

Computing IV Portfolio (section 202)

Shriya Thakur

April 29, 2022

Contents

1	PS0: "Hello World" with SFML	3
2	PS1: Linear Feedback Shift Register(PS1a) and Photomagic(PS1b):	7
3	PS2: NBody Simulation	15
4	PS3: Recursive Sierpinski	26
5	PS4: Circular Buffer and Guitar String simulation	32
6	PS5: DNA Sequence Alignment	41
7	PS6: Frequency Analysis	47
8	PS7:Kronos Time Parsing	55
	Time to complete: A week	

1 PS0: "Hello World" with SFML

1.1 Discussion

This assignment was a brief introduction to SFML and its libraries and to get our tools set up for the rest of the semester.

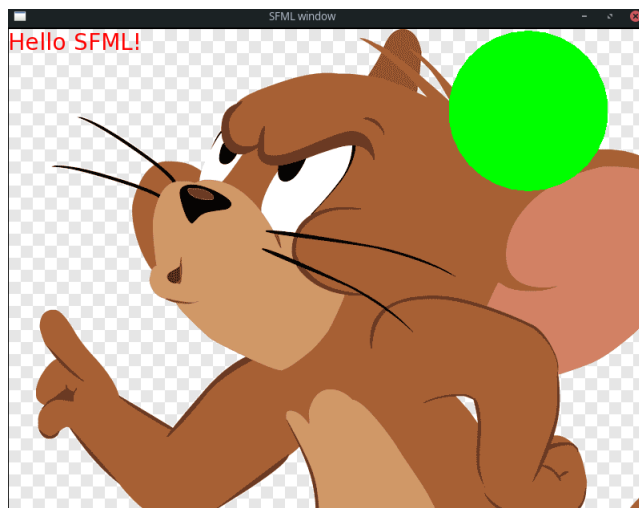


Figure 1: Hello SFML!

1.2 Discussion of the assignment itself and what you accomplished:

This assignment was an introductory assignment to get or build environment set up for the rest of the semester. However, I already use Manjaro Linux and thus didn't have to worry about setting up Virtual Box or the Windows Linux Subsystem so I started following the steps directly to get SFML set up and running. The first sub part of of this assignment was to get a Green Circle printed in an SFML window using using SFML tutorial available at-[urlhttps://www.sfml-dev.org/tutorials/2.5/start-linux.php](https://www.sfml-dev.org/tutorials/2.5/start-linux.php) The second subpart was to get a sprite printed on the screen and the green circle printed on the top right corner of the screen.

1.3 What I already knew:

I had an introduction to coding in C++ before I started so I didn't have to worry too much about language syntax and semantics. I did have to look into the SFML library and what it had to offer.

1.4 What I learned:

This assignment gave me a brief introduction to SFML and graphics in general via a very simple assignment aka to initially print a Green circle on screen(testing if we have all the tools we need set-up) and then getting a sprite of our choice to display in a window with the Green Circle at the top. Lastly we had to get the sprite to move and do things on certain

keystrokes which was left upto us to decide. I learned how to print a sprite and make it move using some basic commands.

1.5 Challenges:

I had some issues in installing SFML libraries in the directory that I wanted so I had to find where it got installed and specify the path where I wanted it to be installed. I also had an issue with changing the True Type fonts or finding them since I didn't have them pre-installed in any directory. I did also have some issues in getting my sprite to respond to keystrokes(WASD) until I was told that we had to capitalize them in code. Lastly, I had trouble getting the Green Circle and my sprite image(Jerry) to appear in the same window.

1.6 A discussion of one or more key algorithms, data structures, or Object Oriented Designs that were central to the assignment:

We didn't really use any specific algorithms, or data structures for this assignment.

1.7 List any comments here:

X- Sprite rotates Clockwise Y- Sprite rotates Anti-Clockwise W- Sprite moves upward A- Sprite moves leftward S- Sprite moves downward D- Sprite moves rightward Z- Surprise Element! (Sprite changes from Jerry to Tom)

1.8 Does it work? If not, what problems are present?

Yes it works as expected!

1.9 Codebase:

1.9.1 Makefile

```
1 CC = g++
2 CFLAGS = -Wall -Werror -pedantic --std=c++17
3 LIBS = -lboost_unit_test_framework -lsfml-graphics -lsfml-window -lsfml-
    system
4
5 all:sfml-app
6 sfml-app: main.o
7 $(CC) $(CFLAGS) -o sfml-app main.o $(LIBS)
8 main.o: main.cpp
9 $(CC) $(CFLAGS) -c main.cpp $(LIBS)
10 lint:
11 cpplint main.cpp
12 clean:
13 rm -f main.o
14 distclean: clean
15 rm -f sfml-app
```

1.9.2 main.cpp

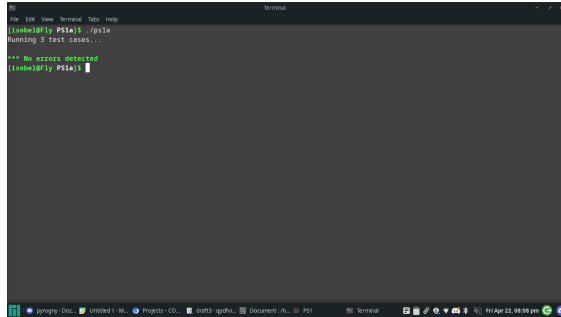
```
1  /*****
2  Copyright Shriya 2022.
3  *Name: Shriya Thakur.
4  *Course name: COMP.2040 vc 1 202.
5  *Assignment: PS0- setting up and introduction to SFML.
6  *Instructor's name: Dr. James Daly.
7  *Date: Jan 23rd, 2021.
8  *Sources Of Help: Office hours, Class Discord Server,
9  *Classmate- Kevin Sprague.
10 *****/
11 #include <iostream>
12 #include <SFML/Audio.hpp>
13 #include <SFML/Graphics.hpp>
14 int main() {
15     // Create the main window
16     sf::RenderWindow window(sf::VideoMode(800, 600), "SFML window");
17     // Load a sprite to display
18     sf::Texture texture;
19     if (!texture.loadFromFile("sprite.png")) {
20         return EXIT_FAILURE;
21     }
22     sf::Sprite sprite(texture);
23     // Create a graphical text to display
24     sf::Font font;
25     if (!font.loadFromFile("Vera.ttf")) {
26         return EXIT_FAILURE;
27     }
28     sf::Text text("Hello SFML!", font, 28);
29     text.setFillColor(sf::Color::Red);
30     sf::CircleShape shape(100.f);
31     shape.setPosition(550.f, 4.f);
32     shape.setFillColor(sf::Color::Green);
33     // Set Sprite position
34     sprite.setPosition(1.f, 2.f);
35     // Start the game loop
36     while (window.isOpen()) {
37         // Process events
38         sf::Event event;
39         while (window.pollEvent(event)) {
40             // Close window: exit
41             if (event.type == sf::Event::Closed) {
42                 window.close();
43             }
44         }
45         // 2.MAKE THE IMAGE SPRITE MOVE
46         if (sf::Keyboard::isKeyPressed(sf::Keyboard::Key::X)) {
47             sprite.rotate(0.1f);
48         }
49         if (sf::Keyboard::isKeyPressed(sf::Keyboard::Key::Y)) {
50             sprite.rotate(-0.1f);
51         }
52         // 3.MAKE THE IMAGE SPRITE RESPOND ON KEYSTROKES
```

```

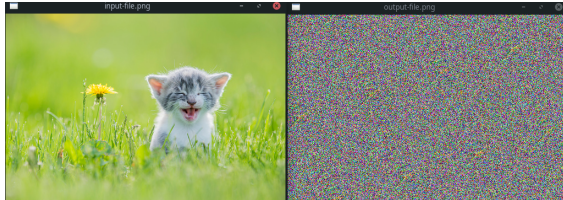
53     // Left
54     if (sf::Keyboard::isKeyPressed(sf::Keyboard::Key::A)) {
55         sprite.move(sf::Vector2f(-0.8f, 0.0));
56     }
57     // Up
58     if (sf::Keyboard::isKeyPressed(sf::Keyboard::Key::W)) {
59         sprite.move(sf::Vector2f(0.0, -0.8f));
60     }
61     // Right
62     if (sf::Keyboard::isKeyPressed(sf::Keyboard::Key::D)) {
63         sprite.move(sf::Vector2f(0.8f, 0.0));
64     }
65     // Down
66     if (sf::Keyboard::isKeyPressed(sf::Keyboard::Key::S)) {
67         sprite.move(sf::Vector2f(0.0, 0.8f));
68     }
69     // 4.MAKE IT DO SOMETHING ELSE
70     if (sf::Keyboard::isKeyPressed(sf::Keyboard::Key::Z)) {
71         // Make Tom appear by changing sprite
72         if (!texture.loadFromFile("tom.png")) {
73             return EXIT_FAILURE;
74         }
75         sf::Sprite tom(texture);
76         tom.setPosition(400.f, 4.f);
77         window.draw(tom);
78     }
79     // Clear screen
80     window.clear();
81     // Draw the sprite
82     window.draw(sprite);
83     // Draw the shape
84     window.draw(shape);
85     // Draw the string
86     window.draw(text);
87     // Update the window
88     window.display();
89 }
90 return EXIT_SUCCESS;
91 }

```

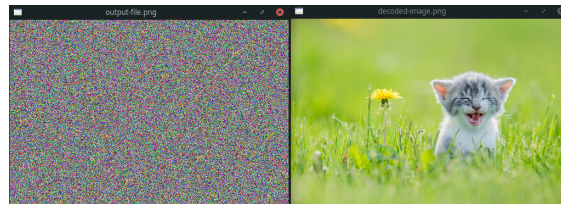
2 PS1: Linear Feedback Shift Register(PS1a) and Photomagic(PS1b):



(a) fig 2: Testing API with Boost tests



(b) fig 3: Image encryption



(c) fig 4: Image decryption

Figure 2: Using a Linear Feedback Shift Register to Encode and Decode an image

2.1 Discuss the assignment itself:

Part a of this assignment had us make a Linear feedback shift register which in simplified terms is a pseudo-random number generator. It takes a linear function of a previous state as input, most commonly, this function is a Boolean exclusive or(XOR). It shifts each bit one position to the left and replaces the vacated bit by the value found by xoring it with the bits previously at the tap positions and pushes that into the position of the least significant bit. Part b uses this and encodes an image and then decode it thus getting back the image we started with. We read 3 arguments from the command line namely- name of input file, name of output file, and a seed. Then use SFML to load and display the images.

2.2 What I already knew:

Besides CPP, and some basic SFML, I already knew how I was going to approach this using arrays to store the bits and that we would use the exclusive or operator.

2.3 What I learned:

Part 1a of this assignment had us code a LFSR API in CPP and test each function using Boost macros. This was my first interaction with using the boost libraries so I learned about the different macros there are like `BOOST_REQUIRE_THROW`, `BOOST_REQUIRE_NO_THROW`,

BOOST_REQUIRE_EQUALS,BOOST_CHECK_EQUALS to test different functions depending on their expected behavior and see if it matched. We had to write two additional boost tests Part 1b had us use that API to encode and decode our image. I also learned how to print two SFML windows beside each other.

2.4 Challenges:

2.5 PS1a:

I was having issues:

1. Coverting characters from the seed into integer bits in the constructor.
2. Deciding which bit indexes to xor together(xor is commutative so order doesnt matter) since my array is indexed left to right whereas the bits are usually indexed right to left.
3. Thinking of what Boost tests to run to make sure my API works as intended.
4. Calculating size of my array.

2.6 PS1b:

The only issue I ran into that took up most of my time in this assignment was getting the image to encrypt. After some debugging I realised there was a bug in my step() function from ps1a. I was changing and returning the least significant bit(lsb) correct but forgot to push it back in my array so the xor in transform() was not working since the least significant bit was not what we were expecting in the array originally.

2.7 Does it work?

yes! the API works since it was tested using Boost macros and also encrypts the image.

2.8 Key algorithms or data structures used:

I mainly used an array to store the bits supplied by the string and and the exclusive or(xor) operator to generate pseudo random numbers.

2.9 List any comments here

Running the executable PhotoMagic builds depicts the encryption and decryption process and running tester depicts that all boost tests passed successfully.

2.10 Codebase

2.10.1 Makefile:

```
1 CC = g++
2 CFLAGS = -Wall -Werror -pedantic --std=c++17
3 LIBS = -lboost_unit_test_framework -lsfml-graphics -lsfml-window -lsfml-
      system
4
5 all:PhotoMagic tester
6 tester: test.o FibLFSR.o
7 $(CC) $(CFLAGS) -o tester test.o FibLFSR.o $(LIBS)
8 PhotoMagic: PhotoMagic.o FibLFSR.o
9 $(CC) $(CFLAGS) -o PhotoMagic PhotoMagic.o FibLFSR.o $(LIBS)
10 PhotoMagic.o: PhotoMagic.cpp
11 $(CC) $(CFLAGS) -c PhotoMagic.cpp $(LIBS)
12 FibLFSR.o: FibLFSR.cpp FibLFSR.h
13 $(CC) $(CFLAGS) -c FibLFSR.cpp $(LIBS)
14 test.o: test.cpp
15 $(CC) $(CFLAGS) -c test.cpp $(LIBS)
16 lint:
17 cpplint test.cpp
18 cpplint --filter=-runtime/references PhotoMagic.cpp
19 cpplint FibLFSR.h
20 cpplint FibLFSR.cpp
21 clean:
22 rm -f FibLFSR.o PhotoMagic.o test.o
23 distclean: clean
24 rm -f PhotoMagic
```

2.10.2 Photomagic:

```
1 /*****
2 Copyright Shriya 2022
3 Name: Shriya Thakur
4 Date: Feb 07th, 2021
5 Course: Comp IV
6 Instructor: Dr Daly
7 Assignment: generating a pseudo-random number
8 Time taken: ~10 of work over 2 days
9 *****/
10 #include "FibLFSR.h"
11 #include <iostream>
12 #include <string>
13 #include <SFML/System.hpp>
14 #include <SFML/Window.hpp>
15 #include <SFML/Graphics.hpp>
16
17 using std::string;
18 using std::cin;
19 using std::cout;
20 using std::endl;
21
```



```

22 void transform(sf::Image& i, FibLFSR* obj) {
23     sf::Color p;
24     sf::Vector2u size = i.getSize();
25     for (unsigned int x = 0; x < size.x; x++) {
26         for (unsigned int y = 0; y < size.y; y++) {
27             p = i.getPixel(x, y);
28             p.r = obj->generate(8) ^ p.r;
29             p.g = obj->generate(8) ^ p.g;
30             p.b = obj->generate(8) ^ p.b;
31             i.setPixel(x, y, p);
32         }
33     }
34 }
35
36 int main(int argc, char** argv) {
37     string inputFileName = argv[1];
38     string outputFileName = argv[2];
39     string initSeed = argv[3];
40     FibLFSR obj(initSeed);
41     // create image
42     sf::Image image;
43     // Load image
44     if (!image.loadFromFile(inputFileName)) {
45         return -1;
46     }
47     // load to texture
48     sf::Texture texture;
49     texture.loadFromImage(image);
50     // load to sprite
51     sf::Sprite sprite;
52     sprite.setTexture(texture);
53     // call transform
54     transform(image, &obj);
55     sf::Texture texture_;
56     texture_.loadFromImage(image);
57     sf::Sprite sprite_;
58     sprite_.setTexture(texture_);
59     sf::Vector2u size = image.getSize();
60     sf::RenderWindow window1(sf::VideoMode(size.x, size.y), inputFileName);
61     sf::RenderWindow window2(sf::VideoMode(size.x, size.y), outputFileName);
62     while (window1.isOpen() && window2.isOpen()) {
63         sf::Event event;
64         while (window1.pollEvent(event)) {
65             if (event.type == sf::Event::Closed) {
66                 window1.close();
67             }
68         }
69         while (window2.pollEvent(event)) {
70             if (event.type == sf::Event::Closed) {
71                 window2.close();
72             }
73         }
74         window1.clear();
75         window1.draw(sprite);

```

```

76     window1.display();
77     window2.clear();
78     window2.draw(sprite_);
79     window2.display();
80 }
81 if (!image.saveToFile(outputFileName)) {
82     return -1;
83 }
84 return 0;
85 }

```

2.10.3 FibLFSR.h:

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Date: Jan 30th, 2021
5  Course: Comp IV
6  Instructor: Dr Daly
7  Assignment: PS1a- generating a pseudo-random number
8  Time take: ~24hours of work over a week
9  *****/
10 #pragma once
11 #include<string>
12 #include<iostream>
13
14 using std::string;
15 using std::ostream;
16
17 class FibLFSR {
18 public:
19     explicit FibLFSR(string seed);
20     int step();
21     int generate(int k);
22     int Size();
23     ~FibLFSR();
24     friend ostream& operator<<(ostream& output, FibLFSR& num);
25 private:
26     // To store size of object
27     int size_;
28     // array of bits
29     int* register_;
30 };

```

2.10.4 FibLFSR.cpp:

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Date: Feb 07th, 2021
5  Course: Comp IV
6  Instructor: Dr Daly
7  Assignment: generating a pseudo-random number
8  Time taken: ~24hr of work over a week

```

```

9  ****
   */
10 #include "FibLFSR.h"
11 #include <iostream>
12 #include <string>
13
14 using std::string;
15 using std::cout;
16 using std::endl;
17 using std::ostream;
18 using std::out_of_range;
19
20 // Constructor to create LFSR with the given initial seed
21 FibLFSR :: FibLFSR(string seed) {
22     size_ = seed.length();
23     // allocate memory to the array equal to length of string
24     register_ = new int[size_];
25     // Loop over string to put into a vector
26     for (int i = 0; i < size_; i++) {
27         // Populate array
28         register_[i] = seed[i] - '0';
29     }
30 }
31
32 // Simulate one step and return the new bit as 0 or 1
33 int FibLFSR :: step() {
34     // Calculate bit indexes
35     int msb = register_[0];
36     // 15 - 10 = 5
37     int bit10 = register_[Size()-1-10];
38     // 15 - 12 = 3
39     int bit12 = register_[Size()-1-12];
40     // 15 - 13 = 2
41     int bit13 = register_[Size()-1-13];
42     // int lsb = register_[15];
43     int lsb = register_[Size()-1];
44     // Left shift bits by 1;
45     for (int i = 0; i < 15 ; i++) {
46         register_[i] = register_[i+1];
47     }
48     // Lsb is the xor of the original msb with bits 10, 12, 13
49     lsb = bit13 ^ bit12 ^ bit10 ^ msb;
50     register_[Size()-1] = lsb;
51     return lsb;
52 }
53
54 // Simulate k steps and return k-bit integer
55 int FibLFSR :: generate(int k) {
56     if (k < 0) {
57         throw out_of_range("Negative numbers not allowed");
58     }
59     int temp = 0;
60     // Call step function k times
61     for (int i = 0; i < k; i++) {

```

```

62     temp *= 2;
63     temp += this->step();
64 }
65 return temp;
66 }
67
68 // Calculate size of LFSR object
69 int FibLFSR :: Size() {
70     return size_;
71 }
72
73 // Output stream overload
74 ostream& operator<<(ostream& output, FibLFSR& num) {
75     for (int i = 0; i < num.Size(); i++) {
76         output << num.register_[i];
77     }
78     cout << endl;
79     return output;
80 }
81
82 // Destructor to prevent memory leaks
83 FibLFSR :: ~FibLFSR() {
84     delete[] register_;
85 }

```

2.10.5 test.cpp:

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Sources of help: Dr Daly, class discord Classmate- Kevin Sprague
5  Date: Jan 30th, 2022
6  *****/
7  #include "FibLFSR.h"
8  #include <iostream>
9  #include <string>
10
11 using std::cout;
12 using std::endl;
13 using std::invalid_argument;
14 using std::out_of_range;
15
16 #define BOOST_TEST_DYN_LINK
17 #define BOOST_TEST_MODULE Main
18 #include <boost/test/unit_test.hpp>
19
20 BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
21     FibLFSR l("1011011000110110");
22     BOOST_REQUIRE(l.step() == 0);
23     BOOST_REQUIRE(l.step() == 0);
24     BOOST_REQUIRE(l.step() == 0);
25     BOOST_REQUIRE(l.step() == 1);
26     BOOST_REQUIRE(l.step() == 1);
27     BOOST_REQUIRE(l.step() == 0);

```

```

28     BOOST_REQUIRE(l.step() == 0);
29     BOOST_REQUIRE(l.step() == 1);
30     FibLFSR l2("1011011000110110");
31     BOOST_CHECK_EQUAL(l2.generate(9), 51);
32 }
33
34 // First implemented test
35 BOOST_AUTO_TEST_CASE(testCase2) {
36     FibLFSR m("1010001111000000");
37     BOOST_REQUIRE(m.step() == 0);
38     BOOST_CHECK_EQUAL(m.step(), 1);
39     BOOST_CHECK_EQUAL(m.step(), 0);
40     BOOST_CHECK_EQUAL(m.step(), 0);
41     BOOST_CHECK_EQUAL(m.step(), 1);
42     BOOST_CHECK_EQUAL(m.step(), 0);
43     BOOST_CHECK_EQUAL(m.step(), 1);
44     BOOST_REQUIRE_EQUAL(m.step(), 0);
45     FibLFSR m2("1010101010101010");
46     m2.generate(10);
47     BOOST_CHECK_EQUAL(m2.generate(10), 288);
48 }
49
50 // Second implemented test- check if size of object is as mentioned and if
51 // generate function is called with good input
52 BOOST_AUTO_TEST_CASE(testCase3) {
53     FibLFSR n("0100001000001111");
54     BOOST_REQUIRE_EQUAL(n.Size(), 16);
55     BOOST_REQUIRE_THROW(n.generate(-1), out_of_range);
56 }

```

3 PS2: NBody Simulation

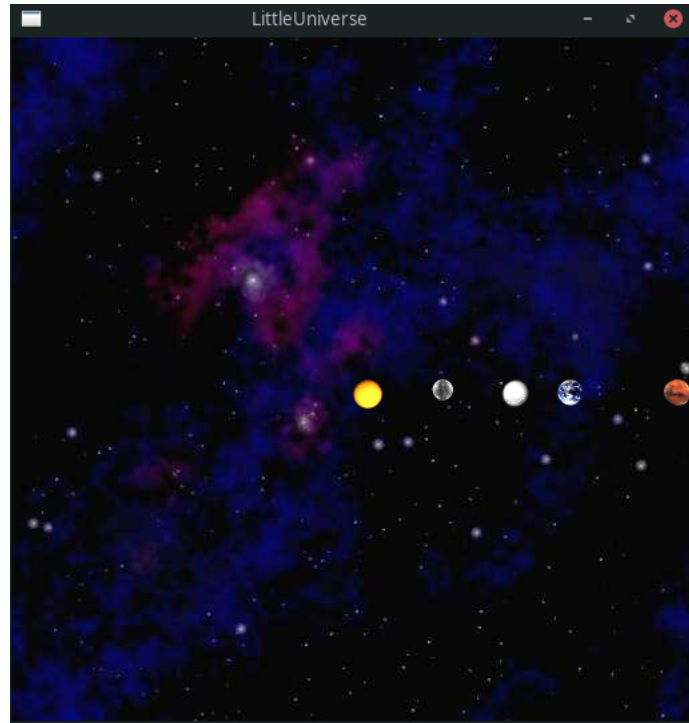


Figure 3: Solar system

3.1 What I already knew

In class, Professor Daly gave us a briefing about scaling the coordinates and also a general recap of Newtonian Physics- particularly using the 3 equations of motion to calculate new velocity and new position of the planets. We also were already familiar with SFML and loading sprites whose names, with other data, are in a text file.

3.2 What I learned

3.3 Discuss the assignment briefly and what Object Oriented Designs and/or data structures use

3.4 For ps2a:

1. Inheritance: I think inheritance was a central theme I used since both my Universe and CelestialBody classes inherit off of Sf::Drawable to get access to the virtual draw function and override it after.
2. Encapsulation: Since we made a class interface and no private variables can be accessed/changed as is unless via a certain function.

3. Data structure- Vector: I used a vector to store shared celestial body pointers in the Universe class.

3.5 For PS2b:

This assignment had us use our User interface from 2a and implement an animation in our mini solar system. the planets now move according to the laws of physics. We used the three equations of motion, and newtons laws to simulate our Solar System animation. I also display time elapsed (in years) in the window and play "Also Sprach Zarathustra" as background music.

3.6 Challenges:

3.7 PS2a:

I had multiple issues in this assignment ranging from what to do, to how to do it, to debugging, and then memory leaks(I still have memory leaks even though I used shared pointers but apparently that is an SFML thing- sf::Sprite, sf::RenderWindow) I had an issue with scaling and setting the sun and other planet positions and had to reference the diagram Dr. Daly made in class multiple times over.

3.8 PS2b:

The physics part was surprisingly easy to get done, I only used Prof Daly's notes to update positions, velocity, acceleration and force. I had to inverse signs for xposn and yvelocity in order to have them travel in expected direction. I was overwriting my private variable for both positions initially in CelestialBody.cpp and therefore the planets were flying out of my screen but that was fixed on using another local variable in the position function. Before that I was having NAN issues due to 0/0 division which was fixed by putting an if condition and continue inside the body if original and new planet positions are same. Lastly, I had been having audio issues. The audio loads and plays as well but on screen I get an error message that says: "ALSOFT] (EE) Failed to set real-time priority for thread: Operation not permitted (1) [ALSOFT] (EE) Failed to set real-time priority for thread: Operation not permitted (1)" After some googling I understood thats because pulseaudio doesnt have real time processing capabilities and I need to use a sound server daemon like JACK. I tried installing it through pacman but most libs were already installed so I am confused why those statements are printed on screen but the audio plays nonetheless! I did some research on this and turns out that I need to make a few changes to my config files so to bypass that I tried to use the command: ulimit -r 200000 but that didnt work. I then tried sudo since it needs admin priviledges but sudo didnt recognise ulimit since sudo searches for ulimit binaries but they dont have one so the error message "command not found" displays This means I need to use another shell, like sh lets say(I currently use bash). however this doesnt happen on the CS server and the audio plays as well so I stopped digging into this.

3.9 Mention any extra comments here:

This was my psxa redo. I had step1, step 1year, and step 3 results wrong initially as mentioned when the grader gave me feedback so I had to update and correct the formulas for calculation yposn and yvelocity in my celestialBody class. Now the values are quite close to what's expected and of course they are going to vary depending on how the implementation is and even though it should exactly match, it wont.I also changed the format for printing out stuff and got rid of the tab. I have also linted all files and have added a lint target to my makefile. I loaded the starfield.jpg file as background for extra credit for ps2a. I have printed the time elapsed in years in the window for extra credit!(PS2b) and had it play music.

3.10 Does your code work succesfully, if not, what are the problems?

Yes, my code works successfully and has no issues as seen in the screenshot before. The numbers for ps2b arent exact but within the acceptable margin of error.

3.11 Describe any Key Algorithms/Data Structures/Object oriented designs

1. Inheritance: I think inheritance was a central theme I used since both my Universe and CelestialBody classes inherit off of Sf::Drawable to get access to the virtual draw function and override it after.
2. Encapsulation: Since we made a class interface and no private variables can be accessed/changed as is unless via a certain function.
3. Data structure- Vector: I used a vector to store shared celestial body pointers in the Universe class.

3.12 Codebase

3.12.1 Makefile

```
1 CC = g++ --std=c++17
2 LIBS = -lsfml-graphics -lsfml-system -lsfml-window -lsfml-audio
3 CFLAGS = -Wall -Werror -pedantic
4
5 all:NBody
6 NBody: Universe.o CelestialBody.o NBody.o
7 $(CC) $(CFLAGS) -o NBody NBody.o Universe.o CelestialBody.o $(LIBS)
8 NBody.o: NBody.cpp
9 $(CC) $(CFLAGS) -c NBody.cpp $(LIBS)
10 Universe.o: Universe.cpp
11 $(CC) $(CFLAGS) -c Universe.cpp $(LIBS)
12 CelestialBody.o: CelestialBody.cpp CelestialBody.h
13 $(CC) $(CFLAGS) -c CelestialBody.cpp $(LIBS)
14 clean:
```



```

15  rm -f Universe.o CelestialBody.o NBody.o
16  distclean: clean
17  rm -f NBody
18  lint:
19  cpplint NBody.cpp
20  cpplint Universe.h
21  cpplint Universe.cpp
22  cpplint --filter=-runtime/references,-build/header_guard --extensions=cpp
    ,h CelestialBody.h
23  cpplint CelestialBody.cpp

```

3.12.2 NBody.cpp

```

1  /*****
2  Copyright Shriya 2022
3  *Name: Shriya Thakur
4  *Date: February 20th, 2021
5  *Sources Of Help: Professor Daly in Office hours, Class Discord,
6  Classmate- Ryan Casey
7  *Assignemnt: PS2b-NBody simulation.
8  *Time taken: ~16 hours of intermitten work for PS2a and 6hrs for 2b
9  *****/
10 #include "Universe.h"
11 #include "CelestialBody.h"
12 #include <iostream>
13 #include <sstream>
14 #include <string>
15 #include <SFML/Audio.hpp>
16
17 using std::stod;
18 using std::endl;
19 using std::cout;
20 using std::ostringstream;
21 using std::string;
22
23 int main(int argc, char** argv) {
24     double startTime = 0;
25     // argv[0] = executable name
26     double endTime = stod(argv[1]);
27     double duration = stod(argv[2]);
28     Universe universe;
29     universe.SetPlanetPosition(512);
30     sf::Texture texture;
31     sf::SoundBuffer buf;
32     if (!buf.loadFromFile("ASZ.ogg")) {
33         cout << "Couldnt load sound file" << endl;
34         return -1;
35     }
36     sf::Sound sound;
37     sound.setBuffer(buf);
38     sound.play();
39     // load texture(use starfiled.jpg) for extra credit
40     texture.loadFromFile("starfield.jpg");
41     sf::RenderWindow window(sf::VideoMode(512, 512), "LittleUniverse");

```

```

42     sf::Sprite sprite;
43     sf::Text text;
44     sf::Font font;
45     if (!font.loadFromFile("Vera.ttf")) {
46         return EXIT_FAILURE;
47     }
48     text.setFillColor(sf::Color::Yellow);
49     text.setPosition(200.0, 450.0);
50     text.setCharacterSize(16);
51     text.setStyle(sf::Text::Italic | sf::Text::Underlined);
52     text.setFont(font);
53     sprite.setTexture(texture);
54     window.setFramerateLimit(60);
55     while (window.isOpen()) {
56         sf::Event event;
57         while (window.pollEvent(event)) {
58             if (event.type == sf::Event::Closed) {
59                 window.close();
60             }
61         }
62         if (startTime < endTime) {
63             universe.step(duration);
64             universe.SetPlanetPosition(512);
65             startTime += duration;
66             double year = (((startTime / 60)/60)/24)/365);
67             string time = "Time elapsed(in years): ";
68             text.setString(time + std::to_string(year));
69             // startTime += duration;
70         }
71         if (startTime >= endTime)
72             break;
73         window.clear();
74         window.draw(sprite);
75         // Universe inherits publicly from draw for this wonderful format
76         window.draw(universe);
77         // Draw text after universe so it is visible LOL
78         window.draw(text);
79         window.display();
80     }
81     cout << universe << endl;
82     return 0;
83 }

```

3.12.3 CelestialBody.h

```

1  /*****
2  *Copyright Shriya 2022
3  *Name: Shriya Thakur
4  *Date: February 13th, 2021
5  *Sources Of Help: Professor daly in Office hours, Class Discord,
6  Classmates- Kevin Sprague, Ryan Casey, Tutor- Ben Friedman
7  *Assignemnt: PS2a- Create UI for NBody simulation.
8  *Time taken: ~3 days of intermitten work
9  *****/

```

```

10 #pragma once
11 #include <cmath>
12 #include <iostream>
13 #include <vector>
14 #include <memory>
15 #include <string>
16 #include <SFML/Graphics.hpp>
17 #include <SFML/Window.hpp>
18 #include <SFML/System.hpp>
19
20 using std::istream;
21 using std::ostream;
22 using std::string;
23 using std::vector;
24 using std::shared_ptr;
25
26 class CelestialBody: public sf::Drawable {
27 public:
28     // Default constructor
29     CelestialBody();
30     // value constructor
31     CelestialBody(double x, double y, double m, string fileName,
32     double xvel, double yvel);
33     // init sprite and texture-helper
34     void Initialize();
35     // Set position of each planet
36     void SetPos(const double& radius, const int& viewport);
37     // Input stream overload
38     friend istream& operator>>(istream& in, CelestialBody& CelBod);
39     // output stream overload
40     friend ostream& operator<<(ostream& out, CelestialBody& CelBod);
41     // destructor
42     ~CelestialBody();
43     // update posn
44     void NewPosition(double stepDuration);
45     // update velocity
46     void NewVelocity(const vector<shared_ptr<CelestialBody>>& planet,
47     double step);
48
49 private:
50     // variables
51     double xpos;
52     double ypos;
53     double mass;
54     string file;
55     double xvelocity;
56     double yvelocity;
57     sf::Texture CelestialTexture;
58     sf::Sprite CelestialSprite;
59     // draw function inherited private since it is private in base class
60     virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
61     const;
62 };

```

3.12.4 CelestialBody.cpp

```
1  /*****
2  Copyright Shriya 2022
3  *Name: Shriya Thakur
4  *Date: February 17th, 2021
5  *Sources Of Help: Professor daly in Office hours, Class Discord,
6  *Classmate - Ryan Casey.
7  *Assignemnt: PS2b- Create UI for NBody simulation.
8  Time taken: ~16 hours
9  *****/
10 #include "CelestialBody.h"
11 #include <cmath>
12 #include <string>
13 #include <iomanip>
14 #include <vector>
15 #include <memory>
16
17 using std::istream;
18 using std::ostream;
19 using std::string;
20 using std::endl;
21 using std::cout;
22 using std::vector;
23 using std::shared_ptr;
24 using std::sqrt;
25 using std::setw;
26 using std::setprecision;
27
28 #define GRAVCONST 6.6e-11
29
30 // Implement default constructor
31 CelestialBody::CelestialBody() {}
32
33 // implement value constructor
34 CelestialBody::CelestialBody(double x, double y,
35 double m, string fileName, double xvel, double yvel) {
36     xpos = x;
37     ypos = y;
38     mass = m;
39     file = fileName;
40     xvelocity = xvel;
41     yvelocity = yvel;
42     // set texture and sprite as available in the file
43     CelestialTexture.loadFromFile(fileName);
44     CelestialSprite.setTexture(CelestialTexture);
45     CelestialSprite.setPosition(x, y);
46 }
47
48 void CelestialBody::draw(sf::RenderTarget& target,
49 sf::RenderStates states) const {
50     target.draw(CelestialSprite, states);
51 }
52
```

```

53 void CelestialBody::SetPos(const double& radius, const int& viewport) {
54     // use picture from class as reference and thankgod for tutoring
55     double windowXPosition = (viewport / 2) + ((xpos / radius) * (viewport /
56         2));
57     double windowYPosition = (viewport / 2) + ((ypos / radius) * (viewport /
58         2));
59     // set sprite at new position
60     CelestialSprite.setPosition(windowXPosition, windowYPosition);
61 }
62 // Helper function to make my life easier in overloading input stream-
63 // sets
64 // sprite and texture from file
65 void CelestialBody:: Initialize() {
66     CelestialTexture.loadFromFile(file);
67     CelestialSprite.setTexture(CelestialTexture);
68 }
69 istream& operator>>(istream& input, CelestialBody& CelBod) {
70     input
71     >> CelBod.xpos >> CelBod.ypos
72     >> CelBod.xvelocity >> CelBod.yvelocity
73     >> CelBod.mass >> CelBod.file;
74     // make call to init func
75     CelBod.Initialize();
76     return input;
77 }
78 ostream& operator<<(ostream& out, CelestialBody& CelBod) {
79     out << setw(15) << std::left << setprecision(6) << CelBod.xpos << setw
80         (15)
81         << std::left << setprecision(6) << CelBod.ypos << setw(15)
82         << std::left << setprecision(6) << CelBod.xvelocity << setw(15)
83         << std::left << setprecision(6) << CelBod.yvelocity << setw(15)
84         << std::left << setprecision(6) << CelBod.mass << setw(15)
85         << std::left << CelBod.file;
86     return out;
87 }
88 CelestialBody::~CelestialBody() {
89     // cout << "Celestial Body destroyed" << endl;
90 }
91 // update position of each celestial body
92 void CelestialBody::NewPosition(double stepDuration) {
93     // xt = x0 + v.t
94     xpos += stepDuration * xvelocity;
95     // yt = y0 + v.t
96     ypos += stepDuration * yvelocity;
97 }
98 // update velocity
99 void CelestialBody::NewVelocity
100 (const vector<shared_ptr<CelestialBody>>& planet , double step) {

```

```

103     double Fx = 0;
104     double Fy = 0;
105     for (auto i : planet) {
106         if (xpos == i->xpos && ypos == i->ypos)
107             continue;
108         // difference in positions
109         double deltaX = i->xpos - xpos;
110         double deltaY = i->ypos - ypos;
111         // distance between objects
112         double objectDistance = sqrt(pow(deltaX, 2) + pow(deltaY, 2));
113         // F = Gm1m2/r^2
114         double F = ((GRAVCONST * i->mass * this->mass) / pow(objectDistance,
115             2));
116         // calc components of net force
117         Fx = F * (deltaX / objectDistance);
118         Fy = F * (deltaY / objectDistance);
119         // F = ma , a = F/m
120         double Ax = Fx / mass;
121         double Ay = Fy / mass;
122         // vt = v0 + a.t
123         this->xvelocity += step * Ax;
124         this->yvelocity += step * Ay;
125     }

```

3.12.5 Universe.h

```

1  /*****
2  *Copyright Shriya 2022
3  *Name: Shriya Thakur
4  *Date: February 13th, 2021
5  *Sources Of Help: Professor Daly in Office hours, Class Discord,
6  Classmates- Kevin Sprague, Ryan Casey.Tutor- Ben Friedman
7  *Assignemnt: PS2b- physics ptsd
8  *****/
9  #pragma once
10 #include "CelestialBody.h"
11 #include <vector>
12 #include <iostream>
13 #include <memory>
14 #include <SFML/Graphics.hpp>
15 #include <SFML/Window.hpp>
16 #include <SFML/System.hpp>
17
18 using std::vector;
19 using std::shared_ptr;
20
21 class Universe : public sf::Drawable {
22 public:
23     Universe();
24     void DisplayDeets();
25     void SetPlanetPosition(const int& viewport);
26     ~Universe();
27     void step(double seconds);

```

```

28     friend ostream& operator<<(ostream& out, Universe& obj);
29 private:
30     virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
31         const;
32     double radius;
33     int numPlanet;
34     vector<shared_ptr<CelestialBody>> planet;
35 };

```

3.12.6 Universe.cpp

```

1  /*****
2  *Copyright Shriya 2022
3  *Name: Shriya Thakur
4  *Date: February 17th, 2021
5  *Sources Of Help: Professor daly in Office hours, Class Discord,
6  *Classmate- Ryan Casey.
7  *Assignemnt: PS2a- Create UI for NBody simulation.
8  *Time taken: ~16 hours
9  *****/
10 #include "Universe.h"
11 #include <iostream>
12
13 using std::cin;
14 using std::cout;
15 using std::make_shared;
16 using std::endl;
17 using std::ostream;
18
19 void Universe::draw(sf::RenderTarget& target, sf::RenderStates states)
20     const {
21     for (int i = 0; i < numPlanet; i++) {
22         target.draw(*planet[i], states);
23     }
24 }
25 // Implementation for constructor
26 Universe::Universe() {
27     // scan input for number of planets and radius
28     cin >> numPlanet >> radius;
29     for (int i = 0; i < numPlanet; i++) {
30         planet.push_back(make_shared<CelestialBody>());
31         cin >> *planet.back();
32     }
33 }
34
35 void Universe::SetPlanetPosition(const int& viewport) {
36     for (int i = 0; i < numPlanet; i++) {
37         planet[i]->SetPos(radius, viewport);
38     }
39 }
40
41 void Universe::DisplayDeets() {
42     for (auto planetPtr : this->planet) {

```

```

43     cout << *planetPtr << endl;
44 }
45 }
46
47 Universe::~~Universe() {
48     // cout << "Universe destroyed" << endl;
49 }
50
51 void Universe::step(double seconds) {
52     // set all velocities
53     for (int i = 0; i < numPlanet; i++) {
54         planet[i]->NewVelocity(planet, seconds);
55     }
56     // set all positions
57     for (int i = 0; i < numPlanet; i++) {
58         planet[i]->NewPosition(seconds);
59     }
60 }
61
62 ostream& operator<<(ostream& out, Universe& obj) {
63     out << "Total Number of planets: " << obj.numPlanet;
64     out << endl;
65     out << "Radius: " << obj.radius << endl;
66     for (auto planetPtr : obj.planet) {
67         out << *planetPtr << endl;
68     }
69     return out;
70 }

```


4 PS3: Recursive Sierpinski

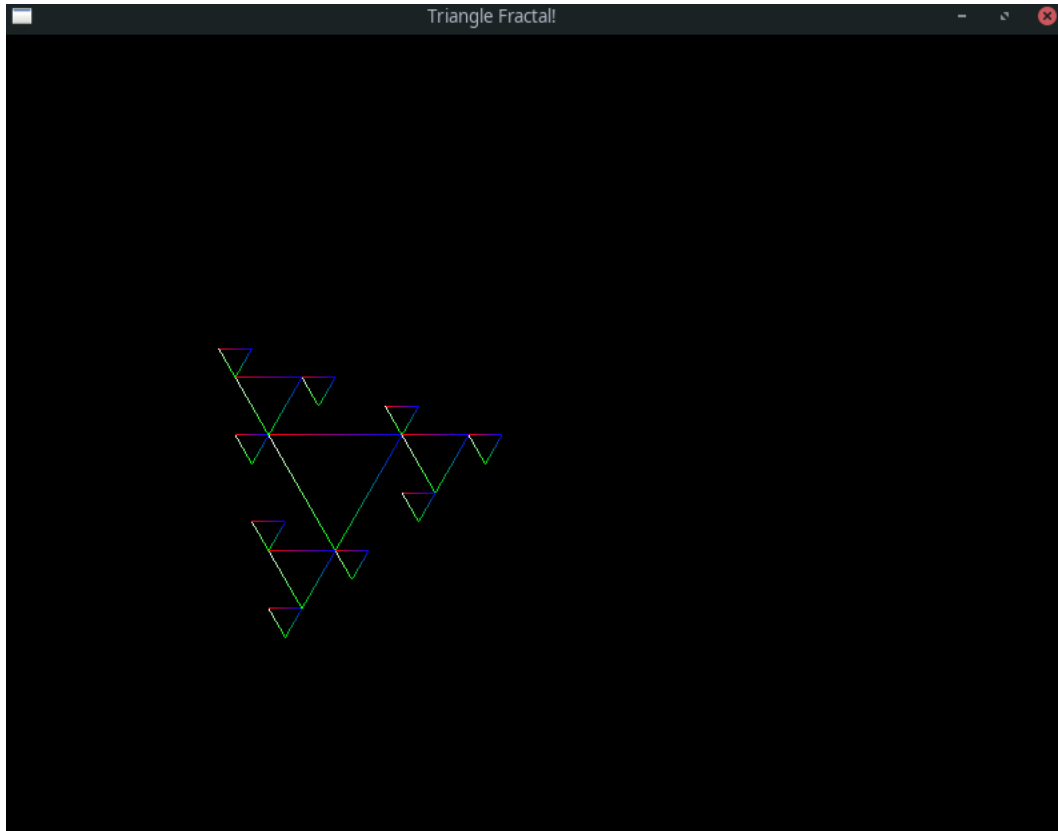


Figure 4: Sierpenski Triangles!

4.1 Discussion of assignment:

This assignment has us make triangles recursively at each vertex of the previous/parent triangle.

4.2 What I already knew

This assignment was based on Recursion. Recursion had been introduced back in Computing 1010. Recursion is faster compared to looping and is based on mathematical induction- Call function with an updated argument until we reach a base case. Give commands for what to do when it reaches that particular base case.

4.3 What I learned

I learned implementing recursive graphics using tools that sfml provides and increasing code readability. This was the first assignment that we had to Lint and get our coding style match up to Google's standards using cpplint.

4.4 Describe any Key Algorithms/Data Structures/OO designs

Inheritance: I had my Triangle class inherit publicly from sf::Drawable. I also used the inbuilt type VertexArray to store the vertices of the triangle and a vector2f to store the position of the triangles to keep track of them during recursion.

4.5 Does it work? If not, what problems are present?

Yes it works since Triangles are printed as expected and the code is bug free as depicted by the screenshot above.

4.6 Challenges:

1. Calculating updated positions where new triangles would be drawn took me a hot second
2. Initailly I was terribly confused about why we even needed a Triangle class since most things could be done inside main but that cleared up by asking prof Daly and my classmates.
3. Given the freedom we had for design implementation, I was overwhelmed with all the different options that we could choose from starting which classes to inherit from since all had advantages and disadvantages. I ended up inheriting from Drawable. I used VertexArray with LineStrip as my primitive since that sounded the most reasonable and had the most documentation available.
4. I was having casting issues from double to float so I did it explicitly using `static_cast<float>` since sf::Vector2f stores floats and not double values.
5. Lastly, I spent 40minutes just because I called `free` and then cleared the window inside main instantly after so none of the child triangles were printing.
6. I couldn't get the rectangle shape to enclose just the triangle in time.

4.7 List any other comments here:

I also did the extra credit and had the triangles change color!

4.8 Codebase

4.8.1 Makefile

```
1 CC = g++ --std=c++17
2 LIBS = -lsfml-graphics -lsfml-system -lsfml-window
3 CFLAGS = -Wall -Werror -pedantic
4
5 all: TFractal
6 TFractal: TFractal.o Triangle.o
```

```

7  $(CC) $(CFLAGS) -o TFractal TFractal.o Triangle.o $(LIBS)
8  TFractal.o: TFractal.cpp
9  $(CC) $(CFLAGS) -c TFractal.cpp $(LIBS)
10 Triangle.o: Triangle.cpp
11 $(CC) $(CFLAGS) -c Triangle.cpp $(LIBS)
12 clean:
13 rm -f TFractal.o Triangle.o TFractal
14 lint:
15 cpplint --filter=--runtime/references, -build/header_guard --extensions=
    cpp,h

```

4.8.2 TFractal.cpp (main)

```

1  /*****
2  Copyright 2022 Shriya
3  Name: Shriya Thakur
4  Date: Feb 28th, 2022
5  Sources Of help: Office hours, Kevin Sprague, Ryan Casey, class Discord
6  Time spent: 32hours
7  *****/
8  #include "Triangle.h"
9  #include <memory>
10 #include <iostream>
11 #include <cmath>
12 #include <vector>
13 #include <SFML/Graphics.hpp>
14 #include <SFML/Window.hpp>
15 #include <SFML/System.hpp>
16
17 using std::stod;
18 using std::stoi;
19 using sf::Vector2f;
20 using sf::Color;
21 using std::shared_ptr;
22 using std::vector;
23
24 void fTree(int recCount, Triangle origObj, sf::RenderTarget& target,
    vector<Triangle> ski) {
25     // base case
26     if ( recCount <= 0 ) {
27         return;
28     } else if ( recCount == 1 ) {
29         //target.draw(origObj);
30         //ski.push_back(this);
31         return;
32         //return origObj;
33     } else {
34         //shared_ptr<Triangle> newt;
35         // get coordinates of parent triangle
36         //target.draw(origObj);
37         sf::Vector2f parent = origObj.GetCoord();
38         float side_ = static_cast<float>(origObj.GetSide());
39         float newSide = static_cast<float>(origObj.GetSide()/2.0);
40         // set new coordinates of top child

```

```

41     sf::Vector2f topCoord = {
42         static_cast<float>(parent.x - (side_/4.0)),
43         static_cast<float>(parent.y - ((sqrt(3)/2.0 * newSide))));
44     Triangle topChild(newSide, topCoord);
45     fTree(recCount-1, topChild, target, ski);
46     //newt = fTree(recCount-1, topChild, target);
47     //target.draw(topChild);
48     ski.push_back(topChild);
49
50     // set new coordinates of left child aaaaaaa
51     sf::Vector2f lCoord;
52     lCoord = {static_cast<float>(parent.x),
53         static_cast<float>(parent.y + sqrt(3)/2.0 * side_)};
54     Triangle leftChild(newSide, lCoord);
55     fTree(recCount-1, leftChild, target, ski);
56     //newt = fTree(recCount-1, leftChild, target);
57     //target.draw(leftChild);
58     ski.push_back(leftChild);
59
60     // set new coordinates for right child
61     sf::Vector2f rCoord;
62     rCoord = {static_cast<float>(parent.x + side_),
63         static_cast<float>(parent.y)};
64     Triangle rightChild(newSide, rCoord);
65     fTree(recCount-1, rightChild, target, ski);
66     //target.draw(rightChild);
67     //newt = fTree(recCount-1, rightChild, target);
68     ski.push_back(rightChild);
69     //return newt;
70 }
71 }
72
73 int main(int argc, char** argv) {
74     // length of side of base equilateral triangle
75     double L = stod(argv[1]);
76     // depth of recursion
77     int N = stoi(argv[2]);
78     sf::Vector2f coord = {200.f, 300.f};
79     Triangle tri(L, coord);
80     //shared_ptr<Triangle> smallTri;
81     vector<Triangle> smallT;
82     sf::RenderWindow window(sf::VideoMode(800, 600), "Triangle Fractal!");
83     window.setFramerateLimit(60);
84     sf::Texture texture;
85     while (window.isOpen()) {
86         sf::Event event;
87         while (window.pollEvent(event)) {
88             if (event.type == sf::Event::Closed)
89                 window.close();
90         }
91         window.clear();
92         //smallTri = fTree(N, tri, window);
93         fTree(N, tri, window, smallT);
94         //window.draw(smallT);

```

```

95     //window.draw(tri);
96     window.display();
97 }
98 return 0;
99 }

```

4.8.3 Triangle.h

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Date: Feb 28th, 2022
5  Sources Of help: Office hours, Kevin Sprague, Ryan Casey, class Discord
6  Time spent: 32hours
7  *****/
8  #pragma once
9  #include <iostream>
10 #include <vector>
11 #include <SFML/Graphics.hpp>
12 #include <SFML/Window.hpp>
13 #include <SFML/System.hpp>
14
15 class Triangle: public sf::Drawable {
16 private:
17     double side;
18     sf::VertexArray triangle_vertices;
19     sf::Vector2f position_;
20     virtual void draw(sf::RenderTarget& target, sf::RenderStates state)
21         const;
22     // sf::Color triangle_color;
23     // sf::RectangleShape triangle_box;
24 public:
25     // constructors
26     Triangle() {}
27     Triangle(double l, sf::Vector2f pos);
28     // Destructor
29     ~Triangle() {}
30     // getters
31     double GetSide() const {
32         return side;
33     }
34     sf::Vector2f GetCoord() const {
35         return position_;
36     }
37     // void SetBox(sf::Vector2f left, sf::Vector2f right);
38 };

```

4.8.4 Triangle.cpp

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Date: Feb 28th, 2022
5  Sources Of help: Office hours, Kevin Sprague, Ryan Casey, class Discord

```

```

6 Time spent: 32hours
7 *****/
8 #include "Triangle.h"
9 #include <cmath>
10
11 Triangle::Triangle(double l, sf::Vector2f pos) {
12     position_ = pos;
13     side = l;
14     // set primitive type using function
15     triangle_vertices.resize(4);
16     triangle_vertices.setPrimitiveType(sf::LineStrip);
17     // start with coordinates passed
18     triangle_vertices[0].position = position_;
19     triangle_vertices[1].position = position_ + sf::Vector2f(side, 0);
20     triangle_vertices[2].position = position_ + sf::Vector2f((side/2),
21     (side * (sqrt(3)/2)));
22     triangle_vertices[3].position = position_;
23     // set color
24     triangle_vertices[0].color = sf::Color::Red;
25     triangle_vertices[1].color = sf::Color::Blue;
26     triangle_vertices[2].color = sf::Color::Green;
27     // this.setFillColor(newColor);
28     // triangle_color = newColor;
29 }
30 void Triangle::draw(sf::RenderTarget& target, sf::RenderStates state)
31     const {
32     target.draw(triangle_vertices, state);
33 }
34 /*void Triangle::SetBox(sf::Vector2f left, sf::Vector2f right)
35 {
36     triangle_box = sf::RectangleShape();
37     triangle_box.setPosition(left);
38     triangle_box.setSize(sf::Vector2f(right.x-left.x , right.y-left.y));
39 }*/

```

5 PS4: Circular Buffer and Guitar String simulation

5.1 Discussion

5.2 PS4a:

This assignment had us make a circular buffer API with fixed capacity to use in an upcoming project.

5.3 PS4b:

This assignment had us use the API we made to emulate a virtual keyboard.

5.4 What I already knew

I knew I would implement the circular buffer that would handle the string sounds. Along with the object oriented design concepts, we needed the sfml event class for the piano keystrokes and to ultimately produce sound. I used the STL queue from CPP since it makes the buffer circular for me.

5.5 What I learned

I didnt know much about sound syntesis or sound theory. This was also my first time learning about the Karplus-String algorithm. As I went through with this assignment, I realised how useful it is to generate a variety of plucked string sounds or even drum like timbre.

5.6 Challenges

5.6.1 ps4a:

PS4a was relatively easy. Implementing data structures in cpp is so much easier so I didnt really run into problems besides usual debugging issues. Calculating the time and space complexity was erroring until I found out what implementation type CPP used for their inbuilt queue type.

5.6.2 PS4b:

I ran into multiple issues writing KGuitarSim.cpp. Initially I passed in number of channels as 37 rather than 2 until I read documentation. I was also having a hard time with keyPressed so I have used TextEntered instead. While implementing a lambda I was confused what capture is for the longest time until I translated a ruby code to C++ and finally understood. Initially peek and dequeue were throwing an exception while attempting the extra credit until I added checks to see if that they wouldnt peek/dequeue if the queue was empty.

5.7 Describe any Key Algorithms/Data Structures/Object oriented designs

I used a queue data structure to implement the buffer. The buffer class consists of functions

1. A default constructor.
2. Value constructor to initialise the buffer with a given (fixed) capacity.
3. Accessor function to return the size of the buffer.
4. Boolean functions to check if the buffer is empty or full
5. Enqueue to push positive unsigned integers into the queue.
6. Dequeue to delete and return an item from the front of the queue.
7. peek to return (but not delete) items from the front of queue.

5.8 Does it work? If not, what problems are present?

Yes my CircularBuffer(mentioned StringSound down below) implementation passes all the unit tests mentioned. CircularBuffer class tested for:

1. capacity less than 1- testcase1
2. capacity 3 and empty queue should pass the isEmpty() test case- testcase1
3. Check size of queue(with capacity 10) after enqueueing 3 elememnts.- testcase2
4. Testing enqueue and dequeue - testcase3
5. test peek- tescase4 and 5
6. test size - testcase 6

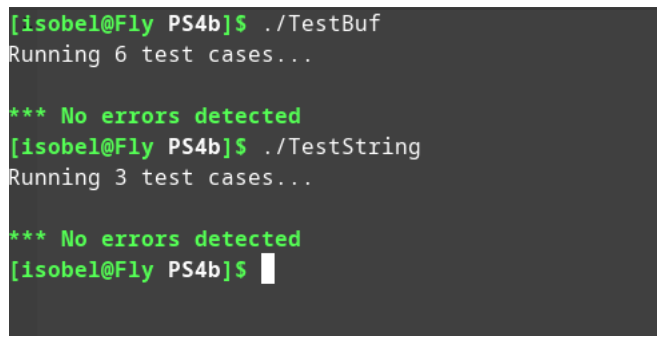
I used boost_require_no_throw to test some of the exceptions like when the buffer capacity is less than 1.

I finished the whole assignment. I implemented the StringSound class functions and had exceptions thrown when unwanted values were passed as input. I used the boost testing macros to test StringSound and CircularBuffer member functions. I listened carefully to the sounds emitted after pressing each keystroke. I threw an invalid_argument for when the frequency was a negative value in the stringSound constructor. I also threw invalid argument exception when an empty vector was passed into the second constructor. I used boost macros to check if the exceptions were handled properly and also to test tic and pluck using boost_require_throw and boost_check_equal.

yes this works, the notes go from low to high pitch as we travel from left to right on the string, as heard with the ear.

5.9 List any other comments here:

I attempted the extra credit for ps4a(using a lambda expression) and also for ps4b.The notes change from a low pitch(very flat) to higher pitch as we traverse across the keys given: "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' "



```
[isobel@Fly PS4b]$ ./TestBuf
Running 6 test cases...

*** No errors detected
[isobel@Fly PS4b]$ ./TestString
Running 3 test cases...

*** No errors detected
[isobel@Fly PS4b]$
```

Figure 5: Boost test for CircularBuffer and StringSound class work!

5.10 Codebase:

5.10.1 Makefile

```
1 CC = g++ -g --std=c++17
2 LIBS = -lsfml-graphics -lsfml-system -lsfml-window -lsfml-audio -
        lboost_unit_test_framework
3 CFLAGS = -Wall -Werror -pedantic
4 LINTFLAGS = --filter=--runtime/references,-build/header_guard,-build/c++11
        --extensions=cpp,h
5 all:KSGuitarSim TestBuf TestString
6 KSGuitarSim: KSGuitarSim.o CircularBuffer.o StringSound.o
7 $(CC) $(CFLAGS) -o KSGuitarSim CircularBuffer.o StringSound.o KSGuitarSim
        .o $(LIBS)
8 KSGuitarSim.o:KSGuitarSim.cpp
9 $(CC) $(CFLAGS) -c KSGuitarSim.cpp $(LIBS)
10 CircularBuffer.o: CircularBuffer.cpp CircularBuffer.h
11 $(CC) $(CFLAGS) -c CircularBuffer.cpp $(LIBS)
12 StringSound.o: StringSound.cpp StringSound.h
13 $(CC) $(CFLAGS) -c StringSound.cpp $(LIBS)
14 test.o: test.cpp
15 $(CC) $(CFLAGS) -c test.cpp $(LIBS)
16 StringSoundTest.o: StringSoundTest.cpp
17 $(CC) $(CFLAGS) -c StringSoundTest.cpp $(LIBS)
18 clean:
19 rm -f CircularBuffer.o StringSound.o KSGuitarSim.o test.o StringSoundTest
        .o TestBuf KSGuitarSim TestString
20 lint:
21 cpplint $(LINTFLAGS) CircularBuffer.h
22 cpplint $(LINTFLAGS) CircularBuffer.cpp
23 cpplint $(LINTFLAGS) StringSound.cpp
24 cpplint $(LINTFLAGS) StringSound.h
```

```

25  cpplint $(LINTFLAGS) test.cpp
26  cpplint $(LINTFLAGS) StringSoundTest.cpp
27  TestBuf: CircularBuffer.o StringSound.o test.o
28  $(CC) $(CFLAGS) -o TestBuf CircularBuffer.o StringSound.o test.o $(LIBS)
29  TestString: CircularBuffer.o StringSound.o StringSoundTest.o
30  $(CC) $(CFLAGS) -o TestString CircularBuffer.o StringSound.o
    StringSoundTest.o $(LIBS)

```

5.10.2 KSGuitarSim.cpp (main)

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Sources Of help: Office hours, class notes, Kevin Sprague, Ryan Casey
5  Date: March 28th 2022
6  *****/
7  #include <SFML/Graphics.hpp>
8  #include <SFML/System.hpp>
9  #include <SFML/Audio.hpp>
10 #include <SFML/Window.hpp>
11 #include <math.h>
12 #include <limits.h>
13 #include <iostream>
14 #include <string>
15 #include <exception>
16 #include <stdexcept>
17 #include <vector>
18 #include <memory>
19 #include "CircularBuffer.h"
20 #include "StringSound.h"
21
22 #define CONCERT_A 220.0
23 #define SAMPLES_PER_SEC 44100
24 #define NUM_KEYS 37
25 using std::vector;
26 using std::string;
27 using std::runtime_error;
28 using std::make_shared;
29 using std::shared_ptr;
30 vector<sf::Int16> makeSamples(StringSound& gs) {
31     vector<sf::Int16> samples;
32     gs.pluck();
33     int duration = 8; // seconds
34     for (int i= 0; i < SAMPLES_PER_SEC * duration; i++) {
35         gs.tic();
36         samples.push_back(gs.sample());
37     }
38     return samples;
39 }
40
41 int main(int argc, char** argv) {
42     sf::RenderWindow window(sf::VideoMode(300, 200), "Virtual Keyboard!");
43     sf::Event event;
44     sf::Sound sound;

```

```

45     double freq = 0;
46     vector<vector<sf::Int16>> samples;
47     vector<shared_ptr<sf::SoundBuffer>> KeyboardSoundBuf;
48     vector<sf::Sound> KeyboardSounds;
49     string Keyboard = "q2we4r5ty7u8i9op-[]=zxdcfvgbnjmk,.;/' ";
50     for(int i = 0; i < NUM_KEYS; i++)
51     {
52         // set frequency
53         freq = 440.0 * pow(2.0, ((i - 24) / 12.0));
54         // create stringSound object with given frequency
55         StringSound obj(freq);
56         // push into samples
57         samples.push_back(makeSamples(obj));
58         shared_ptr<sf::SoundBuffer> buffer_ = make_shared<sf::SoundBuffer>();
59         KeyboardSoundBuf.push_back(buffer_);
60         if (!buffer_>loadFromSamples(&samples[i][0], samples[i].size(), 1,
61             SAMPLES_PER_SEC))
62         {
63             throw runtime_error("sf::SoundBuffer: failed to load from samples.");
64         }
65         sound.setBuffer(*buffer_);
66         KeyboardSounds.push_back(sound);
67     }
68     while (window.isOpen()) {
69         while (window.pollEvent(event)) {
70             if(event.type == sf::Event::Closed)
71             {
72                 window.close();
73             }
74             if (event.type == sf::Event::TextEntered)
75             {
76                 for(int i = 0; i < NUM_KEYS; i++)
77                 {
78                     if (Keyboard[i] == static_cast<unsigned char>(event.text.unicode))
79                     {
80                         KeyboardSounds[i].play();
81                         break;
82                     }
83                 }
84                 window.clear();
85                 window.display();
86             }
87         }
88         return 0;
89     }

```

5.10.3 CircularBuffer.h

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Sources Of help: Office hours, class notes

```

```

5 Date: March 3rd 2022
6 *****/
7 #pragma once
8 #include <stdint.h>
9 #include <stdbool.h>
10 #include <iostream>
11 #include <vector>
12 #include <queue>
13 #include <SFML/Graphics.hpp>
14 #include <SFML/Window.hpp>
15 #include <SFML/System.hpp>
16
17 using std::vector;
18 using std::queue;
19
20 class CircularBuffer {
21 public:
22     // Create and empty ring buffer
23     CircularBuffer() {}
24     // Create an empty ring buffer with default max capacity
25     explicit CircularBuffer(size_t capacity);
26     size_t size() const;
27     bool isEmpty();
28     bool isFull();
29     void enqueue(int16_t x);
30     int16_t dequeue();
31     int16_t peek();
32     void _Empty();
33     ~CircularBuffer() {}
34 private:
35     unsigned int buffer_capacity;
36     queue<int16_t> buffer;
37 };

```

5.10.4 CircularBuffer.cpp

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Sources Of help: Office hours, class notes
5  Date: March 3rd 2022
6  *****/
7  #include "CircularBuffer.h"
8
9  using std::vector;
10 using std::invalid_argument;
11 using std::runtime_error;
12 using std::queue;
13
14 CircularBuffer::CircularBuffer(size_t capacity) {
15     if (capacity < 1)
16         throw invalid_argument
17             ("CircularBuffer constructor: capacity must be greater than zero.");
18     buffer_capacity = capacity;

```

```

19 }
20
21 size_t CircularBuffer::size() const {
22     return buffer.size();
23 }
24
25 bool CircularBuffer::isEmpty() {
26     return buffer.size() == 0;
27 }
28
29 bool CircularBuffer::isFull() {
30     return buffer.size() == buffer_capacity;
31 }
32 // Add item x to front
33 void CircularBuffer::enqueue(int16_t x) {
34     // runtime error if buffer is full
35     if (this->buffer.size() >= this->buffer_capacity)
36         throw runtime_error("enqueue: can't enqueue to a full ring");
37     buffer.push(x);
38 }
39 // Delete and return item from front
40 int16_t CircularBuffer::dequeue() {
41     if (isEmpty())
42         throw runtime_error("dequeue: can't dequeue from empty ring");
43     int16_t x = buffer.front();
44     buffer.pop();
45     return x;
46 }
47 // Return but do not delete item from front
48 int16_t CircularBuffer::peek() {
49     if (!isEmpty())
50         return buffer.front();
51     else
52         throw runtime_error("peek: can't peek from empty ring");
53 }
54
55 // Empty buffer
56 void CircularBuffer::_Empty() {
57     while (!this->isEmpty()) {
58         this->dequeue();
59     }
60 }

```

5.10.5 test.cpp

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Sources Of help: Office hours, class notes, Kevin Sprague, Ryan Casey
5  Date: March 28th 2022
6  *****/
7  #include "CircularBuffer.h"
8  #include "StringSound.h"
9  #include <stdexcept>

```

```

10 #include <exception>
11
12 using std::cout;
13 using std::endl;
14 using std::invalid_argument;
15 using std::out_of_range;
16 using std::runtime_error;
17
18 #define BOOST_TEST_DYN_LINK
19 #define BOOST_TEST_MODULE Main
20 #include <boost/test/unit_test.hpp>
21
22 BOOST_AUTO_TEST_CASE(testCase1) {
23     BOOST_REQUIRE_NO_THROW(CircularBuffer b1(-1));
24     CircularBuffer b2(3);
25     BOOST_TEST((b2.isEmpty()));
26 }
27
28 BOOST_AUTO_TEST_CASE(testCase2) {
29     CircularBuffer b2(10);
30     b2.enqueue(34);
31     b2.enqueue(49);
32     b2.enqueue(78);
33     BOOST_CHECK_EQUAL(b2.size(), 3);
34     BOOST_TEST(!(b2.isEmpty()));
35 }
36
37 BOOST_AUTO_TEST_CASE(testCase3) {
38     // test dequeue
39     CircularBuffer b4(3);
40     b4.enqueue(1);
41     b4.enqueue(4);
42     b4.enqueue(9);
43     BOOST_TEST(((b4.isFull())));
44     b4.dequeue();
45     b4.dequeue();
46     b4.dequeue();
47     BOOST_TEST((b4.isEmpty()));
48 }
49
50 BOOST_AUTO_TEST_CASE(testCase4) {
51     // test peek
52     CircularBuffer b4(5);
53     b4.enqueue(1);
54     b4.enqueue(4);
55     b4.enqueue(9);
56     b4.enqueue(16);
57     b4.enqueue(25);
58     BOOST_TEST((b4.isFull()));
59     int16_t x = b4.peek();
60     BOOST_REQUIRE_EQUAL(x, 1);
61     BOOST_TEST(((b4.isFull())));
62 }
63

```

```

64 BOOST_AUTO_TEST_CASE(testCase5) {
65     // test peek on empty queue
66     BOOST_REQUIRE_NO_THROW(CircularBuffer b5(7));
67     CircularBuffer b5(4);
68     b5.enqueue(4);
69     b5.enqueue(7);
70     b5.enqueue(8);
71     b5.enqueue(15);
72     BOOST_REQUIRE_EQUAL(b5.peek(), 4);
73 }
74
75 BOOST_AUTO_TEST_CASE(testCase6) {
76     // test size
77     CircularBuffer b6(7);
78     BOOST_REQUIRE_EQUAL(b6.size(), 0);
79 }

```

6 PS5: DNA Sequence Alignment

6.1 Discussion

We used dynamic programming and reverse engineered. First computed the edit distance between the two input strings. We first built the matrix bottom to top, and right to left. In each of the boxes, filled in the optimal edit distance which was calculated using a written min to calculate the min of three numbers.(referred to formula: $opt[i][j] = \min \{opt[i+1][j+1] + 0/1, opt[i+1][j] + 2, opt[i][j+1] + 2\}$ from the pdf) which gave us the best distance in the [0][0] box. then we traverse the matrix again, this time in reverse order, to recover our computed edit distance. Needleman-Wunsch Using a helper function to return the number in each cell of the matrix instead of doing the math each time. modifiedAt(int row, int col) For ease, we used a helper function that would give us the value at any point in the matrix when passed the rows and column number to it.

6.2 Comments on pair programming approach:

We brainstormed the algorithm- using needleman Wunsch and how to implement it. before that decided on what data structures we would be using. After much fiddling with using string pushback in alignment, we decided to build the string using stringstream instead.

6.3 What I already knew:

A genetic sequence is a string formed by four bases: Adenine (A), Thymine (T), Guanine(G), Cytosine (C). I also knew that I would need dynamic programming and a 2D matrix structure and that Needleman Wunsch would come in handy.

6.4 What I learned:

I learned that dynamic programming is an optimization technique that breaks down the big issue at hand into smaller sub problems and solves them and the optimal solution to the big problem depends on the optimal solution to the smaller sub problems.

6.5 Challenges:

I was having out of bounds errors all the time. I think this assignment taught me the value of size_t. At one point my core was dumping on every other run and not every run of the program which was weird until I separated the initialization of i and j in alignment() to different lines. Besides this, because of pair programming, I wasn't too stuck.

6.6 Valgrind Analysis:

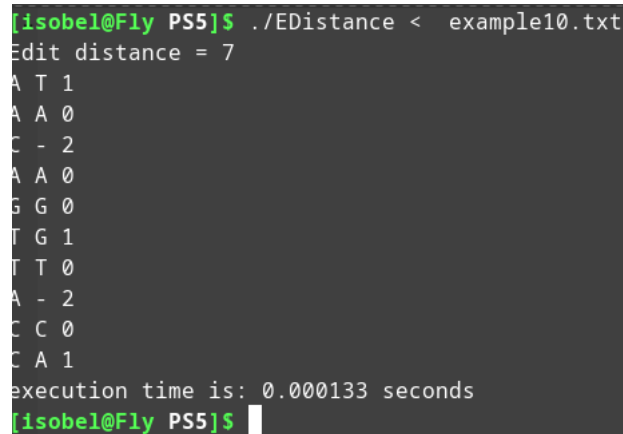
On running each of these with valgrind, no memory leaks or errors were detected in any of example10.txt, ecoli2500.txt, ecoli5000.txt, ecoli10000.txt. However, while testing the ecoli20000 and upward, it took too long even without valgrind so i couldnt rerun with valgrind. Approximately 400mb were used while running the ecoli10000.txt as said by valgrind.

6.7 Describe any key algorithms/Data structures or object oriented designs used:

I used strings to store the two DNA sequences that we read from file. I used `size_t` to store the number of columns and rows and a vector of Integers for the matrix. abstraction- Only member functions of EDistance class can edit the matrix.

6.8 Does it work, if not, what problems are present?

Yes this assignment works as expected for all the files given besides some where it took too long to run.



```
[isobel@Fly PS5]$ ./EDistance < example10.txt
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: 0.000133 seconds
[isobel@Fly PS5]$
```

Figure 6: Output for example10.txt

6.9 List any other comments here:

CPU speed(in MHz): 2100MHz == 2.1 GHz Implementation method: Needleman Wunsch
Matrix method: vector of integers Operating System: Linux Distribution:Manjaro Model
name: Intel(R) Core(TM) i3-8145U

6.10 Codebase

6.10.1 Makefile

```
1 CC = g++ --std=c++17
2 LIBS = -lsfml-system
3 CFLAGS = -O3 -Wall -Werror -pedantic -g -gdb
4
5 all:EDistance
6 EDistance: EDistance.o main.o
7 $(CC) $(CFLAGS) -o EDistance EDistance.o main.o $(LIBS)
8 EDistance.o: EDistance.cpp EDistance.h
9 $(CC) $(CFLAGS) -c EDistance.cpp $(LIBS)
10 main.o: main.cpp
```

```

11 $(CC) $(CFLAGS) -c main.cpp $(LIBS)
12 clean:
13 rm -f main.o EDistance.o test.o EDistance
14 lint:
15 cpplint main.cpp
16 cpplint EDistance.h
17 cpplint EDistance.cpp

```

6.10.2 main.cpp

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Date: April 3rd, 2022
5  Partner: Kevin Sprague
6  *****/
7  #include "EDistance.h"
8  #include <iostream>
9  #include <SFML/System.hpp>
10
11 using sf::Clock;
12 using sf::Time;
13 using std::cout;
14 using std::cin;
15 using std::endl;
16 int main(int argc, char** argv) {
17     Clock clock;
18     string x, y;
19     cin >> x >> y;
20     EDistance E(x, y);
21     int edDistance = E.optDistance();
22     cout << "Edit distance = " << edDistance << endl;
23     string a = E.alignment();
24     Time t = clock.getElapsedTime();
25     cout << a;
26     // E.print();
27     cout << "execution time is: " << t.asSeconds() << " seconds \n";
28     return 0;
29 }

```

6.10.3 EDistance.h

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Date: April 3rd, 2022
5  Partner: Kevin Sprague
6  *****/
7  #pragma once
8  #include <iostream>
9  #include <vector>
10 #include <algorithm>
11 #include <string>
12 #include <ctime>

```

```

13
14 using std::string;
15 using std::vector;
16 using std::size_t;
17 class EDistance {
18 public:
19     // constructor that accepts two strings
20     EDistance(string a, string b);
21     // penalty of aligning characters
22     static int penalty(char a, char b);
23     // returns minimum of three arguments
24     static int min(int a, int b, int c);
25     int optDistance();
26     string alignment();
27     int& modifiedAt(int row, int column) {
28         unsigned int a = 0;
29         a = col_ * row + column;
30         return matrix_.at(a);
31     }
32     void print();
33 private:
34     string x_;
35     string y_;
36     size_t col_;
37     size_t row_;
38     vector<int> matrix_;
39 };

```

6.10.4 EDistance.cpp

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Date: April 3rd, 2022
5  Partner: Kevin Sprague
6  *****/
7  #include "EDistance.h"
8  #include <iostream>
9  #include <vector>
10 #include <string>
11 #include <sstream>
12
13 using std::string;
14 using std::vector;
15 using std::size_t;
16 using std::stringstream;
17 using std::cout;
18 using std::endl;
19 using std::ostream;
20 // constructor that accepts two strings
21 EDistance::EDistance(string a, string b) {
22     x_ = a;
23     y_ = b;
24     row_ = a.length() + 1;

```

```

25     col_ = b.length() + 1;
26     matrix_.resize((row_) * (col_));
27 }
28 // penalty of aligning characters
29 int EDistance::penalty(char a, char b) {
30     if (a == b)
31         return 0;
32     else
33         return 1;
34 }
35 // returns minimum of three arguments
36 int EDistance::min(int a, int b, int c) {
37     int min;
38     if (a <= b)
39         min = a;
40     else
41         min = b;
42     if (min <= c)
43         return min;
44     else
45         return c;
46 }
47 // Build matrix and return element at 0, 0
48 int EDistance::optDistance() {
49     // traverse matrix top to bottom, left to right
50     // last column
51     for (size_t i = 0; i < row_; i++) {
52         modifiedAt(i, (col_ - 1)) = 2 * (row_ - 1 - i);
53     }
54     // bottomest row
55     for (size_t j = 0; j < col_; j++) {
56         modifiedAt((row_ - 1), j) = 2 * (col_ - 1 - j);
57     }
58     for (size_t i = row_ - 1; i > 0; i--) {
59         for (size_t j = col_ - 1; j > 0; j--) {
60             char a = x_[i-1];
61             char b = y_[j-1];
62             // opt[i][j] = min { opt[i+1][j+1] + 0/1, opt[i+1][j] + 2, opt[i][j]
               + 1 } + 2 }//NOLINT
63             modifiedAt(i-1, j-1) = min(modifiedAt(i, j) + penalty(a, b),
64                 (modifiedAt(i, j-1) + 2), modifiedAt(i-1, j) + 2);
65         }
66     }
67     return modifiedAt(0, 0);
68 }
69
70 string EDistance::alignment() {
71     size_t i = 0;
72     size_t j = 0;
73     stringstream ss;
74     while (i < row_ - 1 || j < col_ - 1) {
75         if (i == row_ - 1) {
76             ss << "- " << y_[j] << " 2\n";
77             j++;

```

```

78     } else if (j == col_-1) {
79         ss << x_[i] << "- 2\n";
80         i++;
81     } else {
82         if (modifiedAt(i, j) == modifiedAt(i+1, j)+2) {
83             ss << x_[i] << " - 2\n";
84             i++;
85         } else if (modifiedAt(i, j) == modifiedAt(i, j+1)+2) {
86             ss << "- " << y_[j] << " 2\n";
87             j++;
88         } else {
89             ss << x_[i] << " " << y_[j] << " " << penalty(x_[i], y_[j]) << "\n"
90                 ;
91             i++;
92             j++;
93         }
94     }
95     return ss.str();
96 }
97 void EDistance::print() {
98     // cout << "string 1: " << x_ << endl << "string 2: " << y_ << endl;
99     for (size_t i = 0; i < row_; i++) {
100         for (size_t j = 0; j < col_; j++) {
101             cout << modifiedAt(i, j) << "\t";
102         }
103         cout << endl;
104     }
105 }

```

7 PS6: Frequency Analysis

7.1 Discussion

Analyze an input text for transition between a fixed number of characters and the following letter and produce a probabilistic model(Markov model) of the text and use that to generate gibberish that is surprisingly reasonable. This assignment asks us to do a frequency analysis. It generates text based on text that it is given, uses sizes of chunks of it to parse, and a size to output. It uses random character generation based on observed probabilities in order to create somewhat readable text. It almost feels like "suggestions" that my phone gives me when typing a text to someone.

7.2 What I already knew

I also knew I needed a map or dictionary of strings as keys and a vector of characters as value, an integer to store the order of our Markov Model, and a string of text.

7.3 What I learned

I learned more about Markov models and how they work and use that to recognize and predict patterns in any text provided. The constructor which I was particularly proud of makes it circular by appending the first "k" characters and then loop. In `kRand()`, I used the `std::random_device` as a random number generator and called that repeatedly in `generate` on the initial input string to generate the next character. Successive k-grams were formed by using the most recent k characters in the newly generated text.

7.4 Challenges:

Initially I was having a lot of memory access errors. After that, due to multiple data structures used- maps, strings, vector of characters, I was confused while accessing a particular element of each so that was showing errors. It took me a while to figure out the implementation for `generate()`. After all that, I was having trouble to use a lambda as an argument. Using "accumulate" would be ideal and pass a lambda to it.

7.5 Describe any key Algorithms, Data Structures Or Object Oriented designs used:

Maps were incredibly useful to me for storing and accessing kgrams and their frequencies. I used a map of strings(key value) and vector of characters(mapped value) I had to make use of the friend specifier in the insertion `||` operator overloader, which required specifying the object each time I referenced it. To make it circular, I basically appended the first "k" characters and had it loop so we dont read too far. I also used `std::random_device` in `kRand()` to generate non deterministic random numbers. I used the `map.find()` and `map.end()` as well inside `kRand()` and if the target wasnt found I had it throw an exception.

7.6 Did you complete the whole assignment, does it work? If not, what problems are present?

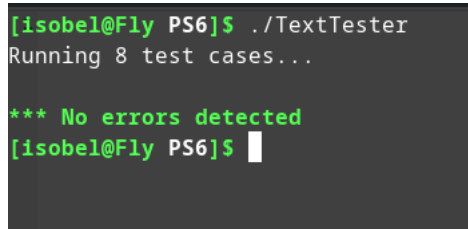
Yes, I completed the entire assignment. Before writing TextWriter.cpp, I wrote test.cpp which uses the Boost testing library to test each function and their exceptions, a coding style known as Test Driven Development. I used BOOST_REQUIRE_THROW macro when testing exceptions. Yes my implementation passes the unit tests I set up. I implemented 8 tests. I used the BOOST_REQUIRE_THROW to test for exceptions and BOOST_REQUIRE to see if the code worked with (good) input passed.

7.7 List any comments here:

I used a lambda inside RandWriter.cpp in freq(string, char): return count_if(m_map.at(kgram).begin(), m_map.at(kgram).end(), [=](int a) return a == c;); TestBuf is the executable to test functions of the circular buffer class and TestString is the executable to test the StringSound class.

1. Make all: Builds two executables- TextWriter and TextTester.
2. TextTester builds test.cpp which contains all unit tests.
3. TextWriter is the driver program.

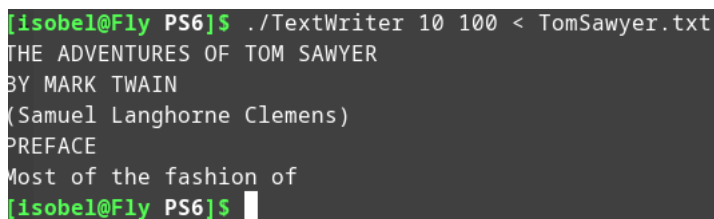
All my files are linted and meet Google standards of coding.



```
[isobel@Fly PS6]$ ./TextTester
Running 8 test cases...

*** No errors detected
[isobel@Fly PS6]$
```

Figure 7: Boost testing for functions of RandWriter class!



```
[isobel@Fly PS6]$ ./TextWriter 10 100 < TomSawyer.txt
THE ADVENTURES OF TOM SAWYER
BY MARK TWAIN
(Samuel Langhorne Clemens)
PREFACE
Most of the fashion of
[isobel@Fly PS6]$
```

Figure 8: Markov Model Of Natural Language

7.8 Codebase

7.8.1 Makefile

```
1 CC = g++ --std=c++17
2 LIBS = -lsfml-graphics -lsfml-system -lsfml-window -
        lboost_unit_test_framework
3 CFLAGS = -g -Wall -Werror -pedantic
4
5 all:TextWriter TextTester
6 TextWriter: TextWriter.o RandWriter.o
7 $(CC) $(CFLAGS) -o TextWriter TextWriter.o RandWriter.o $(LIBS)
8 TextTester: test.o RandWriter.o
9 $(CC) $(CFLAGS) -o TextTester test.o RandWriter.o $(LIBS)
10 RandWriter.o: RandWriter.cpp RandWriter.h
11 $(CC) $(CFLAGS) -c RandWriter.cpp $(LIBS)
12 TextWriter.o: TextWriter.cpp
13 $(CC) $(CFLAGS) -c TextWriter.cpp $(LIBS)
14 test.o: test.cpp
15 $(CC) $(CFLAGS) -c test.cpp $(LIBS)
16 clean:
17 rm -f RandWriter.o TextWriter.o TextWriter test.o TextTester
18 lint:
19 cpplint --filter=-runtime/references test.cpp
20 cpplint RandWriter.h
21 cpplint RandWriter.cpp
22 cpplint TextWriter.cpp
```

7.8.2 main.cpp

```
1 /*
        *****
2 Copyright Shriya 2022
3 Name: Shriya Thakur
4 Date: April 11th, 2021
5 Sources Of help: Dr. Daly, CClassmate- Keving Sprague
6 *****
        */
7 #include "RandWriter.h"
8 #include <iostream>
9 #include <stdexcept>
10 #include <functional>
11
12 using std::string;
13 using std::cin;
14 using std::endl;
15 using std::cout;
16
17 int main(int argc, char** argv) {
18     // 3 command line arguments needed
19     if (argc != 3)
20         return -987353;
21     string text = "";
```



```

22     string line;
23     while (getline(cin, line))
24         text += line + "\n";
25     RandWriter rw(text, atoi(argv[1]));
26     string kgram = text.substr(0, atoi(argv[1]));
27     cout << rw.generate(kgram, atoi(argv[2])) << endl;
28     return 0;
29 }

```

7.8.3 RandWriter.h

```

1 // Copyright Shriya 2022
2 #pragma once
3 #include <iostream>
4 #include <map>
5 #include <string>
6 #include <vector>
7
8 using std::map;
9 using std::string;
10 using std::vector;
11 using std::ostream;
12
13 class RandWriter {
14 public:
15     RandWriter() {}
16     RandWriter(string text, int k);
17     int orderK() const;
18     int freq(string kgram) const;
19     int freq(string kgram, char c) const;
20     char kRand(string kgram);
21     string generate(string kgram, int L);
22     ~RandWriter() {}
23     friend ostream& operator<<(ostream& out, RandWriter& rw);
24 private:
25     map<string, vector<char>> m_map;
26     string m_string;
27     int m_order;
28 };

```

7.8.4 RandWriter.cpp

```

1 /*
    *****
2 Copyright Shriya 2022
3 Name: Shriya Thakur
4 Date: April 11th, 2021
5 Sources Of help: Dr. Daly, CClassmate- Keving Sprague
6 *****
    */
7 #include "RandWriter.h"
8 #include <stdexcept>
9 #include <random>

```

```

10 #include <vector>
11 #include <functional>
12 #include <algorithm>
13
14 using std::map;
15 using std::string;
16 using std::invalid_argument;
17 using std::random_device;
18 using std::uniform_int_distribution;
19 using std::cout;
20 using std::runtime_error;
21 using std::ostream;
22 using std::endl;
23 using std::find;
24
25 // program behaviour doesnt change depending on order
26 RandWriter::RandWriter(string text, int k) {
27     if (k < 0)
28         throw runtime_error("Constructor(): order cannot be negative");
29     m_order = k;
30     m_string = text;
31     string circular = m_string + m_string.substr(0, m_order);
32     auto len = m_string.length();
33     string temp = "";
34     for (unsigned int i = 0; i < len; i++) {
35         temp = circular.substr(i, m_order);
36         char ch = circular[i+m_order];
37         m_map[temp].push_back(ch);
38     }
39 }
40
41 // number of occurrences of kgram in text
42 int RandWriter::orderK() const {
43     return m_order;
44 }
45
46
47 int RandWriter::freq(string kgram) const {
48     if (static_cast<int>(kgram.length()) != m_order)
49         throw runtime_error("freq(string): String length doesnt equal order");
50     else
51         return m_map.at(kgram).size();
52 }
53
54 int RandWriter::freq(string kgram, char c) const {
55     if (static_cast<int>(kgram.length()) != m_order)
56         throw runtime_error("freq(string, char): String length doesnt equal
57                             order");
58     if (m_map.find(kgram) == m_map.end())
59         return 0;
60     return count_if (m_map.at(kgram).begin(), m_map.at(kgram).end(),
61                     [=](int a){ return a == c; } );
62 }

```

```

63 char RandWriter::kRand(string kgram) {
64     if (static_cast<int>(kgram.length()) != m_order)
65         throw runtime_error("kRand(): String length didnt match order");
66     else if (m_map.find(kgram) == m_map.end())
67         throw runtime_error("kRand():kGram doesnt exist");
68     random_device dev;
69     uniform_int_distribution<int> dist(0, m_map.at(kgram).size() - 1);
70     return m_map.at(kgram).at(dist(dev));
71 }
72
73
74 string RandWriter::generate(string kgram, int L) {
75     if (static_cast<int>(kgram.size()) != m_order) {
76         throw runtime_error("generate: kgram is not of size == order");
77     }
78     string out = kgram;
79     if (m_order == 0) {
80         for (int i = 0; i < L; i++)
81             out.push_back(kRand(""));
82         return out;
83     }
84     if (m_map[kgram].size() <= 0)
85         throw runtime_error("generate: Kgram size is less than or equal to 0");
86     while (static_cast<int>(out.size()) < L) {
87         out.push_back(kRand(kgram));
88         kgram = out.substr(out.size() - m_order, m_order);
89     }
90     return out;
91 }
92
93 ostream& operator<<(ostream& out, RandWriter& rw) {
94     // map iterator
95     for (const auto& a : rw.m_map) {
96         out << a.first << " : [";
97         for (size_t i = 0; i < (a.second.size()); i++) {
98             if (i == (a.second.size() - 1))
99                 out << a.second[i] << " ";
100             else
101                 out << a.second[i] << ", ";
102         }
103         out << "]\n";
104     }
105     return out;
106 }

```

7.8.5 test.cpp

```

1  /*****
2  Copyright Shriya 2022
3  Name: Shriya Thakur
4  Sources of help:Dr Daly, class discord Classmate- Kevin Sprague
5  Date:April 11, 2022
6  *****/

```

```

7  #include "RandWriter.h"
8  #include <iostream>
9  #include <string>
10 #include <map>
11 #include <vector>
12
13 using std::cout;
14 using std::endl;
15 using std::invalid_argument;
16 using std::out_of_range;
17 using std::runtime_error;
18
19 #define BOOST_TEST_DYN_LINK
20 #define BOOST_TEST_MODULE Main
21 #include <boost/test/unit_test.hpp>
22
23 BOOST_AUTO_TEST_CASE(testConstructor) {
24     BOOST_REQUIRE_THROW(RandWriter r("gagggagaggcgagaaa", -1), runtime_error);
25 }
26
27 BOOST_AUTO_TEST_CASE(testOrderK) {
28     RandWriter r("gagggagaggcgagaaa", 6);
29     BOOST_REQUIRE(r.orderK() == 6);
30 }
31
32 BOOST_AUTO_TEST_CASE(testFreq) {
33     int x;
34     RandWriter r("gagggagaggcgagaaa", 0);
35     BOOST_REQUIRE_NO_THROW(x = r.freq(""));
36     BOOST_REQUIRE(x == 17);
37 }
38
39 BOOST_AUTO_TEST_CASE(testFreq2) {
40     RandWriter r("abba", 3);
41     BOOST_REQUIRE(r.freq("abb", 'a') == 1);
42     RandWriter r1("gagggagaggcgagaaa", 3);
43     BOOST_REQUIRE(r1.freq("gag", 'g') == 2);
44 }
45
46 BOOST_AUTO_TEST_CASE(testkRand) {
47     RandWriter r("abcdef", 4);
48     char c = r.kRand("abcd");
49     BOOST_REQUIRE(c == 'e');
50     RandWriter r1("gagggagaggcgagaaa", 8);
51     char c1 = r1.kRand("gagggaga");
52     BOOST_REQUIRE(c1 == 'g');
53 }
54 BOOST_AUTO_TEST_CASE(testkRandException) {
55     RandWriter r("abcdef", 4);
56     BOOST_REQUIRE_THROW(r.kRand("abc"), runtime_error);
57     RandWriter r1("gagggagaggcgagaaa", 5);
58     BOOST_REQUIRE_THROW(r1.kRand("gaa"), runtime_error);
59 }
60 BOOST_AUTO_TEST_CASE(testGenerate) {

```

```

61  RandWriter r("abcd", 3);
62  string s = r.generate("abc", 4); //NOLINT
63  BOOST_REQUIRE(s == "abcd");
64  RandWriter r1("gagggagaggcgagaaa", 5);
65  string s1 = r1.generate("gaggc", 6); //NOLINT
66  BOOST_REQUIRE(s1 == "gaggcg");
67 }
68 BOOST_AUTO_TEST_CASE(testGenerateException) {
69     string s; //NOLINT
70     RandWriter r("abc", 3);
71     BOOST_REQUIRE_THROW(s = r.generate("ab", 4), runtime_error);
72     string s1; //NOLINT
73     RandWriter r1("gagggagaggcgagaaa", 3);
74     BOOST_REQUIRE_THROW(s1 = r1.generate("gagg", 4), runtime_error);
75 }

```

8 PS7:Kronos Time Parsing

8.1 Discussion

This assignment had us Parse a log file at every boot cycle and record the Time Stamps. I used a combination of boost regexes and strings. I first read in input from the file name passed as argument. I then ran a loop on it until end. I used `regex_search` to see if we needed to start a boot cycle and print appropriate messages accordingly and time of boot started. Similarly, I used `regex_search` to see if we found the matching boot ending regex. In this case, calculated the duration this boot cycle took and set my `isBooting` flag to false once over before printing out data to the log report file.

8.2 What I already knew

I knew what regular expressions were and how their syntax works since I took Foundations Of Computer Science in an earlier semester. I knew how to parse a particular statement and use a target to see if the regex we made matches that target provided.

8.3 What I learned?

I learned using regular expressions to parse log files and match them with boot messages using the `regex_search`. This gave me a deeper understanding of what the Boost regex library has to offer and what all it supports.

8.4 Challenges:

Initially when checked the sample cpp files put on blackboard, I was frantically confused. After some thought, I realized we just needed regexes to match up starting and ending boot and for time stamps. I spent sometime writing them down and using the proper sequences for each character present. After all this, I spent sometime getting my loop to run as expected and also entered a flag to check if the system is booting or not and reset it when needed. After all this, I was struggling with writing a lambda because I was having trouble with what to return value if the `regex_search` failed until I explicitly specified return type to string.

8.5 Describe any key Algorithms/Data Structures or Object Oriented designs used:

I used Boost regular expressions to record the start and ending Boot messages and also for time stamps. I used a string to store the actual message and then used `regex_search` to see if the regular expression initialised matched the string.

8.6 Does it work? If not, what problems are present

Yes it works. We were given the sample output file for when we parse the 5th log file. To test my output, I used *vimdiff* and compared my generated report file for the 5th log file

to the file we were given. As seen in 9 the outputs match exactly thus no differences are highlighted.

8.7 list any comments here

The files were linted and I attempted the extra credit. I made a lambda to extract time.

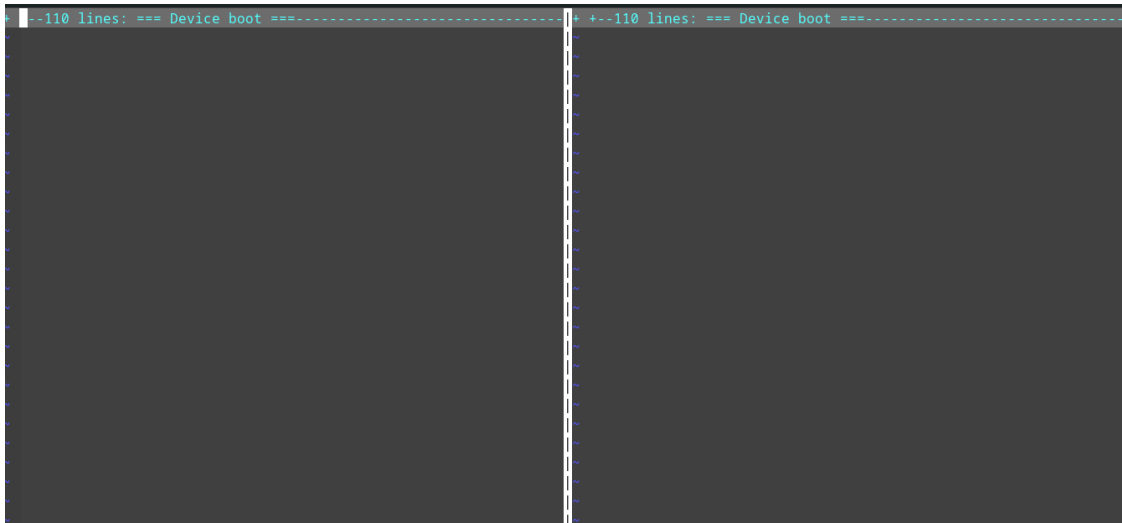


Figure 9: Generated log report file for 5th log file matched with given report file

8.8 Codebase

8.8.1 Makefile

```

1 CC = g++ --std=c++17
2 LIBS = -lboost_regex -lboost_date_time
3 CFLAGS = -g -Wall -Werror -pedantic
4
5 all:PS7
6 PS7:kronos.o
7 $(CC) $(CFLAGS) -o PS7 kronos.o $(LIBS)
8 kronos.o:kronos.cpp
9 $(CC) $(CFLAGS) -c kronos.cpp $(LIBS)
10 clean:
11 rm -f PS7 kronos.o
12 lint:
13 cpplint kronos.cpp

```

8.8.2 Kronos.cpp

```

1 /*****
2 Copyright Shriya 2022
3 Name: Shriya Thakur
4 Date: April 20th, 2022
5 Assignment: Parsing Kronos Log file

```

```

6 Time take: 18hours
7 Sources Of help- Dr Daly, Classmate- Kevin Sprague, Tutoring
8 * website links as mentioned in Readme file
9 *****/
10 #include <iostream>
11 #include <string>
12 #include <fstream>
13 #include <boost/regex.hpp>
14 #include "boost/date_time/posix_time/posix_time.hpp"
15
16 using std::endl;
17 using std::string;
18 using std::ifstream;
19 using std::ofstream;
20 using std::getline;
21 using std::cerr;
22 using std::endl;
23 using boost::regex_search;
24 using boost::smatch;
25 using boost::regex;
26 using boost::posix_time::time_from_string;
27 using boost::posix_time::ptime;
28 using boost::posix_time::time_duration;
29
30 int main(int argc, char** argv) {
31     if (argc != 2) {
32         cerr << "Incorrect number of arguments" << endl;
33     }
34     string fn(argv[1]);
35     // make rpt file from log file
36     string outfn = fn + ".rpt";
37     ifstream inputFile(fn);
38     ofstream outputFile(outfn);
39     if (!inputFile.is_open() || !outputFile.is_open()) {
40         cerr << "Couldnt open specified files" << endl;
41         return 1;
42     }
43     // boolean for if boot is in process defaults to false
44     bool isBooting = false;
45     // make regex for start of boot
46     regex beginBoot{"\\(log\\.c\\.166\\) server started"};
47     // make regex for end of boot
48     regex endBoot {"oejs\\.AbstractConnector:Started SelectChannelConnector"};
49     // make regex for a time stamp
50     regex timeRegex{"\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}"};
51     // make objects for regex matches
52     smatch s1;
53     // modify smatch object pass by reference
54     auto extract_time = [=, &s1](string x)-> string
55     { if (regex_search(x, s1, timeRegex))
56         {return s1.str(0);} else {return NULL;}};
57     // record start time
58     ptime startTime;

```



```

59 // record end time
60 ptime endTime;
61 // make string object for each line in file
62 string line;
63 time_duration duration;
64 // store line number for output
65 unsigned int lineNumber = 0;
66 while (getline(inputFile, line)) {
67     lineNumber++;
68     // use regex search to see if we start a boot cycle
69     if (regex_search(line, beginBoot)) {
70         startTime = time_from_string(extract_time(line));
71         if (isBooting) {
72             outputFile << "**** Incomplete boot **** " << endl << endl;
73         }
74         outputFile << "=== Device boot ===" << endl
75             << lineNumber << "(" << fn << "): "
76             << extract_time(line) << " Boot Start" << endl;
77         // reset value of flag
78         isBooting = true;
79     }
80     // Boot ending sequence
81     if (regex_search(line, endBoot)) {
82         endTime = time_from_string(extract_time(line));
83         duration = endTime - startTime;
84         isBooting = false;
85         outputFile << lineNumber << "(" << fn << "): " << extract_time(line)
86             << " Boot Completed" << endl << "\tBoot Time: "
87             << duration.total_milliseconds() << "ms " << endl << endl;
88     }
89 }
90 inputFile.close();
91 outputFile.close();
92 return 0;
93 }

```