65         largest_new_x = largest->position().x + largest->velocity().x
66             * dt + 0.5* largest_accel_x * pow(dt, 2);
67         largest_new_y = largest->position().y + largest->velocity().y
68             * dt + 0.5* largest_accel_y * pow(dt, 2);
69         body.setPosition({new_x, new_y});
70         largest->setPosition({largest_new_x, largest_new_y});
71         body.setVelocity({static_cast<float>(body.velocity().x + accel_x *
72             dt),
73             static_cast<float>(body.velocity().y + accel_y * dt)});
74         largest->setVelocity({static_cast<float>(largest->velocity().x +
75             largest_accel_x * dt),
76             static_cast<float>(largest->velocity().y + largest_accel_y * dt)
77     });
78 }
79 } // namespace NB

```

```

1 // Copyright 2025 Brendan Kelleher
2 #define BOOST_TEST_MODULE NBodyTest
3 #include <sstream>
4 #include<string>
5 #include <boost/test/included/unit_test.hpp>
6 #include "../CelestialBody.hpp"
7 #include "../Universe.hpp"
8
9 namespace NB {
10 BOOST_AUTO_TEST_CASE(InputStreamOperatorTest) {
11     std::string input = "100.0 200.0 10.0 15.0 5.97e24 earth.png";
12     std::istringstream inputStream(input);
13     CelestialBody body;
14     inputStream >> body;
15     BOOST_CHECK_EQUAL(body.position().x, 100.0f);

```

```

16     BOOST_CHECK_EQUAL(body.position().y, 200.0f);
17     BOOST_CHECK_EQUAL(body.velocity().x, 10.0f);
18     BOOST_CHECK_EQUAL(body.velocity().y, 15.0f);
19     BOOST_CHECK_CLOSE(body.mass(), 5.97e24f, 1e20);
20 }
21 BOOST_AUTO_TEST_CASE(test_operator_output_stream) {
22     std::string input = "100.0 200.0 10.0 15.0 5.0 planet.png";
23     std::istringstream inputStream(input);
24     CelestialBody body;
25     inputStream >> body;
26
27     // Act: Use a stringstream to capture the output of operator<<
28     std::ostringstream oss;
29     oss << body;
30
31     // Assert: Check if the output is as expected
32     std::string expected_output = "100 200 10 15 5 planet.png";
33     BOOST_CHECK_EQUAL(oss.str(), expected_output);
34 }
35 BOOST_AUTO_TEST_CASE(test_step) {
36     std::string input =
37         "5\n"
38         "2.50e+11\n"
39         "1.4960e+11 2.0000e+00 0.0000e+00 2.9800e+04 5.9740e+24 earth.gif\n"
40         "2.2790e+11 6.0250e+08 -6.3860e+01 2.4100e+04 6.4190e+23 mars.gif\n"
41         "5.7875e+10 1.1975e+09 -9.8933e+02 4.7900e+04 3.3020e+23 mercury.gif\n"
42         "0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 1.9890e+30 sun.gif\n"
43         "1.0819e+11 8.7500e+08 -2.8329e+02 3.5000e+04 4.8690e+24 venus.gif\n";
44
45     std::istringstream inputStream(input);
46     Universe universe;
47     inputStream >> universe;
48     double dt = 25000;
49     universe.step(dt);
50
51
52     BOOST_CHECK_CLOSE(universe[0].velocity().x, -1.4820e+02, 0.1);
53     BOOST_CHECK_CLOSE(universe[3].velocity().x, 1.3235e-03, 0.1);
54     BOOST_CHECK_CLOSE(universe[0].velocity().y, 2.9800e+04, 0.1);
55     BOOST_CHECK_CLOSE(universe[0].position().y, 7.4500e+08, 0.1);
56     dt = 0;
57     universe.step(dt);
58     BOOST_CHECK_CLOSE(universe[0].velocity().x, -1.4820e+02, 0.1);
59     BOOST_CHECK_CLOSE(universe[3].velocity().x, 1.3235e-03, 0.1);
60 }
61
62 } // namespace NB

```

## 4.6 Output

```

Universe Data:
5 2.5e+11
-1.70168e+11 -1.31716e+10 1986.59 -27774.2 5.974e+24 earth.gif
-2.04299e+11 -4.20010e+11 11867.4 -20461 6.419e+23 mars.gif
-5.54517e+10 1.03551e+11 -30489.2 -15320.4 3.302e+23 mercury.gif
6.91311e+06 1.20896e+08 -0.011366 0.358066 1.989e+30 sun.gif
-1.49780e+11 -1.5027e+10 2807.65 -30460.2 4.869e+24 venus.gif

```

## 5 PS4: Sokoban

### 5.1 Discussion

The goal of this project was to create a program that can draw and then allow the user to play a game of Sokoban. The program takes a file name from standard input, this file contains the necessary information for the game state in the form of characters in a grid. Using the characters in the text file allow the game board to be built by assigning each tile on the grid a sprite based on the corresponding character and then drawing all the sprites. To implement movement I used event listeners waiting for a keystroke of w, a, s, d, or the arrow keys and then changed the characters in the game board to update the game state. If the player attempts to move into a wall, or into a box that has a wall behind it the player is unable to move. This can be implemented by checking the characters in the current game state in the direction the player is attempting to move. To know if the player has won, every time the player inputs a valid move the game checks if either there are no empty storages, or if there are no boxes not on storages. If one of these conditions are met a win message is output.

### 5.2 Key Algorithms Used

A vital decision while working on this project was choosing how to store the characters that are provided through standard input and represent the gamestate. I chose to store them as a vector of characters. To populate this vector I used an overloaded extraction operation that read from standard input and based on the height, width, and characters, a nested for loop is used to assign each character to the correct index in the vector. In order to find the correct index for a given character, you multiply the y value by the width of the current map and add the x value. This method of finding the index is used heavily in the movePlayer function to be able to assess each tile in the grid.

### 5.3 What I learned

This project helped to further improve my skills with SFML's draw function. It also helped me learn about debugging a game with different conditions and moving parts. The amount of special cases and quirky things that happened while I was building the project helped me to build my debugging skills.

### 5.4 Challenges

When building this project my first struggle came when trying to scale the sprites to the right size and aligning the gameboard correctly in the window correctly. I chose to scale the sprites based on the width and height of the gameboard. To align the camera I adjusted the viewpoint so the top left tile in the grid lined up with the top left corner of the window. While implementing the logic for the game I faced a multitude of bugs that arose due to lack of proper checking for special cases.

### 5.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = Sokoban.hpp
6 # Your compiled .o files
7 OBJECTS = Sokoban.o
8 # The name of your program
9 PROGRAM = Sokoban
10 STATIC_LIB = Sokoban.a
11 TEST_FILE = test.cpp
12 TEST_EXEC = test
```



```

13
14 .PHONY: all clean lint test
15
16
17 all: $(PROGRAM) $(TEST_EXEC)
18
19 # Wildcard recipe to make .o files from corresponding .cpp file
20 %.o: %.cpp $(DEPS)
21     $(CC) $(CFLAGS) -c $<
22
23 $(STATIC_LIB): $(OBJECTS)
24     ar rcs $@ $^
25 $(PROGRAM): main.o $(OBJECTS) $(STATIC_LIB)
26     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
27
28 $(TEST_EXEC): $(TEST_FILE) $(STATIC_LIB)
29     $(CC) $(CFLAGS) -o $@ $(TEST_FILE) $(STATIC_LIB) $(LIB)
30 clean:
31     rm *.o $(PROGRAM)
32
33 lint:
34     cpplint *.cpp *.hpp

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <iostream>
3 #include<string>
4 #include <fstream>
5 #include<cmath>
6 #include "./Sokoban.hpp"
7 int main(int argc, char* argv[]) {
8     SB::Sokoban S;
9     std::string filename = argv[1];
10    std::ifstream file("sokoban/" + filename);
11    if (!file.is_open()) {
12        std::cerr << "Error opening file: " << filename << std::endl;
13        return 1;
14    }
15    file >> S;
16    std::cout << S;
17    int windowHeight = S.width() * 50;
18    int windowWidth = S.height() * 50;
19    sf::RenderWindow window(sf::VideoMode(1000, 1000), "Sokoban");
20    sf::View view(sf::FloatRect(0, 0, windowWidth, windowHeight));
21    view.setCenter((windowWidth / 2), (windowHeight/2));
22    window.setView(view);
23    while (window.isOpen()) {
24        sf::Event event;
25        while (window.pollEvent(event)) {
26            if (event.type == sf::Event::Closed) {
27                window.close();
28            }
29            if (event.type == sf::Event::KeyPressed) {
30                if (event.key.code == sf::Keyboard::W ||
31                    event.key.code == sf::Keyboard::Up) {
32                    S.movePlayer(SB::Direction::Up);
33                }
34                if (event.key.code == sf::Keyboard::A ||
35                    event.key.code == sf::Keyboard::Left) {
36                    S.movePlayer(SB::Direction::Left);
37                }

```

```

38         if (event.key.code == sf::Keyboard::S ||
39             event.key.code == sf::Keyboard::Down) {
40             S.movePlayer(SB::Direction::Down);
41         }
42         if (event.key.code == sf::Keyboard::D ||
43             event.key.code == sf::Keyboard::Right) {
44             S.movePlayer(SB::Direction::Right);
45         }
46         if (event.key.code == sf::Keyboard::R) {
47             S.reset();
48         }
49     }
50 }
51 window.clear();
52 window.draw(S);
53 window.display();
54 }
55 file.close();
56 return 0;
57 }

```

```

1  // Copyright 2025 Brendan Kelleher
2  #pragma once
3
4  #include <iostream>
5  #include <vector>
6  #include <SFML/Graphics.hpp>
7
8  namespace SB {
9      enum class Direction {
10         Up, Down, Left, Right
11     };
12
13     class Sokoban : public sf::Drawable {
14     public:
15         static const int TILE_SIZE = 64;
16
17         Sokoban();
18         // Sokoban(const std::string&); // Optional
19
20         unsigned int pixelHeight() const; // Optional
21         unsigned int pixelWidth() const; // Optional
22
23         unsigned int height() const;
24         unsigned int width() const;
25
26         sf::Vector2u playerLoc() const;
27
28         bool isWon() const;
29         void movePlayer(Direction dir);
30         void reset();
31
32         void undo(); // Optional XC
33         void redo(); // Optional XC
34
35     protected:
36         void draw(sf::RenderTarget& target, sf::RenderStates states) const
37             override;
38     private:

```

```

39     // Any fields you need go here.
40 friend std::ostream& operator<<(std::ostream& out, const Sokoban& s);
41 friend std::istream& operator>>(std::istream& in, Sokoban& s);
42 int map_height;
43 int map_width;
44 std::vector<char> gameState;
45 std::vector<char> origState;
46 sf::Texture wall;
47 sf::Texture box;
48 sf::Texture empty;
49 sf::Texture storage;
50 sf::Texture player_forward;
51 sf::Texture player_backward;
52 sf::Texture player_right;
53 sf::Texture player_left;
54 int numBox;
55 int numStore;
56 bool won;
57 };
58
59
60 } // namespace SB

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <iostream>
3 #include "/autograder/playground/Sokoban.hpp"
4 namespace SB {
5 Sokoban::Sokoban() {
6     map_height = 0;
7     map_width = 0;
8     won = false;
9     if (!wall.loadFromFile("sokoban/block_06.png")) {
10         std::cerr << "Error loading texture: " << std::endl;
11         return;
12     }
13     if (!box.loadFromFile("sokoban/crate_03.png")) {
14         std::cerr << "Error loading texture: " << std::endl;
15         return;
16     }
17     if (!empty.loadFromFile("sokoban/ground_01.png")) {
18         std::cerr << "Error loading texture: " << std::endl;
19         return;
20     }
21     if (!storage.loadFromFile("sokoban/ground_04.png")) {
22         std::cerr << "Error loading texture: " << std::endl;
23         return;
24     }
25     if (!player_forward.loadFromFile("sokoban/player_05.png")) {
26         std::cerr << "Error loading texture: " << std::endl;
27         return;
28     }
29     if (!player_backward.loadFromFile("sokoban/player_08.png")) {
30         std::cerr << "Error loading texture: " << std::endl;
31         return;
32     }
33     if (!player_right.loadFromFile("sokoban/player_17.png")) {
34         std::cerr << "Error loading texture: " << std::endl;
35         return;
36     }
37     if (!player_left.loadFromFile("sokoban/player_20.png")) {

```

```

38         std::cerr << "Error loading texture: " << std::endl;
39         return;
40     }
41 }
42 std::istream& operator>>(std::istream& in, Sokoban& s) {
43     in >> s.map_height >> s.map_width;
44     s.gameState.resize(s.map_height * s.map_width);
45     for (int i = 0; i < s.map_height; i++) {
46         for (int j = 0; j < s.map_width; j++) {
47             int index = i * s.map_width + j;
48             in >> s.gameState[index];
49         }
50     }
51     s.origState.resize(s.map_height * s.map_width);
52     for (int x = 0; x < s.map_width; x++) {
53         for (int y = 0; y < s.map_height; y++) {
54             int index = y * s.map_width + x;
55             s.origState[index] = s.gameState[index];
56             if (s.gameState[index] == 'A') {
57                 s.numBox++;
58             }
59             if (s.gameState[index] == 'a') {
60                 s.numStore++;
61             }
62         }
63     }
64     return in;
65 }
66 std::ostream& operator<<(std::ostream& out, const Sokoban& s) {
67     out << s.map_height << " " << s.map_width << std::endl;
68     for (int i = 0; i < s.map_height; i++) {
69         for (int j = 0; j < s.map_width; j++) {
70             int index = i * s.map_width + j;
71             out << s.gameState[index];
72         }
73         out << std::endl;
74     }
75     return out;
76 }
77 void Sokoban::draw(sf::RenderTarget& target, sf::RenderStates states) const
78 {
79     sf::Sprite wall_sprite(wall);
80     sf::Sprite box_sprite(box);
81     sf::Sprite empty_sprite(empty);
82     sf::Sprite storage_sprite(storage);
83     sf::Sprite player_sprite(player_forward);
84     sf::Vector2u textureSize = wall.getSize();
85     float desiredWidth = 50.0f;
86     float desiredHeight = 50.0f;
87     float scaleX = desiredWidth / textureSize.x;
88     float scaleY = desiredHeight / textureSize.y;
89     wall_sprite.setScale(scaleX, scaleY);
90     box_sprite.setScale(scaleX, scaleY);
91     empty_sprite.setScale(scaleX, scaleY);
92     storage_sprite.setScale(scaleX, scaleY);
93     player_sprite.setScale(scaleX, scaleY);
94     for (int i = 0; i < map_height; i++) {
95         for (int j = 0; j < map_width; j++) {
96             int index = i * map_width + j;

```

```

96         switch (gameState[index]) {
97             case '@':
98                 empty_sprite.setPosition(j * desiredWidth, i * desiredHeight)
99                 ;
100                 target.draw(empty_sprite, states);
101                 player_sprite.setPosition(j * desiredWidth, i * desiredHeight
102             );
103                 target.draw(player_sprite, states);
104                 break;
105             case '.':
106                 empty_sprite.setPosition(j * desiredWidth, i * desiredHeight)
107                 ;
108                 target.draw(empty_sprite, states);
109                 break;
110             case '#':
111                 wall_sprite.setPosition(j * desiredWidth, i * desiredHeight);
112                 target.draw(wall_sprite, states);
113                 break;
114             case 'A':
115                 empty_sprite.setPosition(j * desiredWidth, i * desiredHeight)
116                 ;
117                 target.draw(empty_sprite, states);
118                 box_sprite.setPosition(j * desiredWidth, i * desiredHeight);
119                 target.draw(box_sprite, states);
120                 break;
121             case 'a':
122                 storage_sprite.setPosition(j * desiredWidth, i *
123             desiredHeight);
124                 target.draw(storage_sprite, states);
125                 break;
126             case 'b':
127                 storage_sprite.setPosition(j * desiredWidth, i *
128             desiredHeight);
129                 target.draw(storage_sprite, states);
130                 player_sprite.setPosition(j * desiredWidth, i * desiredHeight
131             );
132                 target.draw(player_sprite, states);
133                 break;
134             case '1':
135                 storage_sprite.setPosition(j * desiredWidth, i *
136             desiredHeight);
137                 target.draw(storage_sprite, states);
138                 box_sprite.setPosition(j * desiredWidth, i * desiredHeight);
139                 target.draw(box_sprite, states);
140                 break;
141         }
142     }
143 }
144 unsigned int Sokoban::height() const {
145     return map_height;
146 }
147 unsigned int Sokoban::width() const {
148     return map_width;
149 }
150 sf::Vector2u Sokoban::playerLoc() const {
151     sf::Vector2u playerLoc;
152     for (int y = 0; y < map_height; y++) {
153         for (int x = 0; x < map_width; x++) {

```

```

147         int index = y * map_width + x;
148         if (gameState[index] == '@' || gameState[index] == 'b') {
149             playerLoc.y = y;
150             playerLoc.x = x;
151         }
152     }
153 }
154 return playerLoc;
155 }
156 bool Sokoban::isWon() const {
157     if (numBox >= numStore) {
158         for (int i = 0; i < map_height; i++) {
159             for (int j = 0; j < map_width; j++) {
160                 int index = i * map_width + j;
161                 if (gameState[index] == 'a' || gameState[index] == 'b') {
162                     return false;
163                 }
164             }
165         }
166     } else {
167         for (int i = 0; i < map_height; i++) {
168             for (int j = 0; j < map_width; j++) {
169                 int index = i * map_width + j;
170                 if (gameState[index] == 'A' || gameState[index] == 'b') {
171                     return false;
172                 }
173             }
174         }
175     }
176     std::cout << "You won!" << std::endl;
177     return true;
178 }
179 void Sokoban::movePlayer(Direction dir) {
180     sf::Vector2u playerPos = playerLoc();
181     int playerIndex = playerPos.y * map_width + playerPos.x;
182     int offset = 0;
183     switch (dir) {
184         case Direction::Right: offset = 1;
185             if ((playerIndex + 1) % map_width == 0) return;
186             break;
187         case Direction::Left: offset = -1;
188             if (playerIndex % map_width == 0) return;
189             break;
190         case Direction::Up: offset = -width();
191             if (playerIndex < map_width) return;
192             break;
193         case Direction::Down: offset = width();
194             if (playerIndex >= map_width * (map_height - 1)) return;
195             break;
196     }
197     int newIndex = playerIndex + offset;
198     int newBoxIndex = newIndex + offset;
199     if (newBoxIndex >= static_cast<int>(width()) * static_cast<int>(height())
200 ) {
201         if (gameState[newIndex] == 'A' || gameState[newIndex] == '1') {
202             return;
203         }
204     }
205     if (isWon() || gameState[newIndex] == '#') {

```

```

205     return;
206 }
207 if (gameState[newIndex] == 'A') {
208     if ((dir == Direction::Left || dir == Direction::Right) &&
209         (newBoxIndex / map_width != newIndex / map_width)) return;
210     if (gameState[newBoxIndex] != 'A' && gameState[newBoxIndex] != '#'
211         && gameState[newBoxIndex] != '1') {
212         if (gameState[newBoxIndex] == 'a') {
213             gameState[newBoxIndex] = '1';
214         } else {
215             gameState[newBoxIndex] = 'A';
216         }
217         gameState[newIndex] = '@';
218         if (gameState[playerIndex] == 'b') {
219             gameState[playerIndex] = 'a';
220         } else {
221             gameState[playerIndex] = '.';
222         }
223         isWon();
224     }
225     return;
226 } else if (gameState[newIndex] == '1') {
227     if ((dir == Direction::Left || dir == Direction::Right) &&
228         (newBoxIndex / map_width != newIndex / map_width)) {
229         return;
230     }
231     if (gameState[newBoxIndex] != '#' && gameState[newBoxIndex] != 'A')
232     {
233         if (gameState[newBoxIndex] == 'a') {
234             gameState[newBoxIndex] = '1';
235         } else {
236             gameState[newBoxIndex] = 'A';
237         }
238         gameState[newIndex] = 'b';
239         if (gameState[playerIndex] == 'b') {
240             gameState[playerIndex] = 'a';
241         } else {
242             gameState[playerIndex] = '.';
243         }
244         isWon();
245     }
246 } else if (gameState[newIndex] == '.' ||
247     gameState[newIndex] == 'a' ||
248     gameState[newIndex] == 'b') {
249     if (gameState[newIndex] == 'a') {
250         gameState[newIndex] = 'b';
251     } else {
252         gameState[newIndex] = '@';
253     }
254     if (gameState[playerIndex] == 'b') {
255         gameState[playerIndex] = 'a';
256     } else {
257         gameState[playerIndex] = '.';
258     }
259     isWon();
260 }
261 void Sokoban::reset() {
262     for (int x = 0; x < static_cast<int>(width()); x++) {

```

```

263     for (int y = 0; y < static_cast<int>(height()); y++) {
264         int index = y * map_width + x;
265         gameState[index] = origState[index];
266     }
267 }
268 }
269 } // namespace SB

```

```

1  // Copyright 2025 Brendan Kelleher
2  #define BOOST_TEST_MODULE NBodyTest
3  #include <sstream>
4  #include<string>
5  #include <boost/test/included/unit_test.hpp>
6  #include "Sokoban.hpp"
7
8
9  namespace SB {
10 BOOST_AUTO_TEST_CASE(MovementTest) {
11     Sokoban S;
12     std::ifstream file("sokoban/level1.lvl");
13     file >> S;
14     BOOST_CHECK_EQUAL(S.playerLoc().x, 3);
15     BOOST_CHECK_EQUAL(S.playerLoc().y, 6);
16     S.movePlayer(Direction::Left);
17     BOOST_CHECK_EQUAL(S.playerLoc().x, 2);
18     BOOST_CHECK_EQUAL(S.playerLoc().y, 6);
19     S.movePlayer(Direction::Up);
20     BOOST_CHECK_EQUAL(S.playerLoc().x, 2);
21     BOOST_CHECK_EQUAL(S.playerLoc().y, 5);
22     S.movePlayer(Direction::Right);
23     BOOST_CHECK_EQUAL(S.playerLoc().x, 3);
24     BOOST_CHECK_EQUAL(S.playerLoc().y, 5);
25     S.movePlayer(Direction::Down);
26     BOOST_CHECK_EQUAL(S.playerLoc().x, 3);
27     BOOST_CHECK_EQUAL(S.playerLoc().y, 6);
28 }
29 BOOST_AUTO_TEST_CASE(ResetTest) {
30     Sokoban S;
31     std::ifstream file("sokoban/level1.lvl");
32     file >> S;
33     S.movePlayer(Direction::Right);
34     S.movePlayer(Direction::Right);
35     S.movePlayer(Direction::Right);
36     S.movePlayer(Direction::Right);
37     S.movePlayer(Direction::Up);
38     BOOST_CHECK_EQUAL(S.playerLoc().x, 7);
39     BOOST_CHECK_EQUAL(S.playerLoc().y, 5);
40     S.reset();
41     BOOST_CHECK_EQUAL(S.playerLoc().x, 3);
42     BOOST_CHECK_EQUAL(S.playerLoc().y, 6);
43 }
44 BOOST_AUTO_TEST_CASE(IsWonTest) {
45     Sokoban S;
46     std::ifstream file("sokoban/level1.lvl");
47     file >> S;
48     S.movePlayer(Direction::Right);
49     S.movePlayer(Direction::Right);
50     S.movePlayer(Direction::Right);
51     S.movePlayer(Direction::Right);
52     S.movePlayer(Direction::Up);

```



```

53     BOOST_CHECK_EQUAL(S.isWon(), false);
54     S.reset();
55     BOOST_CHECK_EQUAL(S.playerLoc().x, 3);
56     BOOST_CHECK_EQUAL(S.playerLoc().y, 6);
57     S.movePlayer(Direction::Right);
58     S.movePlayer(Direction::Right);
59     S.movePlayer(Direction::Right);
60     S.movePlayer(Direction::Right);
61     S.movePlayer(Direction::Up);
62     S.movePlayer(Direction::Right);
63     S.movePlayer(Direction::Down);
64     S.movePlayer(Direction::Up);
65     S.movePlayer(Direction::Up);
66     S.movePlayer(Direction::Up);
67     S.movePlayer(Direction::Left);
68     S.movePlayer(Direction::Left);
69     S.movePlayer(Direction::Left);
70     S.movePlayer(Direction::Up);
71     BOOST_CHECK_EQUAL(S.isWon(), true);
72 }
73 BOOST_AUTO_TEST_CASE(BoxesTest) {
74     Sokoban S;
75     std::ifstream file("sokoban/level1.lvl");
76     file >> S;
77     S.movePlayer(Direction::Down);
78     S.movePlayer(Direction::Right);
79     S.movePlayer(Direction::Right);
80     S.movePlayer(Direction::Right);
81     S.movePlayer(Direction::Up);
82     S.movePlayer(Direction::Up);
83     S.movePlayer(Direction::Up);
84     S.movePlayer(Direction::Up);
85     S.movePlayer(Direction::Right);
86     S.movePlayer(Direction::Up);
87     S.movePlayer(Direction::Left);
88     BOOST_CHECK_EQUAL(S.playerLoc().x, 7);
89     BOOST_CHECK_EQUAL(S.playerLoc().y, 2);
90     S.reset();
91     BOOST_CHECK_EQUAL(S.playerLoc().x, 3);
92     BOOST_CHECK_EQUAL(S.playerLoc().y, 6);
93     S.movePlayer(Direction::Right);
94     S.movePlayer(Direction::Right);
95     S.movePlayer(Direction::Right);
96     S.movePlayer(Direction::Right);
97     S.movePlayer(Direction::Right);
98     BOOST_CHECK_EQUAL(S.playerLoc().x, 7);
99     BOOST_CHECK_EQUAL(S.playerLoc().y, 6);
100 }
101 BOOST_AUTO_TEST_CASE(OffScreenTest) {
102     Sokoban S;
103     std::ifstream file("sokoban/pushup.lvl");
104     file >> S;
105     S.movePlayer(Direction::Left);
106     S.movePlayer(Direction::Up);
107     S.movePlayer(Direction::Right);
108     S.movePlayer(Direction::Right);
109     S.movePlayer(Direction::Right);
110     BOOST_CHECK_EQUAL(S.playerLoc().x, 3);
111     BOOST_CHECK_EQUAL(S.playerLoc().y, 1);

```

```

112     S.reset();
113     BOOST_CHECK_EQUAL(S.playerLoc().x, 2);
114     BOOST_CHECK_EQUAL(S.playerLoc().y, 2);
115     S.movePlayer(Direction::Right);
116     S.movePlayer(Direction::Right);
117     S.movePlayer(Direction::Right);
118     S.movePlayer(Direction::Right);
119     S.movePlayer(Direction::Right);
120     BOOST_CHECK_EQUAL(S.playerLoc().x, 4);
121     BOOST_CHECK_EQUAL(S.playerLoc().y, 2);
122 }
123 BOOST_AUTO_TEST_CASE(LotsOfBoxesTest) {
124     Sokoban S;
125     std::ifstream file("sokoban/level5.lvl");
126     file >> S;
127     S.movePlayer(Direction::Left);
128     S.movePlayer(Direction::Up);
129     S.movePlayer(Direction::Right);
130     S.movePlayer(Direction::Right);
131     S.movePlayer(Direction::Right);
132     BOOST_CHECK_EQUAL(S.isWon(), false);
133     S.reset();
134     S.movePlayer(Direction::Right);
135     S.movePlayer(Direction::Right);
136     S.movePlayer(Direction::Right);
137     S.movePlayer(Direction::Right);
138     S.movePlayer(Direction::Up);
139     S.movePlayer(Direction::Up);
140     S.movePlayer(Direction::Up);
141     S.movePlayer(Direction::Up);
142     S.movePlayer(Direction::Left);
143     S.movePlayer(Direction::Up);
144     S.movePlayer(Direction::Right);
145     BOOST_CHECK_EQUAL(S.isWon(), true);
146 }
147 BOOST_AUTO_TEST_CASE(LotsOfTargetsTest) {
148     Sokoban S;
149     std::ifstream file("sokoban/level6.lvl");
150     file >> S;
151     S.movePlayer(Direction::Right);
152     S.movePlayer(Direction::Up);
153     S.movePlayer(Direction::Up);
154     S.movePlayer(Direction::Up);
155     S.movePlayer(Direction::Up);
156     S.movePlayer(Direction::Right);
157     S.movePlayer(Direction::Up);
158     S.movePlayer(Direction::Left);
159     BOOST_CHECK_EQUAL(S.isWon(), false);
160     S.reset();
161     S.movePlayer(Direction::Right);
162     S.movePlayer(Direction::Up);
163     S.movePlayer(Direction::Up);
164     S.movePlayer(Direction::Up);
165     S.movePlayer(Direction::Up);
166     S.movePlayer(Direction::Right);
167     S.movePlayer(Direction::Up);
168     S.movePlayer(Direction::Left);
169     S.movePlayer(Direction::Down);
170     S.movePlayer(Direction::Down);

```

```
171 S.movePlayer(Direction::Down);
172 S.movePlayer(Direction::Down);
173 S.movePlayer(Direction::Right);
174 S.movePlayer(Direction::Right);
175 S.movePlayer(Direction::Right);
176 S.movePlayer(Direction::Up);
177 S.movePlayer(Direction::Right);
178 S.movePlayer(Direction::Down);
179 BOOST_CHECK_EQUAL(S.isWon(), true);
180 }
181 } // namespace SB
```

## 5.6 Output



## 6 PS5: DNA Alignment

### 6.1 Discussion

The goal of this project was to create a program that can calculate the optimal edit distance between two genetic sequences. The sequences consist of 4 letters, A, T, G and C. The program takes the sequences through standard input and constructs an EDistance object. The edit distance is found by aligning the two sequences and adding up the penalties that their mismatches cause. A gap in the sequence costs 2 points and mismatched letters costs 1 point. To find the optimal edit distance between two sequences an algorithm must be used. Once the optimal edit distance is found the alignment must be recovered. This is done by working in reverse through the sequences until both are recovered.

### 6.2 Key Algorithms Used

In order to find the optimal edit distance between two given sequences I chose to utilize dynamic programming. I used a vector of vectors serving as a matrix to hold the necessary values to calculate the optimal edit distance. Each dimension of the matrix represents one of the two sequences and their letters align with the indexes. The first values inputted to the matrix are those on the bottom and right edge as those are always the same. Then starting from the bottom-right most point in the vector each value is calculated. Each individual value is equal to the minimum of 3 options, the value to the bottom right plus the penalty between letters, the value to the right plus two or the value to the bottom plus two. Once all values are calculated the top left corner is equal to the optimal edit distance. To find the optimal alignment of these sequences requires going backwards through the matrix created solving for edit distance. Starting at the top left corner the program looks at the same three values that it did for edit distance and chooses the one that is the smallest and is a possible option based on the values present. Each time the program moves to the bottom-right through the array a character from each sequence and the penalty between them is recorded. If the program moves right or to the bottom a gap and a penalty of two is recorded.

### 6.3 What I learned

This project served as an introduction to dynamic programming. The ability to store values and use them for later like this seems that it could be very useful in the future. This project also required the use of valgrind which I have not used in a while and this project was a nice refresher.

### 6.4 Challenges

This project was mostly smooth sailing. The most difficult part for me was recovering the alignment. Building the conditions to correctly go back through the matrix and get the correct steps took some trial and error.

### 6.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = EDistance.hpp
6 # Your compiled .o files
7 OBJECTS = EDistance.o
8 # The name of your program
9 PROGRAM = EDistance
10 STATIC_LIB = EDistance.a
11 TEST_FILE = test.cpp
12 TEST_EXEC = test
13
```

```

14 .PHONY: all clean lint test
15
16
17 all: $(PROGRAM) $(TEST_EXEC)
18
19 # Wildcard recipe to make .o files from corresponding .cpp file
20 %.o: %.cpp $(DEPS)
21     $(CC) $(CFLAGS) -c $<
22
23 $(STATIC_LIB): $(OBJECTS)
24     ar rcs $@ $^
25 $(PROGRAM): main.o $(OBJECTS) $(STATIC_LIB)
26     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
27
28 $(TEST_EXEC): $(TEST_FILE) $(STATIC_LIB)
29     $(CC) $(CFLAGS) -o $@ $(TEST_FILE) $(STATIC_LIB) $(LIB)
30 clean:
31     rm *.o $(PROGRAM)
32
33 lint:
34     cpplint *.cpp *.hpp

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include<iostream>
3 #include<string>
4 #include <SFML/Graphics.hpp>
5 #include "./EDistance.hpp"
6 int main(int argc, char* argv[]) {
7     sf::Clock clock;
8     std::string input1;
9     std::string input2;
10    std::cin >> input1 >> input2;
11    EDistance E(input1, input2);
12    std::string output;
13    int editDist = E.optDistance();
14    output = E.alignment();
15    sf::Time t = clock.getElapsedTime();
16    std::cout << "Edit Distance = " << editDist << std::endl;
17    std::cout << output;
18    std::cout << "Execution time is " << t.asSeconds()
19        << " seconds" << std::endl;
20    return 0;
21 }

```

```

1 // Copyright 2025 Brendan Kelleher
2 #pragma once
3
4 #include <string>
5 #include <vector>
6 class EDistance {
7 public:
8     EDistance(const std::string& s1, const std::string& s2);
9
10    static int penalty(char a, char b);
11    static int min3(int a, int b, int c);
12
13    int optDistance();
14    std::string alignment();
15
16    void populateEdges();

```

```

17     void printMatrix();
18 private:
19 std::vector<std::vector<int>> matrix;
20 int s1_len;
21 int s2_len;
22 std::string string1 = "";
23 std::string string2 = "";
24 int edit_distance;
25 };

```

```

1  // Copyright 2025 Brendan Kelleher
2  #include <iostream>
3  #include<vector>
4  #include<string>
5  #include<algorithm>
6  #include "/autograder/playground/EDistance.hpp"
7  EDistance::EDistance(const std::string& s1, const std::string& s2) {
8      s1_len = s1.length();
9      s2_len = s2.length();
10     string1 = s1;
11     string2 = s2;
12
13     matrix.resize(s1_len + 1, std::vector<int>(s2_len + 1, 0));
14     for (int x = 0; x <= static_cast<int>(s1.length()); x++) {
15         for (int y = 0; y <= static_cast<int>(s2.length()); y++) {
16             matrix[x][y] = 0;
17         }
18     }
19 }
20 int EDistance::penalty(char a, char b) {
21     if (a == b) {
22         return 0;
23     }
24     return 1;
25 }
26 int EDistance::min3(int a, int b, int c) {
27     return std::min(a, std::min(b, c));
28 }
29 int EDistance::optDistance() {
30     populateEdges();
31     for (int y = s2_len-1; y >= 0; y--) {
32         for (int x = s1_len-1; x >= 0; x--) {
33             matrix[x][y] = min3(matrix[x+1][y+1] +
34                 penalty(string1[x], string2[y]), matrix[x+1][y] + 2,
35                 matrix[x][y+1] + 2);
36         }
37     }
38     edit_distance = matrix[0][0];
39     return matrix[0][0];
40 }
41 std::string EDistance::alignment() {
42     int x = 0;
43     int y = 0;
44     std::string alignOutput = "";
45     while (x < static_cast<int>(string1.length())
46         && y < static_cast<int>(string2.length())) {
47         if (matrix[x][y] == matrix[x+1][y+1] && min3(matrix[x+1][y+1]
48             + penalty(string1[x], string2[y]),
49             matrix[x+1][y] + 2, matrix[x][y+1] + 2)
50             == matrix[x+1][y+1] + penalty(string1[x], string2[y])) {

```

```

51     alignOutput += std::string(1, string1[x]) +
52     " " + std::string(1, string2[y]);
53     alignOutput += " 0\n";
54     x++;
55     y++;
56 } else if (matrix[x][y] == matrix[x+1][y+1]+1 && min3(matrix[x+1][y+1]
57     + penalty(string1[x], string2[y]), matrix[x+1][y] + 2,
58     matrix[x][y+1] + 2) == matrix[x+1][y+1] +
59     penalty(string1[x], string2[y])) {
60     alignOutput += std::string(1, string1[x]) +
61     " " + std::string(1, string2[y]);
62     alignOutput += " 1\n";
63     x++;
64     y++;
65 } else if (matrix[x][y] == matrix[x+1][y]+2 && min3(matrix[x+1][y+1]
66     + penalty(string1[x], string2[y]),
67     matrix[x+1][y] + 2, matrix[x][y+1] + 2)
68     == matrix[x+1][y]+2) {
69     alignOutput += std::string(1, string1[x]) + " -";
70     alignOutput += " 2\n";
71     x++;
72 } else if (matrix[x][y] == matrix[x][y+1]+2) {
73     alignOutput += "- " + std::string(1, string2[y]);
74     alignOutput += " 2\n";
75     y++;
76 }
77 }
78 while (x < static_cast<int>(string1.length())) {
79     alignOutput += std::string(1, string1[x]) + " -";
80     alignOutput += " 2\n";
81     x++;
82 }
83 while (y < static_cast<int>(string2.length())) {
84     alignOutput += "- " + std::string(1, string2[y]);
85     alignOutput += " 2\n";
86     y++;
87 }
88 return alignOutput;
89 }
90 void EDistance::populateEdges() {
91     int num = 0;
92     for (int x = s1_len; x >= 0; x--) {
93         matrix[x][s2_len] = num;
94         num += 2;
95     }
96     num = 0;
97     for (int y = s2_len; y >= 0; y--) {
98         matrix[s1_len][y] = num;
99         num += 2;
100     }
101 }
102 void EDistance::printMatrix() {
103     for (int x = 0; x <= static_cast<int>(s1_len); x++) {
104         for (int y = 0; y <= static_cast<int>(s2_len); y++) {
105             std::cout << matrix[x][y] << " ";
106             if (matrix[x][y] <= 9)
107                 std::cout << " ";
108         }
109         std::cout << std::endl;

```

```

110     }
111 }

1 // Copyright 2025 Brendan Kelleher
2 #define BOOST_TEST_MODULE NBodyTest
3 #include <sstream>
4 #include<string>
5 #include <boost/test/included/unit_test.hpp>
6 #include "../EDistance.hpp"
7
8
9
10 BOOST_AUTO_TEST_CASE(Min3Test) {
11     EDistance E("AACAGTTACC", "TAAGGTCA");
12     int num = E.min3(8, 5, 19);
13     BOOST_CHECK_EQUAL(num, 5);
14     num = E.min3(4, 5, 4);
15     BOOST_CHECK_EQUAL(num, 4);
16 }
17
18 BOOST_AUTO_TEST_CASE(PenaltyTest) {
19     EDistance E("AACAGTTACC", "TAAGGTCA");
20     int num = E.penalty('A', 'G');
21     BOOST_CHECK_EQUAL(num, 1);
22     num = E.penalty('A', 'A');
23     BOOST_CHECK_EQUAL(num, 0);
24 }
25 BOOST_AUTO_TEST_CASE(OptDistanceTest) {
26     EDistance E("AACAGTTACC", "TAAGGTCA");
27     int num = E.optDistance();
28     BOOST_CHECK_EQUAL(num, 7);
29 }
30 BOOST_AUTO_TEST_CASE(AlignmentTest) {
31     std::string ExpOutput =
32     "A T 1\nA A 0\nC - 2\nA A 0\nG G 0\nT G 1\nT T 0\nA - 2\nC C 0\nC A 1\n"
33     ;
34     std::string CutOutput =
35     "A T 1\nA A 0\nC - 2\nA A 0\nG G 0\nT G 1\nT T 0\nA - 2\nC C 0\nC";
36     EDistance E("AACAGTTACC", "TAAGGTCA");
37     E.optDistance();
38     std::string ActOutput = E.alignment();
39     BOOST_CHECK_EQUAL(ActOutput, ExpOutput);
40     BOOST_CHECK_EQUAL(ActOutput.length(), ExpOutput.length());
41     BOOST_CHECK_NE(ActOutput, CutOutput);
42 }

```

## 6.6 Output



```
Edit Distance = 7
A T 1
A A 0
C - 2
A A 0
G G 0
T G 1
T T 0
A - 2
C C 0
C A 1
Execution time is 0.00042 seconds
```

## 7 PS6: RandWriter

### 7.1 Discussion

The goal of this project was to create a program that can generate text based on a given input text. This is achieved by using a markov model, which involves tracking what characters come after each string of a given length and at what frequency. The order of the model affects how long the string used for comparison is and increases the overall accuracy of the model. To be able to implement this model, the program takes an order for the markov model from the user and extracts each individual string of that length from the given text. The program also tracks the frequency at which each character appears after each substring given. To acutally generate the text, a letter is genrated based on the curent substring. What this letter is is randomly determined based on the letters that followed that string in the text. So if the string was followed by three a's and 2 e's there is a 60 percent chance the next letter will be an a and 40 percent chance it will be a e. This character generation continues, changing the string each time a new letter is generated until the generated string reaches the user's desired length.

### 7.2 Key Algorithms Used

In order to utilize the markov model a data structure that could hold each unique substring in the text and the frequencies at which different letters follow them were needed. I chose to use an unordered map, with strings as the key, and another unordered map with character as the key and ints as the values for the values. The string keys are the unique substrings in the text and the unordered map of char and int allows each character to be connected to the amount of times it appears. To be able to determine what letter is generated each time one is needed the program finds the point in the map where the current string is the key and looks at all of the charcters and their frequencies and using random selection and discrete distribution one is chosen at random based on a percentage of how often they appear after the current string.

### 7.3 What I learned

This project allowed me to use a data structure that I do not have much experience with. Before this project I had a vague undestanding of maps, but I had never thought to use one as the value of another one. Buildng this project allowed me to see how potentially powerful this could be and broadened my programming horizons.

### 7.4 Challenges

This project was also mostly smooth sailing. The part that I struggled most with was accessing the contents of the map without potentially unwantingly changing them. I attempted to use direct access using keys to be able to access information but could not due to the freq functions both being const functions.

### 7.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = RandWriter.hpp
6 # Your compiled .o files
7 OBJECTS = RandWriter.o
8 # The name of your program
9 PROGRAM = TextWriter
10 STATIC_LIB = TextWriter.a
11 TEST_FILE = test.cpp
12 TEST_EXEC = test
```

```

13
14 .PHONY: all clean lint test
15
16
17 all: $(PROGRAM) $(TEST_EXEC)
18
19 # Wildcard recipe to make .o files from corresponding .cpp file
20 %.o: %.cpp $(DEPS)
21     $(CC) $(CFLAGS) -c $<
22
23 $(STATIC_LIB): $(OBJECTS)
24     ar rcs $@ $^
25 $(PROGRAM): TextWriter.o $(OBJECTS) $(STATIC_LIB)
26     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
27
28 $(TEST_EXEC): $(TEST_FILE) $(STATIC_LIB)
29     $(CC) $(CFLAGS) -o $@ $(TEST_FILE) $(STATIC_LIB) $(LIB)
30 clean:
31     rm *.o $(PROGRAM)
32
33 lint:
34     cpplint *.cpp *.hpp

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include<iostream>
3 #include<string>
4 #include <SFML/Graphics.hpp>
5 #include "./RandWriter.hpp"
6 int main(int argc, char* argv[]) {
7     try {
8         int k = atoi(argv[1]);
9         int l = atoi(argv[2]);
10        std::string inputText((std::istreambuf_iterator<char>(std::cin)),
11                               std::istreambuf_iterator<char>());
12        RandWriter R(inputText, k);
13        std::string kgram = inputText.substr(0, k);
14        std::string outputText = R.generate(kgram, l);
15        std::cout << outputText << std::endl;
16    }
17    catch(const std::invalid_argument& arg) {
18        std::cerr << arg.what() << std::endl;
19    }
20    return 0;
21 }

```

```

1 // Copyright 2025 Brendan Kelleher
2 #ifndef RANDWRITER_H
3 #define RANDWRITER_H
4 #include <string>
5 #include <stdexcept>
6 #include <random>
7 #include <unordered_map>
8 class RandWriter {
9     public:
10        // Create a Markov model of order k from given text
11        // Assume that text has length at least k.
12        RandWriter(const std::string& str, size_t k);
13
14        size_t orderK() const; // Order k of Markov model
15

```

```

16 // Number of occurrences of kgram in text
17 // Throw an exception if kgram is not length k
18 int freq(const std::string& kgram) const;
19 // Number of times that character c follows kgram
20 // if order=0, return num of times that char c appears
21 // (throw an exception if kgram is not of length k)
22 int freq(const std::string& kgram, char c) const;
23
24 // Random character following given kgram
25 // (throw an exception if kgram is not of length k)
26 // (throw an exception if no such kgram)
27 char kRand(const std::string& kgram);
28 // Generate a string of length L characters by simulating a trajectory
29 // through the corresponding Markov chain. The first k characters of
30 // the newly generated string should be the argument kgram.
31 // Throw an exception if kgram is not of length k.
32 // Assume that L is at least k
33 std::string generate(const std::string& kgram, size_t l);
34
35 private:
36 // Private member variables go here
37 std::unordered_map<std::string, std::unordered_map<char, int>>> map;
38 size_t K;
39 std::string orig_text;
40 std::mt19937 gen;
41 };
42
43 #endif // RANDWRITER_H

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include<vector>
3 #include<string>
4 #include "/autograder/playground/RandWriter.hpp"
5 RandWriter::RandWriter(const std::string& str, size_t k) : gen(std::
    random_device {}()) {
6     size_t len = str.length();
7     orig_text = str;
8     K = k;
9     if (len < k) throw std::invalid_argument("string length is smaller than
    k");
10    for (size_t i = 0; i < len; i++) {
11        std::string kgram;
12
13        for (size_t j = 0; j < k; j++) {
14            kgram += str[(i+j) % len];
15        }
16        char next_char = str[(i+k) % len];
17        map[kgram][next_char]++;
18    }
19 }
20 size_t RandWriter::orderK() const {
21     return K;
22 }
23 int RandWriter::freq(const std::string& kgram) const {
24     int freq = 0;
25     if (kgram.length() != K)
26         throw std::invalid_argument("kgram is of the wrong length");
27     auto kgramIt = map.find(kgram);
28     if (kgramIt == map.end()) return 0;
29     for (const auto& pair : kgramIt->second) {

```

```

30     freq += pair.second;
31 }
32 return freq;
33 }
34 int RandWriter::freq(const std::string& kgram, char c) const {
35     if (kgram.length() != K)
36         throw std::invalid_argument("kgram is of the wrong length");
37     if (K == 0) {
38         int freq = 0;
39         for (size_t i = 0; i < orig_text.length(); i++) {
40             if (orig_text[i] == c) {
41                 freq++;
42             }
43         }
44         return freq;
45     }
46     auto kgramIt = map.find(kgram);
47     if (kgramIt == map.end()) return 0;
48     auto charIt = kgramIt->second.find(c);
49     if (charIt == kgramIt->second.end()) return 0;
50     return charIt->second;
51 }
52 char RandWriter::kRand(const std::string& kgram) {
53     std::vector<char> characters;
54     std::vector<int> weights;
55     if (kgram.length() != K)
56         throw std::invalid_argument("kgram is of the wrong length");
57     auto kgramIt = map.find(kgram);
58     if (kgramIt == map.end())
59         throw std::invalid_argument("kgram was not found in the text");
60     for (const auto& pair : kgramIt->second) {
61         characters.push_back(pair.first);
62         weights.push_back(pair.second);
63     }
64     std::discrete_distribution<> dist(weights.begin(), weights.end());
65     int random_num = dist(gen);
66     return characters[random_num];
67 }
68 std::string RandWriter::generate(const std::string& kgram, size_t l) {
69     std::string markovString = kgram;
70     if (kgram.length() != K)
71         throw std::invalid_argument("kgram is of the wrong length");
72     for (size_t i = K ; i < l; i++) {
73         markovString.push_back
74             (kRand(markovString.substr(markovString.length() - K)));
75     }
76     return markovString;
77 }

```

```

1  // Copyright 2025 Brendan Kelleher
2  #define BOOST_TEST_MODULE NBodyTest
3  #include <sstream>
4  #include<string>
5  #include <boost/test/included/unit_test.hpp>
6  #include "./RandWriter.hpp"
7
8
9
10 BOOST_AUTO_TEST_SUITE(RandWriterTestSuite)
11

```

```

12 BOOST_AUTO_TEST_CASE(Constructor_ValidInput_BuildsModelCorrectly) {
13     RandWriter rw("abcabc", 2);
14
15     BOOST_CHECK_EQUAL(rw.orderK(), 2);
16     BOOST_CHECK_EQUAL(rw.freq("ab", 2);
17     BOOST_CHECK_EQUAL(rw.freq("ab", 'c'), 2);
18     BOOST_CHECK_EQUAL(rw.freq("bc", 'a'), 2);
19 }
20
21 BOOST_AUTO_TEST_CASE(Constructor_KTooLarge_Throws) {
22     BOOST_CHECK_THROW(RandWriter("abc", 4), std::invalid_argument);
23 }
24
25 BOOST_AUTO_TEST_CASE(Freq_KgramNotFound_ReturnsZero) {
26     RandWriter rw("aaaaa", 2);
27     BOOST_CHECK_EQUAL(rw.freq("bb"), 0);
28     BOOST_CHECK_EQUAL(rw.freq("aa", 'b'), 0);
29 }
30
31 BOOST_AUTO_TEST_CASE(Freq_KgramWrongLength_Throws) {
32     RandWriter rw("aaaaa", 2);
33     BOOST_CHECK_THROW(rw.freq("a"), std::invalid_argument);
34     BOOST_CHECK_THROW(rw.freq("aaa", 'a'), std::invalid_argument);
35 }
36
37 BOOST_AUTO_TEST_CASE(Freq_KIsZero_CountsCorrectly) {
38     RandWriter rw("aabbab", 0);
39     BOOST_CHECK_EQUAL(rw.freq("", 'a'), 3);
40     BOOST_CHECK_EQUAL(rw.freq("", 'b'), 3);
41     BOOST_CHECK_EQUAL(rw.freq("", 'c'), 0);
42 }
43
44 BOOST_AUTO_TEST_CASE(KRand_ValidInput_ReturnsExpectedChar) {
45     RandWriter rw("abcabc", 2);
46
47     std::string kgram = "ab";
48     char result = rw.kRand(kgram);
49     BOOST_CHECK(result == 'c');
50 }
51
52 BOOST_AUTO_TEST_CASE(KRand_InvalidKgram_Throws) {
53     RandWriter rw("abcabc", 2);
54     BOOST_CHECK_THROW(rw.kRand("zz"), std::invalid_argument);
55 }
56
57 BOOST_AUTO_TEST_CASE(Generate_ValidInput_GeneratesCorrectLength) {
58     RandWriter rw("abcabc", 2);
59     std::string gen = rw.generate("ab", 10);
60     BOOST_CHECK_EQUAL(gen.length(), 10);
61     BOOST_CHECK_EQUAL(gen.substr(0, 2), "ab");
62 }
63
64 BOOST_AUTO_TEST_CASE(Generate_InvalidKgramLength_Throws) {
65     RandWriter rw("abcabc", 2);
66     BOOST_CHECK_THROW(rw.generate("a", 10), std::invalid_argument);
67 }
68
69 BOOST_AUTO_TEST_SUITE_END()

```

# 7.6 Output

```
• b22keller@Steam:~/ps6$ ./TextWriter 4 200 < romeo.txt
ACT II
PROLOGUE

    But part chamber, the favoury to Friar Laurencher, by my mask?

ROMEO

    Exit

FRIAR LAURENCE

    I beseech the to us!

JULIET

    And his no married with a ghostly coach.
```

## 8 PS7: RandWriter

### 8.1 Discussion

The goal of this project was to create a program that creates a text report describing each time that a Kronos InTouch device restarts. The way to recognize when a boot is starting is when the logging message "server started" appears. I was able to find this message using a regex looking for the exact string by using regex search. The way to recognize when a boot is completed is finding the string "oejs.AbstractConnector:Started SelectChannelConnector" in the text. If a boot has already started and another boot message is found the first boot fails and the second boot starts and looks for either the next start string or the string that signals completion. The times and dates must also be extracted, which can be done using a regex search that searches for dates in the format they appear in the log files. Once the end of the log file is reached information about the boots is output to a report file.

### 8.2 Key Algorithms Used

In order to extraxt all necessary information from the log files I had to utilize mutliple regexes. The two searching for the start boot string and the completion string are very simple as they are just searching for one string that does not change. The regex to find the date and time is more complicated though. The regex for the date and time searches for a sequence of characters that is four digits then a dash, 2 digits then a dash, 2 digits then a dash, then a space, then 2 digits followed by a colon, another two digits followed by a colon, and lastly two more digits. This is the exact format that dates appear in the log files and using this regex pattern along with regex search allows all the necessary dates be extracted.

### 8.3 What I learned

This project was my first time ever using regexes. As they are a completely new tool for me to use I found them interesting to learn about. Learning about regexes allowed me to be able to parse a large input file for specific strings or sequences of characters in an efficient manner. Without the use of regexes it would be much more difficult to produce this program.

### 8.4 Challenges

The most challenging part of this prject for me was learning how to use regex. Coming in to the project I had no experience with them so it took some time to get used to how to use them. I also had some trouble finding the amount of time a boot took, but that was due to not knowing of the microsec clock in the posix time namespace and still attempting to print the time the boot took in milliseconds.

### 8.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++20 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS =
6 # Your compiled .o files
7 OBJECTS = main.o
8 # The name of your program
9 PROGRAM = ps7
10
11
12 .PHONY: all clean lint ps7
13
14
15 all: $(PROGRAM)
16
```



```

17 # Wildcard recipe to make .o files from corresponding .cpp file
18 %.o: %.cpp $(DEPS)
19     $(CC) $(CFLAGS) -c $<
20
21 $(PROGRAM): main.o $(OBJECTS)
22     $(CC) $(CFLAGS) -o $@ $^
23 clean:
24     rm *.o $(PROGRAM)
25
26 lint:
27     cpplint *.cpp *.hpp

```

```

1 // Copyright 2025 Brendan Kelleher
2 #include <iostream>
3 #include <string>
4 #include <fstream>
5 #include <regex>
6 #include "boost/date_time/posix_time/posix_time.hpp"
7 int main(int argc, char* argv[]) {
8     boost::posix_time::ptime endTime;
9     boost::posix_time::ptime startTime =
10     boost::posix_time::microsec_clock::local_time();
11     bool isStarted = false;
12     std::string curLine;
13     std::string filename = argv[1];
14
15     std::ifstream file("logs/" + filename);
16     if (!file) {
17         std::cerr << "File could not be opened.\n";
18         return 1;
19     }
20     std::string backup;
21     std::string output = "";
22     std::string datetime = "";
23     std::string line;
24     std::smatch match;
25     bool succeeded = false;
26     int lineNum = 1;
27     std::ofstream outputFile("trials/" + filename + ".rpt");
28     if (!outputFile) {
29         std::cerr << "Failed to create output file.\n";
30         return 1;
31     }
32     std::regex startPattern("server started");
33     std::regex endPattern
34     ("oejs.AbstractConnector:Started SelectChannelConnector");
35     std::regex timePattern
36     (R"((\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}))");
37     while (getline(file, curLine)) {
38         if (regex_search(curLine, startPattern)) {
39             startTime = boost::posix_time::microsec_clock::local_time();
40             if (std::regex_search(curLine, match, timePattern)) {
41                 datetime = match.str(1);
42             }
43             if (isStarted) {
44                 output = "**** Incomplete boot ****\n\n=== Device boot ===\n\n"
45                 + std::to_string(lineNum) + "(" + filename + ")"
46                 + ": " + datetime + " Boot Start\n";
47             } else {

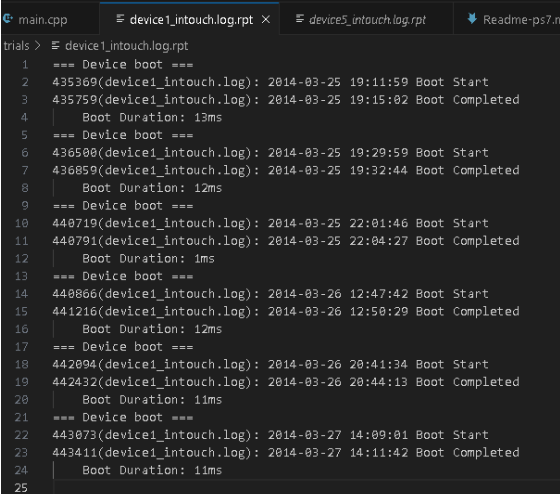
```

```

48         output = "=== Device boot ===\n" + std::to_string(lineNum)
49         + "(" + filename + ")" + ": " +
50         datetime + " Boot Start\n";
51         isStarted = true;
52     }
53     if (!succeeded) {
54         backup = "=== Device boot ===\n" + std::to_string(lineNum)
55         + "(" + filename + ")" + ": " +
56         datetime + " Boot Start\n";
57     }
58 }
59 if (regex_search(curLine, endPattern)) {
60     endTime = boost::posix_time::microsec_clock::local_time();
61     if (std::regex_search(curLine, match, timePattern)) {
62         datetime = match.str(1);
63     }
64     if (isStarted) {
65         boost::posix_time::time_duration diff = endTime - startTime;
66         if (!succeeded) {
67             output = backup;
68         }
69         output += std::to_string(lineNum) + "(" + filename + ")"
70         + ": " + datetime +
71         " Boot Completed\n\tBoot Duration: "
72         + std::to_string(diff.total_milliseconds()) + "ms\n";
73         isStarted = false;
74         succeeded = true;
75     } else {
76         output =
77         "Warning: Boot completed without a detected boot start.\n";
78     }
79 }
80 if (output.length() > 0 && succeeded) {
81     outputFile << output;
82 }
83 output = "";
84 lineNum++;
85 }
86 }

```

## 8.6 Output



```

main.cpp  device1_intouch.log.rpt  device5_intouch.log.rpt  Readme-ps7.m
1  === Device boot ===
2  435369(device1_intouch.log): 2014-03-25 19:11:59 Boot Start
3  435759(device1_intouch.log): 2014-03-25 19:15:02 Boot Completed
4  | Boot Duration: 13ms
5  === Device boot ===
6  436500(device1_intouch.log): 2014-03-25 19:29:59 Boot Start
7  436859(device1_intouch.log): 2014-03-25 19:32:44 Boot Completed
8  | Boot Duration: 12ms
9  === Device boot ===
10 440719(device1_intouch.log): 2014-03-25 22:01:46 Boot Start
11 440791(device1_intouch.log): 2014-03-25 22:04:27 Boot Completed
12 | Boot Duration: 1ms
13 === Device boot ===
14 440866(device1_intouch.log): 2014-03-26 12:47:42 Boot Start
15 441216(device1_intouch.log): 2014-03-26 12:50:29 Boot Completed
16 | Boot Duration: 12ms
17 === Device boot ===
18 442094(device1_intouch.log): 2014-03-26 20:41:34 Boot Start
19 442432(device1_intouch.log): 2014-03-26 20:44:13 Boot Completed
20 | Boot Duration: 11ms
21 === Device boot ===
22 443073(device1_intouch.log): 2014-03-27 14:09:01 Boot Start
23 443411(device1_intouch.log): 2014-03-27 14:11:42 Boot Completed
24 | Boot Duration: 11ms
25

```