Matt Serdukoff
COMP IV Sec 203: Project Portfolio
Spring 2022

Contents:

PS0: Hello World with SFML (Simple Fast Media Library):

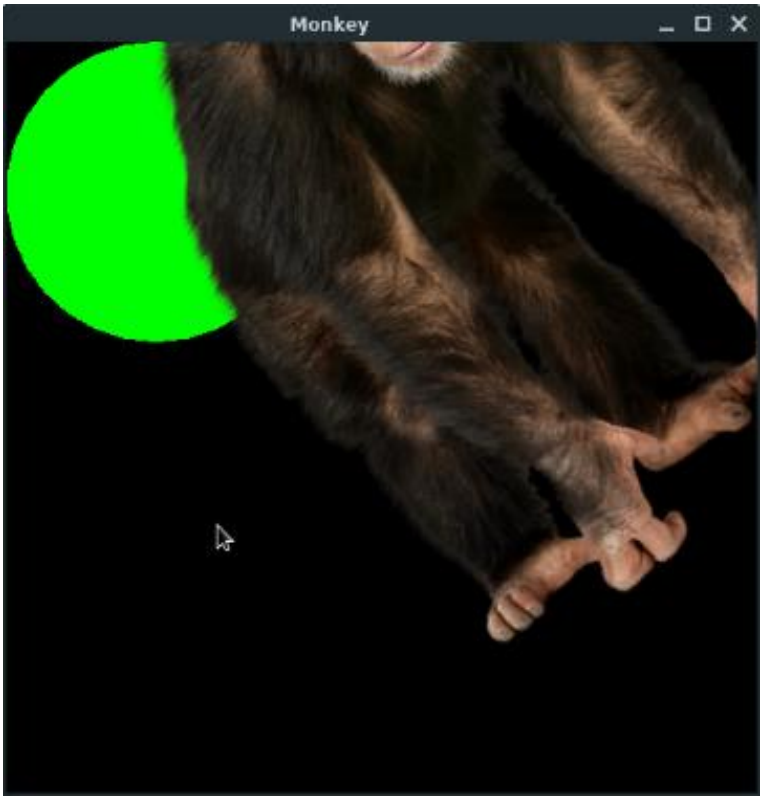
Discussion: This was the first project of the semester and it served as the first exposure to the SFML library that would later be utilized. Additionally, this project, along with PS1 and PS2, were done on a Linux virtual machine using VirtualBox. The initial project demanded to create a green circle. A later stage demanded to create a sprite from an image that would respond to keystrokes. This is what the output shows.

What I learned: Being rather simple, this project allowed me to understand how to use the sf::RenderWindow class and how to create sprites out of images or colors. Furthermore, it taught me how to structure the window loop necessary for SFML programs to display the desired material.

main.cpp

```
1. // COPYRIGHT 2022 MATT SERDUKOFF
2. #include <SFML/Graphics.hpp>
3.
4. int main() {
5.
6.      sf::RenderWindow window(sf::VideoMode(200,200), "SFML Works!");
7.      sf::CircleShape shape(100.f);
8.      shape.setFillColor(sf::Color::Green);
9.
10.     while (window.isOpen()) {
11.
12.             sf::Event event;
13.
14.             while (window.pollEvent(event)) {
15.
16.                     if (event.type == sf::Event::Closed) {
17.
18.                             window.close();
19.
20.                     }
21.
22.             }
23.
24.             window.clear();
25.             window.draw(shape);
26.             window.display();
27.
28.             }
29.
30.     return 0;
31. }
```

Output:



PS1: Linear Feedback Shift Register and Image Encoding:

Discussion: This assignment was done in two parts. The first part was to create a class called FibLFSR that manages a 16-bit Linear Feedback Shift Register algorithm with taps at 10, 12, and 13. Within the FibLFSR class, the constructor takes a seed as an argument in the format of a string. This string consists of a 16 digit long binary number. The step function simulates a step, in which the integers at the tap locations are taken and added together using the bitwise XOR operator (^). The generate function simulates a given number of steps (k).

What I learned: The main learning outcome for this project was learning how to use the Boost test library which was utilized for the first stage of this project. Additionally, I gained better practical knowledge of how member variables, methods, and constructors all work together.

Makefile:

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

PhotoMagic.cpp

```
1. #include <iostream>
2.
3. #include <string>
4. #include "FibLFSR.h"
5.
6. #include <SFML/System.hpp>
7. #include <SFML/Window.hpp>
8. #include <SFML/Graphics.hpp>
9.
10. //function declaration of transform
11. void transform(sf::Image& img, FibLFSR &pLFSR);
12.
13. int main(int argc, char* argv[]) {
14.
15.     //starting variables for cmd line
16.     std::string input_file;
17.     std::string output_file;
18.     std::string binary_pass;
19.
```

```
20.     //arguments
21.     if (argc == 4) {
22.
23.             input_file = argv[1];
24.             output_file = argv[2];
25.             binary_pass = argv[3];
26.
27.     }
28.     else {
29.
30.             std::cout << "Arguments are Invalid" << endl;
31.             return 0;
32.     }
33.
34.     //create object for original image
35.     sf::Image image_original;
36.     image_original.loadFromFile(input_file);
37.
38.     //FibLFSR object for the binary password
39.     FibLFSR bpass(binary_pass);
40.
41.     //2 windows to transform cat.jpg, size of it is 350x250
42.     sf::RenderWindow window_1(sf::VideoMode(350, 250), "Original Cat.jpg");
43.     sf::RenderWindow window_2(sf::VideoMode(350, 250), "Encrypted Cat.jpg");
44.
45.     //now time for textures and sprites
46.     sf::Texture originalTexture;
47.     originalTexture.loadFromFile(input_file);
48.
49.     sf::Sprite sprite1;
50.     sprite1.setTexture(originalTexture);
51.
52.     //transform
53.     transform(image_original, bpass);
54.
55.     //now that transform has been done, we save result to new file
56.     image_original.saveToFile(output_file);
57.
58.     //now for encoded texture and sprite
59.     sf::Texture encodedTexture;
60.     encodedTexture.loadFromFile(output_file);
61.
62.     sf::Sprite sprite2;
63.     sprite2.setTexture(encodedTexture);
64.
65.     //now stuff taken from pixels.cpp
66.     while (window_1.isOpen() && window_2.isOpen()) {
67.
68.             sf:: Event event;
69.             while (window_1.pollEvent(event)) {
70.
71.                     if (event.type == sf::Event::Closed) {
72.
73.                             window_1.close();
74.                     }
```

```
75.
76.             }
77.
78.             while (window_2.pollEvent(event)) {
79.
80.                     if (event.type == sf::Event::Closed) {
81.
82.                             window_2.close();
83.                     }
84.             }
85.
86.             window_1.clear();
87.             window_1.draw(sprite1);
88.             window_1.display();
89.
90.             window_2.clear();
91.             window_2.draw(sprite2);
92.             window_2.display();
93.
94.     }
95.
96.     return 0;
97. }
98.
99. //func definition of transform
100. void transform(sf::Image& img, FibLFSR &pLFSR) {
101.
102.     //m stands for manipulated, dont understand why these have to be Uint8, i
got it from SMFL website
103.     sf::Uint8 m_red;
104.     sf::Uint8 m_green;
105.     sf::Uint8 m_blue;
106.
107.     //height
108.     int imageH = img.getSize().y;
109.     //width
110.     int imageW = img.getSize().x;
111.
112.     sf::Uint8 s; //variable that will be used in loop
113.
114.     //replace the pixels loop
115.     for (int i = 0; i < imageW; i++) { //i = first counter
116.
117.             for (int j = 0; j < imageH; j++) { //j = second counter
118.
119.                     sf::Color px = img.getPixel(i,j); //px is pixel
120.
121.                     //red
122.                     s = pLFSR.generate(100);
123.                     m_red = px.r ^ s;
124.                     //green
125.                     s = pLFSR.generate(100);
126.                     m_green = px.g ^ s;
127.                     //blue
128.                     s = pLFSR.generate(100);
```

```
129.                     m_blue = px.b ^ s;
130.
131.                     sf::Color newColor(m_red, m_green, m_blue); //new color
from SMFL color class
132.
133.                     img.setPixel(i, j, newColor);
134.
135.             }
136.
137.     }
138.
139. }
```

FibLFSR.h

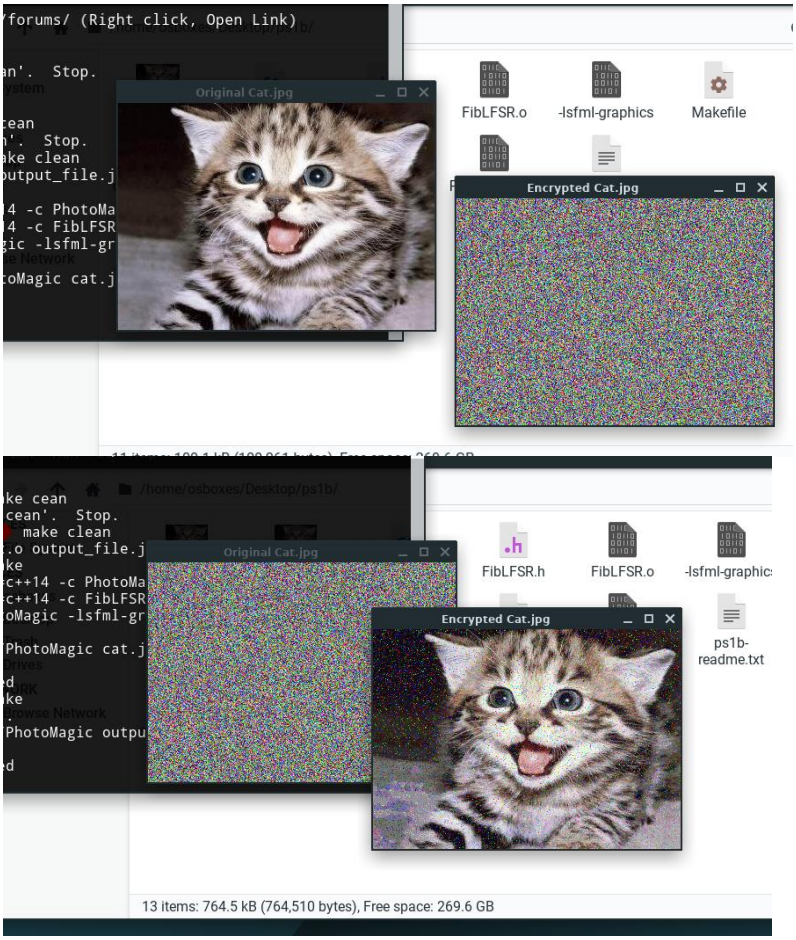
```
1. #include <iostream>
2.
3. using namespace std;
4.
5. class FibLFSR {
6. public:
7.      FibLFSR(string seed);
8.
9.      int step();
10.
11.     int generate(int k);
12.
13.     friend ostream& operator<< (ostream &out, FibLFSR &pLFSR);
14.
15.     string getSeed();
16. private:
17.     //these are changed from 10, 12, 13 since the string and binary num list
indexes in opposite directions
18.     int lead = 0;  //technically not a tap but first bit in string
19.     int tap1 = 2;
20.     int tap2 = 3;
21.     int tap3 = 5;
22.     string bits;
23. };
```

FibLFSR.cpp

```
1. #include <iostream>
2. #include <string>
3. #include "FibLFSR.h"
4.
5. using namespace std;
6.
7. //implementation for FibLFSR class
8. FibLFSR::FibLFSR(string seed) {
9.
10.     bits = seed;
11. }
12.
13. int FibLFSR::step() {
14.
15.     int lead_bit = lead;
```

```
16.     int t1 = tap1;
17.     int t2 = tap2;
18.     int t3 = tap3;
19.
20.     int str_arr[16];
21.
22.     //taking string and turning it into an array of ints
23.     for (int i = 0; i < 15 ; i++) {
24.
25.             str_arr[i] = bits[i] - 48;
26.
27.     }
28.
29.     int new_bit = str_arr[lead_bit] ^ str_arr[t1] ^ str_arr[t2] ^
str_arr[t3];
30.
31.     //feed int arr back into bits
32.     for (int i = 0; i <= 15; i++) {
33.
34.             bits[i] = str_arr[i+1] + 48;
35.
36.     }
37.
38.     str_arr[15] = new_bit;
39.
40.     return new_bit;
41. }
42.
43. int FibLFSR::generate(int k) {
44.
45.     int s = 0;
46.
47.     for (int i = 0; i < k; i++) {
48.
49.             s = (s*2) + step();
50.
51.     }
52.
53.     return s;
54. }
55.
56. //overloaded << operator like in instruction sheet
57. ostream& operator<< (ostream &out, FibLFSR &pLFSR) {
58.
59.     out << pLFSR.bits;
60.
61.     return out; //is this correct?
62. }
63.
64. string FibLFSR::getSeed() {
65.
66.     return bits;
67. }
```

Output Screenshots:



PS2: N-Body Simulation:

Disccusion: PS2 is also a project that was done in two parts. It creates a 2D animated model of the solar system with N bodies. The movement of the planets in relation to the sun is calculated using Newton's law of universal gravitation. In this assignment, the challenge consisted of organizing the program. The CelestialBody class creates bodies from a text file specifying their position, mass, velocity, and the file name from which to create the sprite of that specific planet. It contains many mutator functions that allow the program to manipulate these same parameters and member variables. The class Universe utilizes a vector of CelestialBody objects and stores them within it. Additionally, the Universe class is responsible for drawing the CelestialBody objects at each stage within their orbit.

What I learned: The learning outcome from this project was a better understanding of mutators and when they are necessary in a program, along with using a class to hold objects of another class. Hands-on experience with inter-class operations improved my grasp on the subject and object-oriented programming.

Makefile:

```
CC = g++
CFLAGS = -Wall -Werror -pedantic -std=c++14
SFML = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio

all: NBody

NBody: main.o CelestialBody.o
	$(CC) main.o CelestialBody.o -o NBody $(SFML)

main.o: main.cpp CelestialBody.h Universe.h
	$(CC) $(CFLAGS) -c main.cpp CelestialBody.h Universe.h

CelestialBody.o: CelestialBody.cpp CelestialBody.h Universe.h
	$(CC) $(CFLAGS) -c CelestialBody.cpp CelestialBody.h Universe.h

clean:
	rm main.o
	rm CelestialBody.o
	rm *.gch
	rm NBody
```

main.cpp:

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include "Universe.h"

using namespace std;

int main(int argc, char* argv[]) {

    double simulationTime;
    double stepTime;
```

```
13.     double simTime = 0; //this is for counting
14.
15.     if (argc == 3) {
16.             simulationTime = stod(argv[1]);
17.             stepTime = stod(argv[2]);
18.     }
19.
20.     double N;
21.     double R;
22.
23.     cin >> N;
24.     cin >> R;
25.
26.     Universe universe(N, R);
27.
28.     //Window
29.     sf::RenderWindow window(sf::VideoMode(600, 600), "Our Solar System");
30.     window.setFramerateLimit(60);
31.
32.     //setup background sprite
33.     sf::Image background;
34.     background.loadFromFile("nbody/starfield.jpg");
35.     sf::Texture backgroundTexture;
36.     backgroundTexture.loadFromImage(background);
37.     sf::Sprite backgroundSprite;
38.     backgroundSprite.setTexture(backgroundTexture);
39.
40.     //Music
41.     sf::Music music;
42.     music.openFromFile("nbody/2001.wav");
43.     music.play();
44.
45.     //elapsed time font
46.     sf::Font timeFont;
47.     timeFont.loadFromFile("arial.ttf");
48.
49.     //elapsed time text
50.     sf::Text timeText;
51.     timeText.setFont(timeFont);
52.     timeText.setCharacterSize(20);
53.     timeText.setFillColor(sf::Color::White);
54.
55.     //window event loop
56.     while (window.isOpen()) {
57.
58.             sf::Event event;
59.
60.             while (window.pollEvent(event)) {
61.
62.                     if (event.type == sf::Event::Closed) {
63.
64.                             window.close();
65.                     }
66.             }
67.
```

```
68.             window.clear();
69.             window.draw(backgroundSprite);
70.             timeText.setString("Elapsed Time (Years): " + to_string((simTime)
/ 60 / 60 / 24 / 365));
71.             window.draw(timeText);
72.             window.draw(universe);
73.             window.display();
74.
75.             if (simTime < simulationTime) {
76.
77.                     simTime += stepTime;
78.                     universe.step(stepTime);
79.             }
80.     }
81.
82.     return 0;
83. }
```

CelestialBody.h:

```
1. #ifndef CELESTIALBODY_H
2. #define CELESTIALBODY_H
3.
4. #include <iostream>
5. #include <string>
6. #include <vector>
7.
8. #include <SFML/System.hpp>
9. #include <SFML/Window.hpp>
10. #include <SFML/Graphics.hpp>
11. #include <SFML/Audio.hpp>
12.
13. using namespace std;
14.
15. class CelestialBody: public sf::Drawable {
16.
17. public:
18.     //constructors
19.     CelestialBody(double radius);
20.     CelestialBody(double pos_x, double pos_y, double vel_x, double vel_y,
double _mass,
21.             double radius, string file_name);
22.
23.     //overloaded operator
24.     friend istream& operator>>(istream& input, CelestialBody& body);
25.
26.     //mutators
27.     void setPos();
28.     void setRadius(double r);
29.     void setVelocity(double vX, double vY);
30.     //void setForce(double fX, double fY);
31.
32.     //helpers
33.     void updatePosition(double time);
34.     void updateVelocity(CelestialBody &pBody, double time);
```

```
35.
36.     //getters
37.     double getPosX();
38.     double getPosY();
39.
40.     double getVelX();
41.     double getVelY();
42.
43.     /*
44.     double getAccelX();
45.     double getAccelY();
46.     */
47.
48.     double getMass();
49.     double getRadius();
50.
51. private:
52.     //draw method
53.     virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
const;
54.
55.     sf::Image image;
56.     sf::Sprite sprite;
57.     sf::Texture texture;
58.
59.     //file name string
60.     string fileName;
61.
62.     //variables for celestial bodies
63.     double posX;
64.     double posY;
65.
66.     double velX;
67.     double velY;
68.
69.     double mass;
70.     double radius;
71. };
72.
73. double getForceX(CelestialBody &body1, CelestialBody &body2);
74. double getForceY(CelestialBody &body1, CelestialBody &body2);
75.
76. #endif
```

CelestialBody.cpp:

```
1. #include <iostream>
2. #include <string>
3. #include <vector>
4. #include <fstream>
5. #include <cmath>
6. #include "CelestialBody.h"
7.
8. //CelestialBody class functions
9. CelestialBody::CelestialBody(double radius) {
10.
```

```
11.     this->radius = radius;
12. }
13.
14. CelestialBody::CelestialBody(double pos_x, double pos_y, double vel_x, double
vel_y,
15.     double _mass, double _radius, string file_name) {
16.
17.     //setting data to respective variables
18.     posX = pos_x;
19.     posY = pos_y;
20.     velX = vel_x;
21.     velY = vel_y;
22.     mass = _mass;
23.     radius = _radius;
24.
25.     fileName = file_name;
26.
27.     //setting it up
28.     image.loadFromFile(fileName);
29.     texture.loadFromImage(image);
30.     sprite.setTexture(texture);
31.     sprite.setPosition(posX, posY);
32.
33. }
34.
35. void CelestialBody::setPos() {
36.
37.     double screenPosX = ((posX / radius) * (600 / 2)) + (600 / 2);
38.     double screenPosY = ((posY / radius) * (600 / 2)) + (600 / 2);
39.
40.     sprite.setPosition(screenPosX, screenPosY);
41. }
42.
43. void CelestialBody::setRadius(double r) {
44.
45.     radius = r;
46. }
47.
48. void CelestialBody::setVelocity(double vX, double vY) {
49.
50.     velX = vX;
51.     velY = vY;
52. }
53.
54. void CelestialBody::updateVelocity(CelestialBody &pBody, double time) {
55.
56.
57.     double forceX = getForceX(*this, pBody);
58.     double forceY = getForceY(*this, pBody);
59.
60.     double accelX = forceX / mass;
61.     double accelY = forceY / mass;
62.
63.     velX -= accelX * time;
64.     velY -= accelY * time;
```

```
65. }
66.
67. void CelestialBody::updatePosition(double time) {
68.
69.     posX -= getVelX() * time;
70.     posY -= getVelY() * time;
71. }
72.
73. istream& operator>>(istream& input, CelestialBody& body) { //read from file
74.
75.     input >> body.posX >> body.posY;
76.     input >> body.velX >> body.velY;
77.     input >> body.mass;
78.     input >> body.fileName;
79.
80.     body.image.loadFromFile(body.fileName);
81.     body.texture.loadFromImage(body.image);
82.     body.sprite.setTexture(body.texture);
83.
84.     body.sprite.setPosition(body.posX, body.posY);
85.
86.     return input;
87. }
88.
89.
90. void CelestialBody::draw(sf::RenderTarget& target, sf::RenderStates states)
const {
91.
92.     target.draw(sprite, states);
93. }
94.
95. double CelestialBody::getPosX() {
96.
97.     return posX;
98. }
99.
100. double CelestialBody::getPosY() {
101.
102.     return posY;
103. }
104.
105.
106. double CelestialBody::getVelX() {
107.
108.     return velX;
109. }
110.
111. double CelestialBody::getVelY() {
112.
113.     return velY;
114. }
115.
116. double CelestialBody::getMass() {
117.
118.     return mass;
```

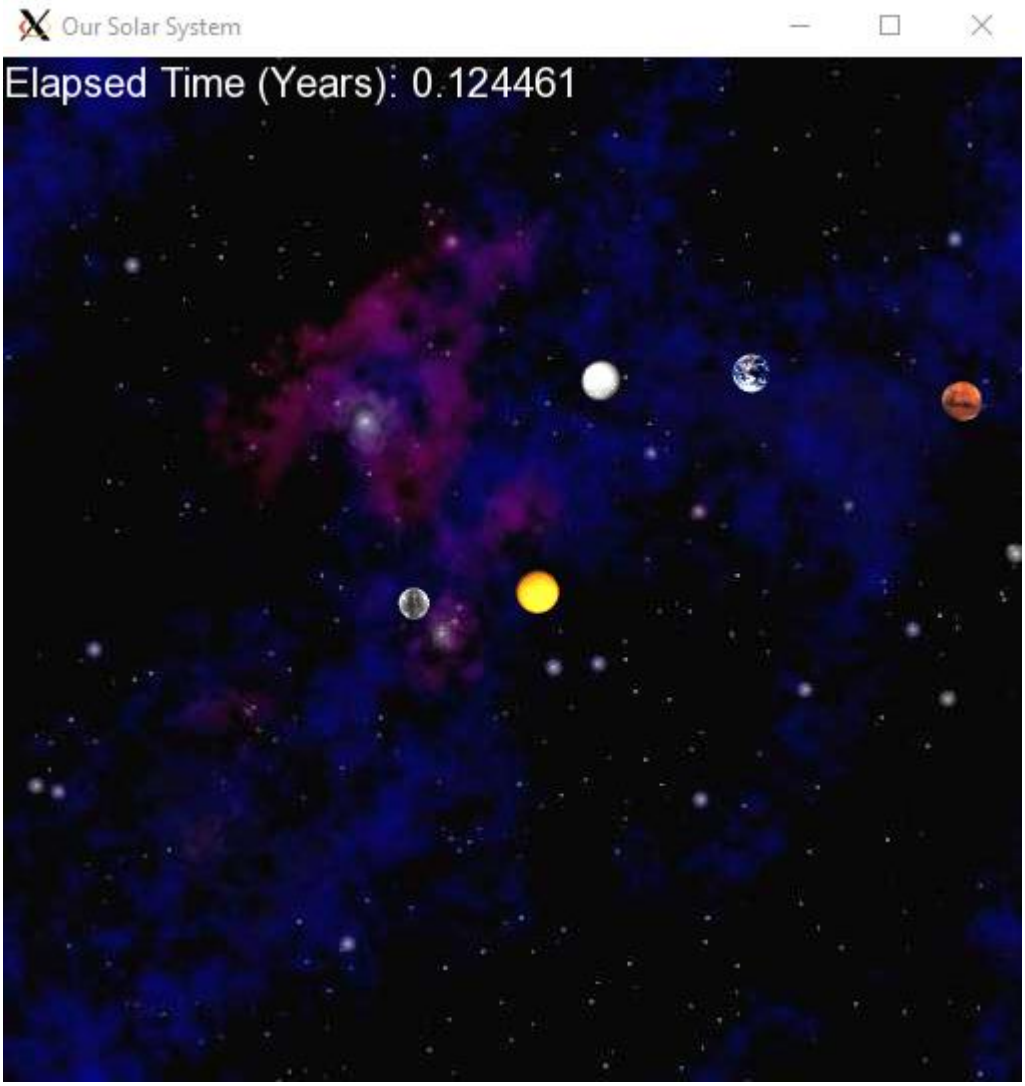
```
119. }
120.
121. double CelestialBody::getRadius() {
122.
123.    return radius;
124. }
125.
126. double getForceX(CelestialBody &body1, CelestialBody &body2) {
127.
128.    double distanceX = body2.getPosX() - body1.getPosX();
129.    double distanceY = body2.getPosY() - body1.getPosY();
130.
131.    double rSQ = pow(distanceX, 2) + pow(distanceY, 2);
132.    double r = sqrt(rSQ);
133.
134.    double force = (6.677e-11 * body1.getMass() * body2.getMass()) / rSQ;
135.
136.    double forceX_ret = force * (distanceX / r);
137.
138.    return forceX_ret;
139. }
140.
141. double getForceY(CelestialBody &body1, CelestialBody &body2) {
142.
143.    double distanceX = body2.getPosX() - body1.getPosX();
144.    double distanceY = body2.getPosY() - body1.getPosY();
145.
146.    double rSQ = pow(distanceX, 2) + pow(distanceY, 2);
147.    double r = sqrt(rSQ);
148.
149.    double force = (6.677e-11 * body1.getMass() * body2.getMass()) / rSQ;
150.
151.    double forceY_ret = force * (distanceY / r);
152.
153.    return forceY_ret;
154. }
```

Universe.h:

```
1. #ifndef UNIVERSE_H
2. #define UNIVERSE_H
3.
4. #include <iostream>
5. #include <vector>
6. #include <string>
7. #include "CelestialBody.h"
8.
9. #include <SFML/System.hpp>
10. #include <SFML/Window.hpp>
11. #include <SFML/Graphics.hpp>
12. #include <SFML/Audio.hpp>
13.
14. using namespace std;
15.
16. class Universe: public sf::Drawable {
17.
```

```
18. public:
19.     Universe(double N, double R);
20.     void step(double seconds);
21. private:
22.
23.     virtual void draw(sf::RenderTarget &target, sf::RenderStates states)
const;
24.
25.     double numPlanets;
26.     double uRadius;
27.     vector <CelestialBody*> bodiesVector;
28. };
29.
30. Universe::Universe(double N, double R) {
31.
32.     for (int i = 0; i < N; i++) {
33.
34.             CelestialBody* newBody = new CelestialBody(R);
35.             cin >> *newBody;
36.             newBody->setPos();
37.
38.             bodiesVector.push_back(newBody);
39.     }
40. }
41.
42. void Universe::draw(sf::RenderTarget &target, sf::RenderStates states) const
{
43.
44.     for (double i = 0; i < bodiesVector.size(); i++) {
45.
46.             target.draw(*bodiesVector[i], states);
47.     }
48. }
49.
50. void Universe::step(double seconds) {
51.
52.     for (auto itr = bodiesVector.begin(); itr != bodiesVector.end(); itr++) {
53.
54.             for (auto itr_2 = bodiesVector.begin(); itr_2 != bodiesVec-
tor.end(); itr_2++) {
55.
56.                     if (*itr != *itr_2) {
57.
58.                             (*itr)->updateVelocity((**itr_2), seconds);
59.                     }
60.             }
61.     }
62.
63.     for (auto itr = bodiesVector.begin(); itr != bodiesVector.end(); itr++) {
64.
65.             (*itr)->updatePosition(seconds);
66.             (*itr)->setPos();
67.     }
68. }
69. #endif
```

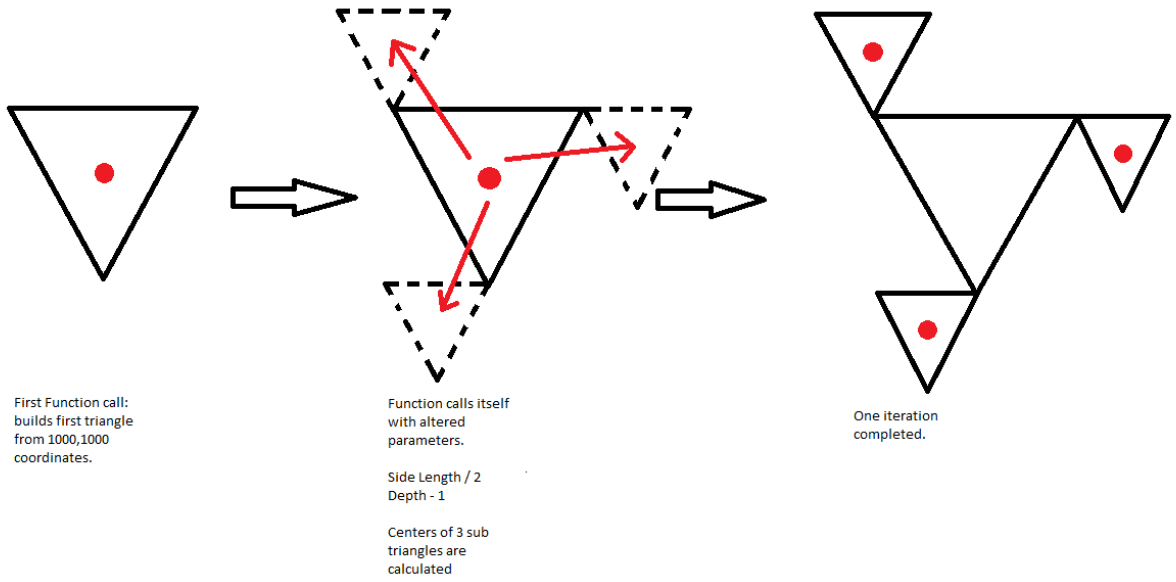
Output Screenshot:



PS3: Recursive Graphics:

Discussion: The objective of this project was to create a drawing that took inspiration from the Sierpinski triangle geometry. This was to be done using a single recursive function and in a way that the program takes the side length and depth of recursion as command lines. The function is called fTree() and operates in a rather unique manner. The function takes 4 parameters; side length, recursion depth, and the screen size using x,y (1000, 1000 in this case). In the first call of the function, it takes the two command lines along with 1000 and 1000. Taking the screen size, it calculates three points arranged in an equilateral triangle. Then, it creates an object of class Triangle with the three points and pushes the object onto the vector `shapes` that stores Triangle objects. Following this, an if statement determining whether the recursion depth is greater than 0 makes sure that the function can continue. Assuming it continues (if the initial recursion depth is more than 0), the function calls itself three times, once for each new triangle that it needs to create. In these three calls, the functions take the same four parameters, but they are adapted for the new geometry. The side length is divided by two, since the triangles are half the size. The depth is decreased by one, to ensure the program does not infinitely cycle. The x and y coordinates are calculated so that they are the center points of each of the three sub-triangles. The first iteration is now complete.

Diagram of the functionality of fTree():



What I learned: This project was a great exercise in recursion. It required a great deal of pen and paper planning to figure out the implementation, and which parameters can be recursively passed down. Additionally, I learned how to use the sf::ConvexShape class, which was essential to the drawing of the triangles.

Makefile:

```
CC = g++
CFLAGS = -Wall -Werror -pedantic -std=c++14
SFML = -lsfml-graphics -lsfml-window -lsfml-system

all: TFractal

TFractal: TFractal.o Triangle.o
	$(CC) TFractal.o Triangle.o -o TFractal $(SFML)
TFractal.o: TFractal.cpp Triangle.h
	$(CC) $(CFLAGS) -c TFractal.cpp Triangle.h
Triangle.o: Triangle.cpp Triangle.h
	$(CC) $(CFLAGS) -c Triangle.cpp Triangle.h
clean:
	rm TFractal.o
	rm Triangle.o
	rm TFractal
	rm Triangle.h.gch
```

TFractal.cpp

```
#include <iostream>
#include <string>
#include <cmath>
#include <cstdlib>
#include <SFML/Graphics.hpp>
#include "Triangle.h"

using std::cout;
using std::endl;

//vector
std::vector<std::unique_ptr<Triangle>> shapes;

// fTree declaration
void fTree(double _sideLength, int depth, double x, double y);

int main(int argc, char* argv[]) {

    double L;
    int N;

    if (argc == 3) {

        L = std::stod(argv[1]); // length
        N = atoi(argv[2]); // depth of recur
    }
    else {

        cout << "Invalid command lines." << endl;
        return 0;
    }

    // double height = (sqrt(3)/2) * L;

    cout << "STEP 1" << endl;
```

```
36.
37.     cout << "STEP 2" << endl;
38.
39.     fTree(L, N, 500, 500);
40.
41.     cout << "STEP 3" << endl;
42.
43.     sf::RenderWindow window(sf::VideoMode(1000, 1000), "Sierpinski");
44.
45.     while (window.isOpen()) {
46.
47.         sf::Event event;
48.
49.         while (window.pollEvent(event)) {
50.
51.             if (event.type == sf::Event::Closed) {
52.
53.                 window.close();
54.             }
55.         }
56.
57.         for (unsigned int i = 0; i < shapes.size(); i++) {
58.
59.             window.draw(*shapes[i]);
60.         }
61.
62.         window.display();
63.     }
64.
65.     return 0;
66. }
67.
68. void fTree(double _sideLength, int depth, double x, double y) {
69.
70.     cout << "1" << endl;
71.
72.     sf::Vector2f p1(x - (_sideLength/2), y - _sideLength/2);
73.     sf::Vector2f p2(x + (_sideLength/2), y - _sideLength/2);
74.     sf::Vector2f p3(x, y + _sideLength/2);
75.
76.     cout << "2" << endl;
77.
78.     Triangle* newShape = new Triangle(p1, p2, p3);
79.
80.     cout << "3" << endl;
81.
82.     shapes.push_back(std::make_unique<Triangle>(*newShape));
83.
84.     cout << "4" << endl;
85.
86.     if (depth > 0) {
87.
88.         cout << "6" << endl;
89.
90.         // A
```

```
91.          fTree(_sideLength/2, depth - 1,
92.              x - (_sideLength/2),
93.              y - (_sideLength/2) - (_sideLength/4));
94.
95.          cout << "7" << endl;
96.          // B
97.          fTree(_sideLength/2, depth - 1,
98.              x + (_sideLength/2) + (_sideLength/4),
99.              y - _sideLength/4);
100.
101.          cout << "8" << endl;
102.          // C
103.          fTree(_sideLength/2, depth - 1,
104.              x - _sideLength/4,
105.              y + (_sideLength/2) + (_sideLength/4));
106.      }
107. }
```

Traingle.h:

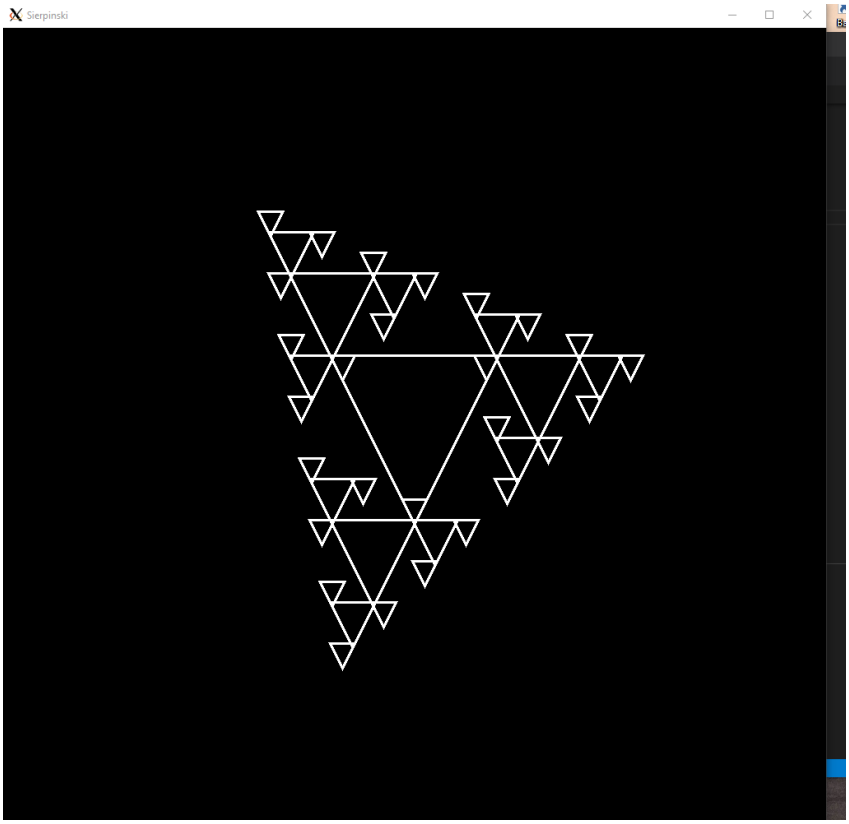
```
1. #ifndef TRIANGLE_H
2. #define TRIANGLE_H
3.
4. #include <iostream>
5. #include <cmath>
6. #include <vector>
7.
8. //SFML
9. #include <SFML/Graphics.hpp>
10.
11. //Triangle class
12. class Triangle: public sf::Drawable {
13.     public:
14.         Triangle(sf::Vector2f p1, sf::Vector2f p2, sf::Vector2f p3);
15.
16.         sf::Vector2f point1();
17.         sf::Vector2f point2();
18.         sf::Vector2f point3();
19.
20.     private:
21.         virtual void draw(sf::RenderTarget &target,
22.             sf::RenderStates states) const;
23.
24.         sf::ConvexShape triangle;
25. };
26. #endif
```

Triangle.cpp:

```
1. #include <iostream>
2. #include "Triangle.h"
3.
4. //implementation for Triangle.h
5. Triangle::Triangle(sf::Vector2f p1, sf::Vector2f p2, sf::Vector2f p3) {
6.
7.     triangle.setPointCount(3);
8.     triangle.setPoint(0, p1);
```

```
9.      triangle.setPoint(1, p2);
10.      triangle.setPoint(2, p3);
11.      triangle.setOutlineColor(sf::Color::White);
12.      triangle.setOutlineThickness(3);
13.      triangle.setFillColor(sf::Color::Transparent);
14. }
15.
16. void Triangle::draw(sf::RenderTarget &target,
17.      sf::RenderStates states) const {
18.
19.      target.draw(triangle, states);
20. }
```

Output Screenshot:



PS4: Synthesizing a Plucked String Sound:

Discussion: Creating this simple synthesizer was a two-step project. In the first half, the CircularBuffer class was created and tested with Boost. In the second part of the assigment, the CircularBuffer was combined with the StringSound class to create a guitar keyboard. A vector of 37 sf::Int16 values was used to hold audio samples that were generated by the StringSound class. Another vector was utilized that additionally held 37 values of type sf::Sound. The output gave errors in terminal, however the program still functioned, meaning that the sounds would still play upon a key being pressed.

What I learned: This project allowed to me to learn more about dynamically allocated arrays. Furthermore, I learned far more about SFML's sound capabilities as the previous projects were almost exclusively visual.

Makefile:

```
1. CC = g++
2. CFLAGS = -Wall -Werror -pedantic -std=c++14
3. SFML = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
4.
5. all: KSGuitarSim
6.
7. KSGuitarSim: KSGuitarSim.o StringSound.o CircularBuffer.o
8.     $(CC) $(CFLAGS) KSGuitarSim.o StringSound.o CircularBuffer.o -o KSGuitar-
Sim $(SFML)
9.
10. KSGuitarSim.o: KSGuitarSim.cpp StringSound.h
11.     $(CC) $(CFLAGS) -c KSGuitarSim.cpp StringSound.h
12.
13. StringSound.o: StringSound.cpp StringSound.h
14.     $(CC) $(CFLAGS) -c StringSound.cpp StringSound.h
15.
16. CircularBuffer.o: CircularBuffer.cpp CircularBuffer.h
17.     $(CC) $(CFLAGS) -c CircularBuffer.cpp CircularBuffer.h
18.
19. clean:
20.     rm *.o
21.     rm *.gch
22.     rm KSGuitarSim
```

KSGuitarSim.cpp:

```
1. // Copyright Matt Serdukoff 2022
2.
3. #include <iostream>
4. #include <string>
5. #include <vector>
6. #include <string>
7. #include <stdexcept>
8. #include <exception>
9. #include <math.h>
10. #include <limits.h>
11. #include <SFML/Audio.hpp>
12. #include <SFML/Graphics.hpp>
13. #include <SFML/System.hpp>
14. #include <SFML/Window.hpp>
```

```
15. #include "StringSound.h"
16.
17. using std::string;
18. using std::cout;
19. using std::endl;
20.
21. #define CONCERT 440.0  // A
22. #define SAMPLE_RATE 44100
23. const int KEY = 37;
24.
25. vector <sf::Int16> createSample(StringSound &sound);
26.
27. int main() {
28.     sf::RenderWindow window(sf::VideoMode(500,500), "KSGuitarSim");
29.     double hz;  // hz is frequency
30.     vector<sf::Int16> _sample;
31.     // keys for playing sounds
32.     string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
33.     vector<std::vector<sf::Int16>> samples(KEY);
34.     vector<sf::SoundBuffer> buffer(KEY);
35.     vector<sf::Sound> sounds(KEY);
36.
37.     for (int i = 0; i < (signed)keyboard.size(); i++) {
38.         // frequency calculation
39.         hz = CONCERT * pow(2, (i - 24) / 12.0);
40.         StringSound temp(hz);
41.         _sample = createSample(temp);
42.         samples[i] = createSample(temp);
43.
44.         if (i >= KEY) {
45.             return -1;
46.         }
47.         if (!buffer[i].loadFromSamples(&samples[i][0],
48.         samples[i].size(),
49.         2,
50.         SAMPLE_RATE)) {
51.             throw std::runtime_error("Failed to load sound");
52.         }
53.         sounds[i].setBuffer(buffer[i]);
54.     }
55.
56.     while (window.isOpen()) {
57.         sf::Event event;
58.         while (window.pollEvent(event)) {
59.             if (event.type == sf::Event::Closed) {
60.                 window.close();
61.             } else if (event.type == sf::Event::TextEntered) {
62.                 if (event.text.unicode < 128) {
63.                     char _key = static_cast<char>(event.text.unicode);
64.                     for (int x = 0; x < (signed)keyboard.size(); x++) {
65.                         if (keyboard[x] == _key) {
66.                             cout << "Key is " << keyboard[x] << endl;
67.                             cout << "Playing key." << endl;
68.                             sounds[x].play();
69.                             break;
```

```
                        }
                    }
                }
            }
        }
        window.clear();
        window.display();
    }
    return 0;
}

vector <sf::Int16> createSample(StringSound &sound) {
    int len = 4;
    vector<sf::Int16> samples;
    sound.pluck();
    for (int i = 0; i < SAMPLE_RATE * len; i++) {
        sound.tic();
        samples.push_back(sound.sample());
    }
    return samples;
}
```

CircularBuffer.h:

```
// Copyright 2022 Matt Serdukoff

#ifndef CIRCULAR_BUFFER_H
#define CIRCULAR_BUFFER_H

#include <iostream>
#include <vector>
#include <stdint.h>

using std::vector;

class CircularBuffer {
 public:
    CircularBuffer(size_t capacity);
    size_t size();
    bool isFull();
    bool isEmpty();
    void enqueue(int16_t x);
    int16_t dequeue();
    int16_t peek();
    void empty();
 private:
    vector <int16_t> buffer;
    int _first, _last, _capacity;
    int num_items;
};

#endif // CIRCULAR_BUFFER_H
```

CircularBuffer.cpp:

```
// Copyright 2022 Matt Serdukoff

```

```cpp
#include <iostream>
#include <vector>
#include <stdint.h>
#include "CircularBuffer.h"

CircularBuffer::CircularBuffer(size_t capacity) {
    if (1 > capacity) {
        throw std::invalid_argument("Capacity must be greater than 0");
    }
    _first = 0;
    _last = 0;
    _capacity = capacity;
    num_items = 0;
    buffer.resize(_capacity);
}

size_t CircularBuffer::size() {
    return num_items;
}

bool CircularBuffer::isFull() {
    if (num_items == _capacity) {
        return true;
    } else {
        return false;
    }
}

bool CircularBuffer::isEmpty() {
    if (num_items == 0) {
        return true;
    } else {
        return false;
    }
}

// enqueue
void CircularBuffer::enqueue(int16_t x) {
    if (isFull()) {
        throw std::runtime_error("Cannot add to full");
        return;
    } else {
        if (_last >= _capacity) {
            _last = 0;
        }
        buffer.at(_last) = x;
        _last++;
        num_items++;
    }
}

// dequeue front and return
int16_t CircularBuffer::dequeue() {
    if (isEmpty()) {
        throw std::runtime_error("Cannot dequeue from empty queue");
```

```
58.         return -1;
59.     } else {
60.         int16_t dq_front =  buffer.at(_first);  // dq_front is "dequeued
front"
61.         buffer.at(_first) = 0;
62.         _first++;
63.         num_items--;
64.         if (_first >= _capacity) {
65.             _first = 0;
66.         }
67.         return dq_front;
68.     }
69. }
70.
71. int16_t CircularBuffer::peek() {
72.     if (isEmpty()) {
73.         throw std::runtime_error("Cannot take peek from empty queue");
74.         return -1;
75.     } else {
76.         return buffer.at(_first);
77.     }
78. }
79.
80. void CircularBuffer::empty() {
81.     _first = 0;
82.     _last = 0;
83.     num_items = 0;
84. }
```

StringSound.h:

```
1. // Copyright Matt Serdukoff 2022
2.
3. #include <iostream>
4. #include <vector>
5. #include <string>
6. #include <cmath>
7. #include <random>
8. #include <SFML/Audio.hpp>
9. #include <SFML/Graphics.hpp>
10. #include <SFML/System.hpp>
11. #include <SFML/Window.hpp>
12. #include "CircularBuffer.h"
13.
14. const int SAMPLE_RATE = 44100;
15. const double DECAY = 0.996;
16.
17. class StringSound {
18.  public:
19.     explicit StringSound(double frequency);
20.     explicit StringSound(vector<sf::Int16> init);
21.     StringSound(const StringSound &obj);
22.     ~StringSound();
23.     void pluck();
24.     void tic();
25.     sf::Int16 sample();
```

```
26.     int time();
27.  private:
28.     CircularBuffer *buff;
29.     int _time;
30.     int n;
31. };
```

StringSound.cpp:

```
1. // Copyright Matt Serdukoff 2022
2.
3. #include "StringSound.h"
4.
5. using std::cout;
6. using std::endl;
7.
8. StringSound::StringSound(double frequency) {
9.     n = ceil(SAMPLE_RATE / frequency);
10.     try {
11.         buff = new CircularBuffer(n);
12.     } catch (const std::bad_alloc &e) {
13.         cout << "Failed to create CircularBuffer" << endl;
14.     }
15.     for (int i = 0; i < n; i++) {
16.         buff->enqueue((int16_t)0);
17.     }
18.     _time = 0;
19. }
20.
21. StringSound::StringSound(vector<sf::Int16> init) {
22.     n = init.size();
23.     try {
24.         buff = new CircularBuffer(n);
25.     } catch (const std::bad_alloc &e) {
26.         cout << "Failed to create CircularBuffer" << endl;
27.     }
28.     vector <sf::Int16>::iterator it;
29.     for (it = init.begin(); it < init.end(); it++) {
30.         buff->enqueue((int16_t)*it);
31.     }
32.     _time = 0;
33. }
34.
35. StringSound::~StringSound() {
36.     cout << "Deleting..." << endl;
37.     delete buff;
38. }
39.
40. void StringSound::pluck() {
41.     std::random_device r;
42.     std::mt19937 gen(r());
43.     std::uniform_int_distribution<int16_t> random(-32768, 32767);
44.     buff->empty();
45.     for (int i = 0; i < n; i++) {
46.         buff->enqueue(random(gen));
47.     }
```

```
48. }
49.
50. void StringSound::tic() {
51.     int16_t start = buff->dequeue();
52.     int16_t second = buff->peek();
53.     int16_t avg = ((start + second) / 2);
54.     int16_t karplus = avg * DECAY;
55.     buff->enqueue((sf::Int16)karplus);
56. }
57.
58. sf::Int16 StringSound::sample() {
59.     sf::Int16 smpl = ((sf::Int16)buff->peek());
60.     return smpl;
61. }
62.
63. int StringSound::time() {
64.     return _time;
65. }
```

Output:

This is the terminal output only, as audio cannot be transmitted through a PDF

```
Deleting...
Key is d
Playing the dkey.
An internal OpenAL call failed in Sound.cpp(73).
Expression:
   alSourcePlay(m_source)
Error description:
   AL_INVALID_OPERATION
   The specified operation is not allowed in the current state.
```

PS5: DNA Alignment:

Discussion: This project calculates the distance of DNA sequences and finds the best possible alignment. To accomplish this, a matrix data structure was created and manipulated to achieve the project's goals. The matrix was a private member variable of the EDistance class. The matrix was filled from right to left, row by row.

What I learned: This project taught me how to do make and work with matrices in C++. Using my experience gained from my Linear Algebra class, I was able to get a hang of it quick.

Makefile:

```
CC = g++
CFLAGS = -g -Wall -Werror -pedantic -std=c++14
SFML = -lsfml-system

all: EDistance

EDistance: main.o EDistance.o
	$(CC) $(CFLAGS) main.o EDistance.o -o EDistance $(SFML)

EDistance.o: EDistance.cpp EDistance.h
	$(CC) $(CFLAGS) -c EDistance.cpp EDistance.h

main.o: main.cpp
	$(CC) $(CFLAGS) -c main.cpp -o main.o

clean:
	rm *.o
	rm EDistance
	rm *.gch
```

main.cpp:

```
// COPYRIGHT 2022 MATT SERDUKOFF
#include <iostream>
#include <SFML/System.hpp>
#include "EDistance.h"

using std::string;
using std::cin;
using std::cout;
using std::endl;

int main() {
    string dna1, dna2;
    cin >> dna1;
    cin >> dna2;

    sf::Clock clock;
    sf::Time time;
    EDistance ED(dna1, dna2);
    int editD = ED.optDistance();
    ED.stringAlignment();  // seg fault here
    time = clock.getElapsedTime();

```

```
23.      cout << "Edit Distance = " << editD << endl;
24.      cout << "Execution time is " << time.asSeconds() << endl;
25. }
```

EDistance.h:

```
1. // COPYRIGHT 2022 MATT SERDUKOFF
2. #ifndef E_DISTANCE_H
3. #define E_DISTANCE_H
4.
5. #include <iostream>
6. #include <vector>
7. #include <string>
8. #include <memory>
9. #include <algorithm>
10.
11. using std::string;
12. using std::unique_ptr;
13.
14. class EDistance {
15.  public:
16.      EDistance(const string& strand1, const string& strand2);
17.      static int penalty(char a, char b);
18.      static int min(int a, int b, int c);
19.      int optDistance();
20.      void stringAlignment() const;
21.  private:  // I started putting _ in front of all variables within a class
22.      string _DNA1;
23.      string _DNA2;
24.      int _rows;
25.      int _columns;
26.      unique_ptr<int[]> _matrix;
27. };
28.
29. #endif  // E_DISTANCE_H
```

EDistance.cpp:

```
1. // COPYRIGHT 2022 MATT SERDUKOFF
2.
3. #include <cmath>
4. #include <cstdlib>
5. #include "EDistance.h"
6.
7. using std::cout;
8. using std::endl;
9. using std::make_unique;
10.
11. EDistance::EDistance(const string& strand1, const string& strand2) {
12.      _DNA1 = strand1;
13.      _DNA2 = strand2;
14.      _rows = _DNA1.size() + 1;
15.      _columns = _DNA2.size() + 1;
16.      // create matrix and fill with 0s
17.      _matrix = make_unique<int[]>(_rows * _columns);
18.      for (int i = 0; i < (_rows * _columns); i++) {
19.          _matrix[i] = 0;
```

```
20.      }
21.      int num = 0;
22.      int i = _rows * _columns - 1;
23.      while (i >= (_rows - 1) * _columns) {
24.          _matrix[i] = num;
25.          num += 2;
26.          i--;
27.      }
28.      num = 0;
29.      i = _rows * _columns - 1;
30.      while (i >= _columns - 1) {
31.          _matrix[i] = num;
32.          num += 2;
33.          i-= _columns;
34.      }
35. }
36.
37. int EDistance::penalty(char a, char b) {
38.      if (a == b) { return 0; }
39.      return 1;
40. }
41.
42. int EDistance::min(int a, int b, int c) {
43.      int min = a;
44.      if (b < a) { min = b; }
45.      if (c < min) { min = c; }
46.      return min;
47. }
48.
49. int EDistance::optDistance() {
50.      auto min = [](int a, int b, int c)->int {
51.          if (a < b) {
52.              return a < c ? a : c;
53.          }
54.          return b < c ? b : c;
55.      };
56.      int i = (_rows - 1) * _columns - 2;
57.      int element1 = _rows - 2;
58.      int element2 = _columns - 2;
59.      while (i >= 0) {
60.          int n1 = _matrix[i + _columns + 1]
61.              + penalty(_DNA1[element1], _DNA2[element2]);
62.          int n2 = _matrix[i + _columns] + 2;
63.          int n3 = _matrix[i + 1] + 2;
64.          _matrix[i] = min(n1, n2, n3);
65.          if ((i % _columns) != 0) {
66.              element2--;
67.              i--;
68.          } else {
69.              element2 = _columns - 2;
70.              element1--;
71.              i -= 2;
72.          }
73.      }
74.      return _matrix[0];
```

```
75. }
76.
77. void EDistance::stringAlignment() const {
78.     int i = 0;
79.     int element1, element2 = 0;
80.     while (i < _rows * _columns - 1) {
81.         if (_DNA1[element1] == _DNA2[element2]
82.         && _matrix[i] == _matrix[i + 1 + _columns]) {
83.             cout << _DNA1[element1] << " "
84.             << _DNA2[element2] << 0 << endl;
85.             i += _columns + 1;
86.             element1++;
87.             element2++;
88.         } else if (_matrix[i] == _matrix[i + 1 + _columns] + 1){
89.             cout << _DNA1[element1] << " "
90.             << _DNA2[element2] << 1 << endl;
91.             i += _columns + 1;
92.             element1++;
93.             element2++;
94.         } else if (_matrix[i] == _matrix[i + _columns] + 2) {
95.             cout << _DNA1[element1] << " - " << 2 << endl;
96.             element1++;
97.         } else if (_matrix[i] == _matrix[i + 1] + 2) {
98.             cout << "- " << _DNA2[element2] << " " << 2 << endl;
99.             element2++;
100.             i++;
101.         }
102.     }
103. }
```

Output Screenshot:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

root@DESKTOP-KGR87NN:~/COMP4/psxb# make
g++ -g -Wall -Werror -pedantic -std=c++14 -c main.cpp -o main.o
g++ -g -Wall -Werror -pedantic -std=c++14 -c EDistance.cpp EDistance.h
g++ -g -Wall -Werror -pedantic -std=c++14 main.o EDistance.o -o EDistance -lsfml-system
root@DESKTOP-KGR87NN:~/COMP4/psxb# ./EDistance < fli8.txt
T T 0
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
Edit Distance: 6
Execution Time:6.2e-05
root@DESKTOP-KGR87NN:~/COMP4/psxb#
```

PS6: Random Writer:

Discussion: The objective of this project was to use kgrams (usually referred to as ngrams) of a various size to create a "random text". The takes two command lines, one that specifies the size of the kgram, and another for how many characters of text to read. The program then reads all character sequences (kgrams) of the specified size from the .txt file that the user also specifies. It counts how frequently they appear in the given text, then outputs random text that mimics the same probability of the discovered character sequences. The output screenshot shows the program running with size 3 kgrams through 1000 characters from a file called tomsawyer.txt. Both the input and output are compared. To create this project, maps were used in a <string, int> format. The string component held the respective kgram, while the second held the frequency. The test.cpp file shows the Boost testing for the various functions in the RandWriter class.

What I learned: Having little experience with maps, much less iterating through them. This project allowed to expand my skills with this new data structure. Once I learned how to use the ::iterator with maps and how to access the first and second elements, the project became rather straight forward.

Makefile:

```
1. CC = g++
2. CFLAGS = -g -Wall -Werror -std=c++14 -pedantic
3. SFML = -lsfml-graphics -lsfml-window -lsfml-system
4. BOOST = -lboost_unit_test_framework
5.
6. all: TextWriter test
7.
8. TextWriter: TextWriter.o RandWriter.o
9.     $(CC) TextWriter.o RandWriter.o -o TextWriter
10.
11. test: test.o RandWriter.o
12.     $(CC) $(CFLAGS) test.o RandWriter.o -o test $(BOOST)
13.
14. TextWriter.o: TextWriter.cpp RandWriter.h
15.     $(CC) $(CFLAGS) -c TextWriter.cpp RandWriter.h
16.
17. RandWriter.o: RandWriter.cpp RandWriter.h
18.     $(CC) $(CFLAGS) -c RandWriter.cpp RandWriter.h
19.
20. test.o: test.cpp
21.     $(CC) $(CFLAGS) -c test.cpp $(BOOST)
22.
23. clean:
24.     rm *.o
25.     rm TextWriter
26.     rm *.gch
27.     rm test
```

TextWriter.cpp:

```
1. // COPYRIGHT 2022 MATT SERDUKOFF
2.
3. #include <string>
4. #include "RandWriter.h"
5.
6. using std::cout;
```

```
using std::endl;
using std::stoi;
using std::cin;

int main(int argc, const char* argv[]) {
    if (argc != 3) {
        cout << "Incorrect number of commands" << endl;
    }
    string k_str(argv[1]);
    string l_str(argv[2]);
    int k = stoi(k_str);
    int L = stoi(l_str);
    string input = "";
    string c_txt = "";
    while (cin >> c_txt) {
        input += " " + c_txt;
        c_txt = "";
    }
    cout << "INPUT IS: " << endl;
    for (int i = 0; i < L; i++) {
        cout << input[i];
    }
    cout << "\n---------- OUTPUT BELOW ------------" << endl;
    string output = "";
    RandWriter obj(input, k);
    output += obj.generate(input.substr(0, k), L);
    for (int i = 0; i < L; i++) {
        cout << output[i];
    }
    cout << endl;
    return 0;
}
```

RandWriter.h:

```
// COPYRIGHT 2022 MATT SERDUKOFF
#ifndef RAND_WRITER_H
#define RAND_WRITER_H

#include <algorithm>
#include <iostream>
#include <map>
#include <string>
#include <stdexcept>

using std::string;
using std::ostream;
using std::map;

class RandWriter {
 public:
    RandWriter(string text, int k);
    int orderK() const;
    int freq(string kgram);
    int freq(string kgram, char c);
    char kRand(string kgram);
```

```
22.     string generate(string kgram, int L);
23.     friend ostream& operator<<(ostream &ouput, RandWriter &r);
24.  private:
25.     int _order;
26.     map <string, int> _kgram;
27.     string _lex;  // lexicon
28. };
29.
30. #endif  // RAND_WRITER_H
```

RandWriter.cpp:

```
1. // COPYRIGHT 2022 MATT SERDUKOFF
2. #include <vector>
3. #include <utility>
4. #include <map>
5. #include <string>
6. #include <stdexcept>
7. #include <algorithm>
8. #include "RandWriter.h"
9.
10. using std::sort;
11. using std::pair;
12. using std::map;
13. using std::runtime_error;
14. using std::cout;
15. using std::endl;
16.
17. RandWriter::RandWriter(string text, int k) {
18.     _order = k;
19.     srand((int)time(NULL));
20.     string txt = text;
21.     for (int i = 0; i < _order; i++) {
22.         txt.push_back(text[i]);
23.     }
24.     int length = text.length();
25.     char temp;
26.     bool in_lex;
27.     for (int i = 0; i < length; i++) {
28.         temp = text.at(i);
29.         in_lex = false;
30.         for (unsigned int j = 0; j < _lex.length(); j++) {
31.             if (_lex.at(j) == temp) {
32.                 in_lex = true;
33.             }
34.         }
35.         if (!in_lex) {
36.             _lex.push_back(temp);
37.         }
38.     }
39.     sort(_lex.begin(), _lex.end());
40.     string temp_str;
41.     int x, y;
42.     for (x = _order; x <= _order + 1; x++) {
43.         for (y = 0; y < length; y++) {
44.             temp_str.clear();
```

```
45.             temp_str = txt.substr(y, x);
46.             _kgram.insert(pair<string, int>(temp_str, 0));
47.         }
48.     }
49.     map <string, int>::iterator itr;
50.     int count = 0;
51.     for (x = _order; x <= _order + 1; x++) {
52.         for (y = 0; y < length; y++) {
53.             temp_str.clear();
54.             temp_str = txt.substr(y, x);
55.             itr = _kgram.find(temp_str);
56.             count = itr->second;
57.             count++;
58.             _kgram[temp_str] = count;
59.         }
60.     }
61. }
62.
63. int RandWriter::orderK() const {
64.     return _order;
65. }
66.
67. int RandWriter::freq(string kgram) {
68.     if (kgram.length() != (unsigned)_order) {
69.         throw runtime_error("Error: kgram is not of length k");
70.     }
71.     map <string, int>::iterator itr;
72.     itr = _kgram.find(kgram);
73.     if (itr == _kgram.end()) {
74.         return 0;
75.     }
76.     return itr->second;
77. }
78.
79. int RandWriter::freq(string kgram, char c)  {
80.     if (kgram.length() != (unsigned)_order) {
81.         throw runtime_error("Error: kgram is not of at least length k");
82.     }
83.     map <string, int>::iterator itr;
84.     kgram.push_back(c);
85.     itr = _kgram.find(kgram);
86.     if (itr == _kgram.end()) {
87.         return 0;
88.     }
89.     return itr->second;
90. }
91.
92. char RandWriter::kRand(string kgram) {
93.     if (kgram.length() != (unsigned)_order) {
94.         throw runtime_error("Error: length k needed RandWriter::kRand");
95.     }
96.     map<string, int>::iterator itr;
97.     itr = _kgram.find(kgram);
98.     if (itr == _kgram.end()) {
99.         throw runtime_error("Error: cannot find kgram RandWriter::kRand");
```

```
    }
    int kgram_frq = freq(kgram);
    int randVal = rand() % kgram_frq;
    double test_frq = 0;
    double rand = static_cast<double>(randVal) / kgram_frq;
    double lvalues = 0;
    for (unsigned int i = 0; i < _lex.length(); i++) {
        test_frq =  static_cast<double>(freq(kgram, _lex[i])) / kgram_frq;
        if (rand < test_frq + lvalues && test_frq != 0) {
            return _lex[i];
        }
    lvalues += test_frq;
    }
    return ' ';
}

string RandWriter::generate(string kgram, int L) {
    if (kgram.length() != (unsigned)_order) {
        throw runtime_error("Error - kgram not of length k. (gen)");
    }
    string final_str = "";
    char return_char;
    final_str += kgram;
    for (unsigned int i = 0; i < (L - (unsigned)_order); i++) {
        return_char = kRand(final_str.substr(i, _order));
        final_str.push_back(return_char);
    }
    return final_str;
}

ostream& operator<<(ostream &output, RandWriter &r) {
    output << endl << "Order: " << r._order  << endl;
    output << "Alphabet: " << r._lex << endl;
    output << "KGRAM MAP: " << endl;
    map <string, int>::iterator itr;
    for (itr = r._kgram.begin(); itr != r._kgram.end(); itr++) {
        output << itr->first << " " << itr->second << endl;
    }
    return output;
}
```

test.cpp:

```
// COPYRIGHT 2022 MATT SERDUKOFF
// COPYRIGHT 2022 MATT SERDUKOFF
#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>

#include "RandWriter.h"

#define BOOST_TEST_DYN_LINK
 #define BOOST_TEST_MODULE Main
 #include <boost/test/unit_test.hpp>

```

```
13. using std::runtime_error;
14. using std::cout;
15. using std::endl;
16.
17. BOOST_AUTO_TEST_CASE(test1) {
18.
19.     // create obj with gagggagaggcgagaaa
20.     RandWriter testobj("gagggagaggcgagaaa", 1);
21.     // test orderK()
22.     BOOST_REQUIRE(testobj.orderK() == 1);
23.     // test freq with false inputs
24.     BOOST_REQUIRE_THROW(testobj.freq(""), runtime_error);
25.     BOOST_REQUIRE_THROW(testobj.freq("xx"), runtime_error);
26.     // test freq with real chars
27.     BOOST_REQUIRE(testobj.freq("a") == 7);
28.     BOOST_REQUIRE(testobj.freq("g") == 9);
29.     BOOST_REQUIRE(testobj.freq("c") == 1);
30.     // test freq 2
31.     BOOST_REQUIRE(testobj.freq("a", 'g') == 5);
32.     BOOST_REQUIRE(testobj.freq("c", 'a') == 0);
33.     BOOST_REQUIRE(testobj.freq("c", 'a') == 0);
34.     BOOST_REQUIRE(testobj.freq("c", 'g') == 1);
35.     BOOST_REQUIRE(testobj.freq("a", 'a') == 2);
36.     BOOST_REQUIRE(testobj.freq("g", 'a') == 5);
37.     BOOST_REQUIRE(testobj.freq("g", 'c') == 1);
38.     BOOST_REQUIRE(testobj.freq("g", 'g') == 3);
39.     // kRand testing
40.     BOOST_REQUIRE_NO_THROW(testobj.kRand("a"));
41.     BOOST_REQUIRE_NO_THROW(testobj.kRand("c"));
42.     BOOST_REQUIRE_NO_THROW(testobj.kRand("g"));
43.     BOOST_REQUIRE_THROW(testobj.kRand("z"), runtime_error);
44.     BOOST_REQUIRE_THROW(testobj.kRand("xyz"), runtime_error);
45. }
46.
47. BOOST_AUTO_TEST_CASE(test2) {
48.     // create obj with gagggagaggcgagaaa
49.     RandWriter testobj("gagggagaggcgagaaa", 0);
50.     // test orderK()
51.     BOOST_REQUIRE(testobj.orderK() == 0);
52.     // test freq
53.     BOOST_REQUIRE(testobj.freq("", 'g') == 9);
54.     BOOST_REQUIRE(testobj.freq("", 'a') == 7);
55.     BOOST_REQUIRE(testobj.freq("", 'c') == 1);
56.     BOOST_REQUIRE(testobj.freq("", 'x') == 0);
57. }
58.
59. BOOST_AUTO_TEST_CASE(test3) {
60.     // create obj with gagggagaggcgagaaa
61.     RandWriter testobj("gagggagaggcgagaaa", 3);
62.     // test kRand()
63.     BOOST_REQUIRE_NO_THROW(testobj.kRand("cga"));
64.     BOOST_REQUIRE_NO_THROW(testobj.kRand("gag"));
65.     BOOST_REQUIRE_NO_THROW(testobj.kRand("gaa"));
66.     BOOST_REQUIRE_NO_THROW(testobj.kRand("ggc"));
67.     BOOST_REQUIRE_THROW(testobj.kRand("gaaaa"), runtime_error);
```

```
68.
69.     BOOST_REQUIRE_NO_THROW(testobj.generate("gag", 4));
70. }
```

Output Screenshot:

```
root@DESKTOP-KGR87NN:~/COMP4/ps6# ./TextWriter 3 1000 < tomsawyer.txt
INPUT IS:
 THE ADVENTURES OF TOM SAWYER BY MARK TWAIN (Samuel Langhorne Clemens) PREFACE Most of the ad
ventures recorded in this book really occurred; one or two were experiences of my own, the re
st those of boys who were schoolmates of mine. Huck Finn is drawn from life; Tom Sawyer also,
 but not from an individual-he is a combination of the characteristics of three boys whom I k
new, and therefore belongs to the composite order of architecture. The odd superstitions touc
hed upon were all prevalent among children and slaves in the West at the period of this story
-that is to say, thirty or forty years ago. Although my book is intended mainly for the enter
tainment of boys and girls, I hope it will not be shunned by men and women on that account, f
or part of my plan has been to try to pleasantly remind adults of what they once were themsel
ves, and of how they felt and thought and talked, and what queer enterprises they sometimes e
ngaged in. THE AUTHOR. HARTFORD, 1876. CHAPTER I "TOM!" No answer. "TO
---------- OUTPUT BELOW ------------
 THAT had now you'd into stant all on you done of shuck." "Answeake of dow of three at the re
light I'd had have, secrealt he float, an on their lood a said: "Becky That?" And sider to al
ittle ain." "If his this below raised be did sore they said: "Say, anything had again. The of
 though. He thours bet was out of clumed in purpriskly, sount's wardificule drope to a clothe
re dier inquick yourses tic's Tom all compresenturder aller's supped to ben I've reats away t
he ward them, ash, no, and me. "What led, wild, not to loor I'll talks-" "Tom?" "No, an exces
-but I costs told hight I know it to were won't the gethin the inted, and andor of coung agai
n. Tom treture inted house have your me been her ared, ope log, and clief; for Sunded in judg
e these the her aftered all two down a men's eyes so masts beins, and would rely ress Wellowe
dding to play with instalk preceivength the the Widown for belight whis to tured in voide are
 coop, but the We need of it was sprisonst of could by-aimself a bothe
root@DESKTOP-KGR87NN:~/COMP4/ps6#
```

PS7: Kronos Time Parsing:

Discussion: This project used the Boost Regex library to evaluate log files of a Kronos time clock. Using regular expressions, the program analyzes log files and creates its own log file in a .rpt format to output its results. The program reads each startup, completeness, time spent, and whether the clock failed. The output section for this project shows an excerpt of one of the input files, and the full output file corresponding to it.

What I learned: This project introduced me to regular expressions. Using the regex library for this project proved to be far easier than it would be to try to accomplish the exact same function without it. I did not fully understand their usefulness until it came to code this project. In the future, I will most definitely be using regular expressions if the condition is appropriate.

Makefile:

```
CC = g++
CFLAGS = -g -Wall -ansi -pedantic -Werror -ansi -std=c++14
BOOST = -lboost_regex -lboost_date_time -lboost_unit_test_framework

all: kronos

kronos: kronos.o
	$(CC) kronos.o $(CFLAGS) $(BOOST) -o kronos

kronos.o: kronos.cpp
	$(CC) -c kronos.cpp $(CFLAGS) -o kronos.o

clean:
	rm *.o
	rm *.rpt
	rm kronos
```

kronos.cpp:

```
// COPYRIGHT 2022 MATT SERDUKOFF
#include <iostream>
#include <string>
#include <fstream>
#include <exception>

#include <boost/regex.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>

using std::cout;
using std::cin;
using std::endl;
using std::string;
using std::fstream;
using std::ofstream;
using std::runtime_error;

using boost::gregorian::date;
using boost::gregorian::from_simple_string;
using boost::gregorian::date_period;
```

```
22. using boost::gregorian::date_duration;
23.
24. using boost::posix_time::ptime;
25. using boost::posix_time::time_duration;
26.
27. // main
28. int main(int argc, char* argv[]) {
29.     if (argc != 2) {
30.         throw runtime_error("Incorrect number of command lines.");
31.     } else {
32.         fstream logFile(argv[1]);
33.         string fileName = argv[1];
34.         ofstream _rpt(fileName + ".rpt", ofstream::out);
35.         string line1;
36.         date Date;
37.         ptime startTime;
38.         bool isBoot;
39.         int lineNum;
40.         boost::regex e_begin_boot(
41.             "([0-9]+)-([0-9]+)-([0-9]+) "
42.                    "([0-9]+):([0-9]+):([0-9]+): "
43.                    "\\(log.c.166\\) server started.*"
44.         );
45.         boost::regex e_end_boot(
46.             "([0-9]+)-([0-9]+)-([0-9]+) "
47.                    "([0-9]+):([0-9]+):([0-9]+).([0-9]+):INFO:"
48.                    "oejs.AbstractConnector:Started SelectChannelCon-
nector@0.0.0.0:9080.*"
49.         );
50.         if (logFile.is_open()) {
51.             while (getline(logFile, line1)) {
52.                 boost::smatch match;
53.                 if (regex_match(line1, match, e_begin_boot)) {
54.                     date temp(stoi(match[1]), stoi(match[2]),
stoi(match[3]));
55.                     Date = temp;
56.                     ptime tempStart(
57.                         Date,
58.                         time_duration(
59.                             stoi(match[4]),
60.                             stoi(match[5]),
61.                             stoi(match[6]))
62.                     );
63.                     startTime = tempStart;
64.                     if (isBoot) {
65.                         _rpt << "-----BOOT FAILED-----" << endl;
66.                         isBoot = false;
67.                     }
68.                     _rpt << "-----BOOTNG DEVICE-----" << endl
69.                          << lineNum << "(" << argv[1] << "): "
70.                          << match[1] << " " << match[2]
71.                          << " " << match[3] << " " << match[4]
72.                          << " " << match[5] << " " << match[6] << endl;
73.                     isBoot = true;
74.                 } else if (regex_match(line1, match, e_end_boot)) {
```

```
                    date temp_datefinish(
                        stoi(match[1]),
                        stoi(match[2]),
                        stoi(match[3]));
                    ptime time_end(
                        temp_datefinish,
                        time_duration(stoi(match[4]),
                            stoi(match[5]),
                            stoi(match[6]))
                    );
                    _rpt << lineNum << "(" << argv[1] << "): "
                        << match[1] << "-" << match[2] << "-" << match[3]
                        << " " << match[4] << ":" << match[5] << ":"
                        << match[6] << " \nBoot Completed" << endl;
                    time_duration _td = time_end - startTime;
                    int bootTime = _td.total_milliseconds();
                    _rpt << "Boot Time: "
                         << bootTime << "ms\n" << endl;
                    isBoot = false;
                }
                lineNum++;
            }
            _rpt.close();
            logFile.close();
        } else {
            throw runtime_error("Error: file cannot be opened.");
        }
    }
    return 0;
}
```

Input/Output file contents using `device1_intouch.log`:

`device1_intouch.log`:

this only shows part of the file, due its length the full file cannot be displayed

```
Jan 29 04:40:07 127.0.0.1 user:[pool-4-thread-1] INTOUCH;0;DEBUG;BIOMET-
RICSERVICE;Deleting image files from /usr/local/ngd/tmpimg older than 5 minutes.
Jan 29 04:40:27 127.0.0.1 user:[pool-15-thread-1] INTOUCH;0;DEBUG;BIOMET-
RICSERVICE;------------------------------Canceling Identification called---------
---------------------
Jan 29 04:40:27 127.0.0.1 user:[pool-15-thread-1] INTOUCH;0;DEBUG;BIOMET-
RICSERVICE;cancelIdentify(): Not in identification mode
Jan 29 04:40:27 127.0.0.1 user:[pool-15-thread-1] INTOUCH;0;DEBUG;DEVICEI-
OSERVICE;---->BIOMETRICSERVICE getDeviceReport()
Jan 29 04:40:27 127.0.0.1 user:[pool-15-thread-1] INTOUCH;0;DEBUG;DEVICEI-
OSERVICE;---->BIOMETRICSERVICE: Executing command :4
Jan 29 04:40:27 (none) java[1976]: INTOUCH;0;DEBUG;BIOMETRICSDK; @@@@@@@@   BII
Call -> Status @@@@@@@@ : 0
Jan 29 04:40:27 (none) java[1976]: INTOUCH;0;DEBUG;BIOMETRICSDK; ##### Biomet-
ricService_getFirmwareVersion ##### : 0
Jan 29 04:40:27 (none) java[1976]: INTOUCH;0;DEBUG;BIOMETRICSDK; @@@@@@@@   BII
Call -> Version @@@@@@@@ : 0
```

```
Jan 29 04:40:27 (none) java[1976]: INTOUCH;0;DEBUG;BIOMETRICSDK; @@@@@@@@   BII
Call -> GetNumTemplates @@@@@@@@ : 0
Jan 29 04:40:27 (none) java[1976]: INTOUCH;0;DEBUG;BIOMETRICSDK; @@@@@@@@   BII
Call -> GetMaxNTemplates @@@@@@@@ : 0
Jan 29 04:40:27 (none) java[1976]: INTOUCH;0;DEBUG;BIOMETRICSDK; @@@@@@@@   BII
Call -> GetFingerDetectTImeout @@@@@@@@ : 0
```

device1_intouch.log.rpt: (output file created)

```
-----BOOTNG DEVICE-----
435367(device1_intouch.log): 2014 03 25 19 11 59
435757(device1_intouch.log): 2014-03-25 19:15:02
Boot Completed
Boot Time: 183000ms

-----BOOTNG DEVICE-----
436498(device1_intouch.log): 2014 03 25 19 29 59
436857(device1_intouch.log): 2014-03-25 19:32:44
Boot Completed
Boot Time: 165000ms

-----BOOTNG DEVICE-----
440717(device1_intouch.log): 2014 03 25 22 01 46
440789(device1_intouch.log): 2014-03-25 22:04:27
Boot Completed
Boot Time: 161000ms

-----BOOTNG DEVICE-----
440864(device1_intouch.log): 2014 03 26 12 47 42
441214(device1_intouch.log): 2014-03-26 12:50:29
Boot Completed
Boot Time: 167000ms

-----BOOTNG DEVICE-----
442092(device1_intouch.log): 2014 03 26 20 41 34
442430(device1_intouch.log): 2014-03-26 20:44:13
Boot Completed
Boot Time: 159000ms

-----BOOTNG DEVICE-----
443071(device1_intouch.log): 2014 03 27 14 09 01
443409(device1_intouch.log): 2014-03-27 14:11:42
Boot Completed
Boot Time: 161000ms
```