Wesley Gallo

Computing 4: Projects Portfolio

Fall 2021

Contents:

PS0: Hello World with SFML

Assignment Discussion:

PS0 was the first and introductory assignment to the computing 4 course, it was used to get used to the basic features of the SFML library (Simple Fast Multimedia Library). These features were uploading an image as a sprite, using the keyboard to control the sprite, drawing, importing images, as well as show other possibilities for SFML.

Implementing PS0 was uploading a random image to use as “sprite”, and then having the program respond to keystrokes. The sprite would respond to the up, down, right, and left arrows, as well as the W, A, S, D keys, which could rotate the image.

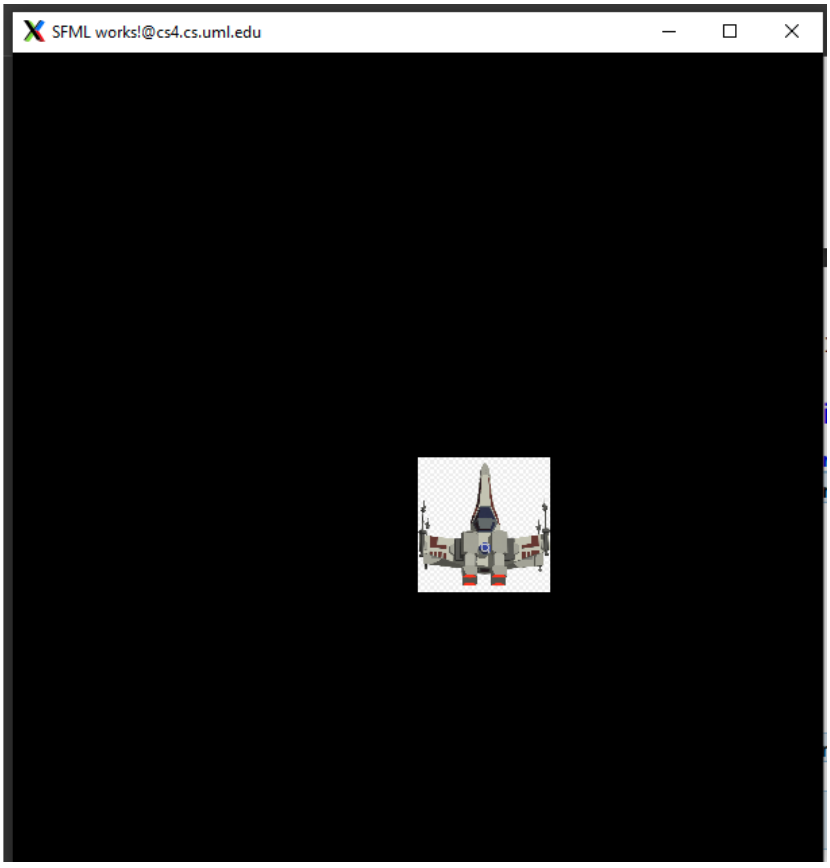
Key Algorithms, data structures, or OO designs that were central to the assignment:

The way the PS0 project worked, was by rending a loop in the SFML window object. The loop would keep running as the window was open, and then each keystroke would be sent to the even handler, and update the frame. The Event object is used to update the sprite and it is redrawn and repeated to “move” the sprite.

What I learned:

The assignment did an excellent job at introducing the SFML library, and some of its capabilities for the course. The information for PS0 was crucial in implementation of many of the future projects, by using the keyboard as an input or to display an object.

Screenshot(s) of Output:



Project Source Code:

main.cpp:

```
1.
2. /**
3. * Main.cpp - PS0
4. * Date: 9/15/2021
5. * created by: Wesley Gallo
6. */
7. #include <SFML/Graphics.hpp>
8.
9. int main()
10.      {
11.          sf::RenderWindow window(sf::VideoMode(600, 600), "SFML works!");
12.
13.          sf::Texture texture;
14.          if (!texture.loadFromFile("Sprite.png"))
15.              return EXIT_FAILURE;
16.          sf::Sprite Sprite;
17.          Sprite.setTexture(texture);
18.          Sprite.setPosition(sf::Vector2f(300, 300));
19.          while (window.isOpen())
20.          {
21.              sf::Event event;
22.              while (window.pollEvent(event))
23.              {
24.                  if (event.type == sf::Event::Closed)
25.                      window.close();
26.              }
27.              if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
28.              {
29.                  Sprite.move(1.f, 0.f);
30.              }
31.              if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
32.              {
33.                  Sprite.move(-1.f, 0.f);
34.              }
35.              else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right))
36.              {
37.                  Sprite.move(1.f, 0.f);
38.              }
39.              else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up))
40.              {
41.                  Sprite.move(0.f, -1.f);
42.              }
43.              else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down))
44.              {
45.                  Sprite.move(0.f, 1.f);
46.              }
47.
```

```
48.             if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
49.             {
50.                 sf::Vector2i mousePos = sf::Mouse::getPosition(window);
51.                 Sprite.setPosition((float)mousePos.x, (float)(mousePos.y));
52.             }
53.             window.clear();
54.             window.draw(Sprite);
55.             window.display();
56.         }
57.         return 0;
58.     }
```

PS1: Linear Feedback Shift Register

Assignment Discussion:

Assignment PS1 involves implementing a Linear Feedback Shift Register class, producing pseudo-random bits, and then using that to implement a simple form of encryption for digital pictures. The first part of the project, we used Boost test framework, ensuring the LFSR would operate correctly and running a test checking the output. The LFSR would operate by shifting one of the bits to the left and replacing the vacated bit by the exclusive or XOR, of the shifted off and the bit previously at a given tap position in the register.

To encrypt the picture, we initialized the registered with a set state, then this would act as the encryption key. Each pixel is presented by an 8-bit value, we than used LFSR to generate 8 bit integers, overwriting the original image until it was encrypted. The encrypted image is then decrypted using the same method.

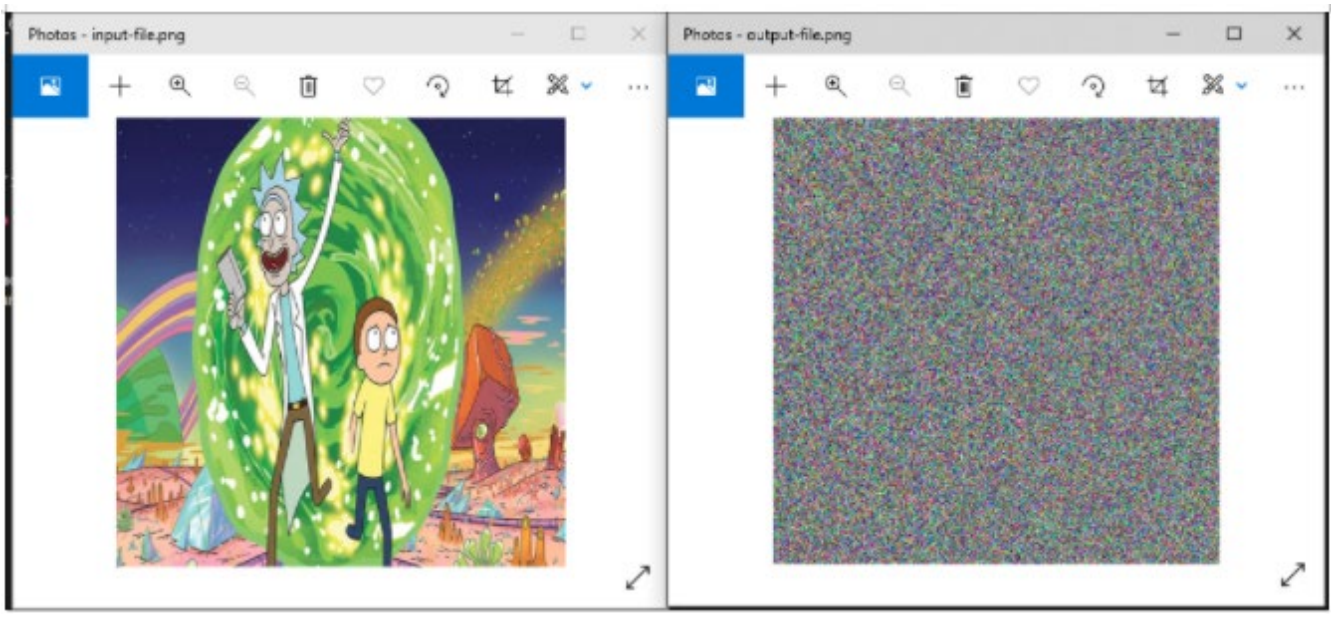
Key Algorithms, data structures, or OO designs that were central to the assignment:

For PS1, we used a vector of boolean, with each boolean representing a bit in the LFSR, using a 1/0 or True/False value. Each value was then held in the register for all the bits, and then was able to use vectors. To move bits from the back of the register to the front.

What I learned:

The PS1 assignment helped me learn about Boost test frame work and bitwise operators. The project required bitwise shifting and using the XOR function, which helped me understand binary values. Using the command line was another additional thing gained from PS1, and entering arguments for input. PS1 also helped me learn about Makefiles and using them to compile code.

Screenshot(s) of Output:



Project Source Code:

Makefile:

```
all: PhotoMagic

PhotoMagic : PhotoMagic.o FibLFSR.o
	g++ FibLFSR.o PhotoMagic.o - lsfml - graphics - lsfml - window - lsfml - system - o PhotoMagic

FibLFSR.o : FibLFSR.cpp FibLFSR.hpp
	g++ FibLFSR.cpp - std = c++11 - c - g - Og - Wall - Werror - pedantic

PhotoMagic.o : PhotoMagic.cpp
		g++ PhotoMagic.cpp - std = c++11 - c - g - Og - Wall - Werror – pedantic

	clean :
		rm PhotoMagic * .o
```

test.cpp:

```
#include <iostream>
#include <string>
#include "FibLFSR.h"
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
```

```
9.
10.           FibLFSR l("1011011000110110");
11.           BOOST_REQUIRE(l.step() == 0);
12.           BOOST_REQUIRE(l.step() == 0);
13.           BOOST_REQUIRE(l.step() == 0);
14.           BOOST_REQUIRE(l.step() == 1);
15.           BOOST_REQUIRE(l.step() == 1);
16.           BOOST_REQUIRE(l.step() == 0);
17.           BOOST_REQUIRE(l.step() == 0);
18.           BOOST_REQUIRE(l.step() == 1);
19.           FibLFSR l2("1011011000110110");
20.           BOOST_REQUIRE(l2.generate(9) == 51);
21.     }
22.
23.     BOOST_AUTO_TEST_CASE(testOne) {
24.           FibLFSR l3("01110011100");
25.           BOOST_REQUIRE(l3.step() == 0);
26.           BOOST_REQUIRE(l3.step() == 1);
27.           BOOST_REQUIRE(l3.step() == 0);
28.           BOOST_REQUIRE(l3.step() == 1);
29.           BOOST_REQUIRE(l3.step() == 0);
30.           BOOST_REQUIRE(l3.generate(7) == 20);
31.     }
32.     BOOST_AUTO_TEST_CASE(testTwo) {
33.           FibLFSR l4("111111111111");
34.           BOOST_REQUIRE(l4.step() == 0);
35.           BOOST_REQUIRE(l4.step() == 0);
36.           BOOST_REQUIRE(l4.step() == 0);
37.           BOOST_REQUIRE(l4.step() == 0);
38.           BOOST_REQUIRE(l4.step() == 0);
39.           BOOST_REQUIRE(l4.generate(9) == 1);
40.     }
```

PhotoMagic.cpp

```
1. /**Copyright 2021 <Wesley Gallo>
2. */
3.
4. #include "FibLFSR.hpp"
5. #include <iostream>
6. #include <string>
7. #include <SFML/System.hpp>
8. #include <SFML/Window.hpp>
9. #include <SFML/Graphics.hpp>
10.
11.     using std::string;
12.     using std::cout;
13.     using std::endl;
```

```
14.
15.      void transform(sf::Image& image, FibLFSR& lsfr);
16.      void alphaToBinary(string& seed);
17.
18.      int main(int argc, char* argv[]) {
19.          if (argc != 4) {
20.              cout << "you did not pass the correct amount of arguments to run the
    program" << endl;
21.              return -1;
22.          }
23.          string inputFileName = argv[1];
24.          string outputFileName = argv[2];
25.          string seed = argv[3];
26.
27.          alphaToBinary(seed);
28.          FibLFSR lfsr(seed);
29.
30.          sf::Image image;
31.          if (!image.loadFromFile(inputFileName)) {
32.              cout << "could not open the image" << inputFileName << endl;
33.              return -1;
34.          }
35.
36.          sf::Texture texture;
37.          texture.loadFromFile(inputFileName);
38.          sf::Sprite origionalSprite(texture);
39.
40.          sf::Texture texture2;
41.          transform(image, lfsr);
42.          texture2.loadFromImage(image);
43.          sf::Sprite encryptedSprite(texture2);
44.
45.          sf::Vector2u size = image.getSize();
46.          sf::RenderWindow oWindow(sf::VideoMode(size.x, size.y), "Origional Image");
47.          sf::RenderWindow eWindow(sf::VideoMode(size.x, size.y), "Encrypted Image");
48.
49.          if (!image.saveToFile(outputFileName)) {
50.              cout << "could not save encrypted image to file" << endl;
51.              return -1;
52.          }
53.
54.          while (oWindow.isOpen() && eWindow.isOpen()) {
55.              sf::Event event;
56.              while (oWindow.pollEvent(event)) {
57.                  if (event.type == sf::Event::Closed)
58.                      oWindow.close();
59.              }
60.              while (eWindow.pollEvent(event)) {
61.                  if (event.type == sf::Event::Closed)
62.                      eWindow.close();
63.              }
64.              oWindow.clear();
```

```
            oWindow.draw(origionalSprite);
            oWindow.display();
            eWindow.clear();
            eWindow.draw(encryptedSprite);
            eWindow.display();
        }

        return 0;
    }

    void transform(sf::Image& image, FibLFSR& lsfr) {
        sf::Color c;
        sf::Vector2u size = image.getSize();

        for (unsigned int x = 0; x < size.x; x++) {
            for (unsigned int y = 0; y < size.y; y++) {
                c = image.getPixel(x, y);
                c.r ^= lsfr.generate(8);
                c.g ^= lsfr.generate(8);
                c.b ^= lsfr.generate(8);
                image.setPixel(x, y, c);
            }
        }
    }

    void alphaToBinary(string& seed) {
        string binaryString;

        for (unsigned int i = 0; i < seed.length(); i++) {
            int j = static_cast<int>(seed[i]);
            while (j != 0) {
                binaryString.push_back(static_cast<char>('0' + (j % 2)));
                j /= 2;
            }
        }

        seed = binaryString;
    }
```

FibLFSR.h

```
/**Copyright 2021 <Wesley Gallo>
*/

#ifndef FIBLFSR_HPP
#define FIBLFSR_HPP
#include <string>
#include <iostream>

using namespace std;

```

```
11.  class FibLFSR {
12.  public:
13.      explicit FibLFSR(string seed);
14.      int step();
15.      int generate(int k);
16.      int getLength() const {
17.          return length;
18.      }
19.      int& operator[](int index) {
20.          return data[index];
21.      }
22.
23.      friend ostream& operator<<(ostream& out, const FibLFSR& object);
24.
25.      ~FibLFSR();
26.  private:
27.      int* data;
28.      int length;
29.  };
30.
31.  ostream& operator<<(ostream& out, const FibLFSR& object);
32.
33.  #endif  // C__USERS_GALLO_TMR57A3_ONEDRIVE_DESKTOP_PHOTOMAGIC_FIBLFSR_HPP_
```

FibLFSR.cpp

```
1. /**Copyright 2021 <Wesley Gallo>
2. * FibLFSR.cpp - PS1b
3. */
4.
5. #include "FibLFSR.hpp"
6. #include <string>
7. #include <iostream>
8.
9. using namespace std;
10.
11.      FibLFSR::FibLFSR(string seed) {
12.          length = seed.length();
13.          data = new int[length];
14.          for (int i = 0; i < length; i++) {
15.              data[i] = (seed[i] == '1') ? 1 : 0;
16.          }
17.      }
18.
19.      int FibLFSR::step() {
20.          int bit = data[0] ^ data[2] ^ data[3] ^ data[5];
21.
22.          for (int i = 0; i < length - 1; i++) {
23.              data[i] = data[i + 1];
24.          }
25.          data[length - 1] = bit;
```

```
26.
27.         return data[length - 1];
28.     }
29.     int FibLFSR::generate(int k) {
30.         int num = 0;
31.         for (int i = 0; i < k; i++) {
32.             num *= 2;
33.             num += step();
34.         }
35.
36.         return num;
37.     }
38.
39.     FibLFSR::~FibLFSR() {
40.         delete[] data;
41.         data = nullptr;
42.     }
43.
44.
45.     ostream& operator<<(ostream& out, const FibLFSR& object) {
46.         for (int i = 0; i < object.length; i++) {
47.             out << object.data[i];
48.         }
49.
50.         return out;
51.     }
```

PS2: N-Body Simulation

Assignment Discussion:

In assignment PS2 we were asked to create a program that loaded and displayed a static universe, using Newtons laws of Physics. In the simulation, the images of the celestial bodies would follow and simulate movement over time. This was represented by each frame of the output animation. The data was then read as input, and would be output as a .txt file.

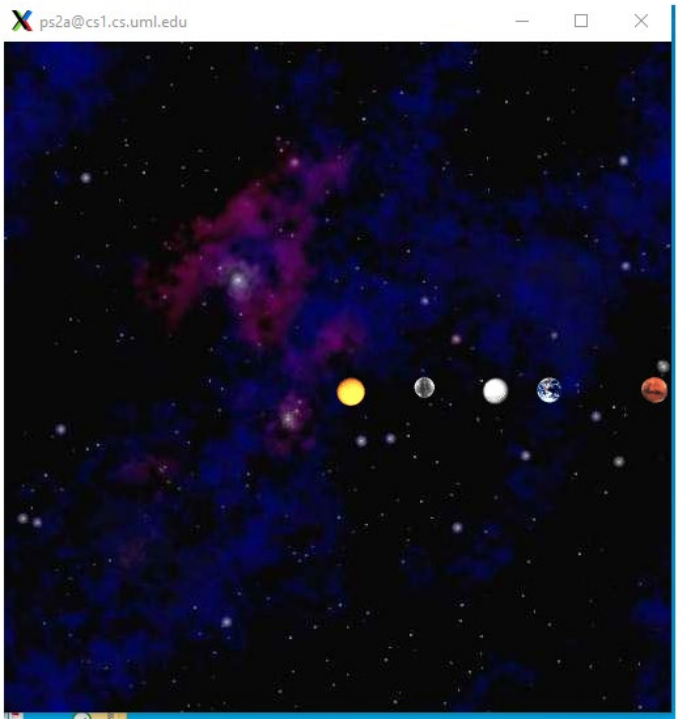
Key Algorithms, data structures, or OO designs that were central to the assignment:

For this assignment we created two classes, a “Universe” class and “CelestialBody” class. The Universe class would hold each of the celestial bodies that were created. The Celestial Body class held the data of the size, position, velocity of each body, and the Universe class would use this information to create and update each image in the simulation. We used static member functions and variables in order to be efficient with the data, and each object in the CelestialBody class would reference the Universe center coordinate and radius. Using static member functions ensured there was only one copy of each value to shared within the instance of the class.

What I learned:

In this assignment we were introduced to smart pointers, which were very useful when programming this assignment. The Universe class used smart pointers to store each celestial body within itself.

Screenshot(s) of Output:



Project Source Code:

Makefile:

```
all: NBody

NBody: CelestialBody.o Universe.o
	g++ CelestialBody.o Universe.o -o NBody -std=c++11 -lsfml-graphics -lsfml-window -lsfml-system

Universe.o: Universe.cpp
	g++ -c Universe.cpp -std=c++11 -Wall -Werror -ansi -pedantic

CelestialBody.o: CelestialBody.cpp CelestialBody.hpp
		g++ -c CelestialBody.cpp CelestialBody.hpp -std=c++11 -Wall -Werror -ansi -pedantic

	clean:
	rm *.o NBody *~
```

Universe.cpp

```
1. /**
2. * Universe.cpp - PS2a
3. * Date: 10/11/2021
4. * created by: Wesley Gallo
5. */
6.
7. #include <SFML/Graphics.hpp>
8. #include <SFML/Window.hpp>
9. #include <iostream>
10.      #include <vector>
11.      #include <cmath>
12.      #include "CelestialBody.hpp"
13.
14.      sf::Vector2f force(CelestialBody p1, CelestialBody p2)
15.      {
16.        sf::Vector2f planet_force;
17.        double totalForce, dx, dy, radius, radius_squared;
18.        double grav = (-6.67e-11);
19.
20.        dx = (((p1.getPos()).x) - ((p2.getPos()).x));
21.        dy = (((p1.getPos()).y) - ((p2.getPos()).y));
22.
23.        radius_squared = pow(dx, 2) + pow(dy, 2);
24.        radius = sqrt(radius_squared);
25.
26.        totalForce = ((grav) * (p1.getMass()) * (p2.getMass())) / (radius_squared);
27.
28.        planet_force.x = totalForce * (dx/radius);
29.        planet_force.y = totalForce * (dy/radius);
30.
31.        return planet_force;
32.      }
33.
34.      void removeWhitespace(std::string& str)
35.      {
36.        for (size_t i = 0; i < str.length(); i++)
37.          {
38.            if (str[i] == ' ' || str[i] == '\n' || str[i] == '\t') {
39.             str.erase(i, 1);
40.             i--;
41.            }
42.          }
43.      }
44.
45.      int main(int argc, char* argv[])
46.      {
47.        int planet_count;
48.
49.        double univ_radius;
50.        double time, time_step, current;
```

```
51.
52.         std::string planet_name;
53.         std::string planet_filename;
54.
55.         std::vector<CelestialBody> planets;
56.
57.         sf::Vector2f t_force;
58.         sf::Image background;
59.         sf::Font font;
60.         sf::Text timestamp;
61.
62.           if(argc != 3)
63.           {
64.             std::cout << "Argument Error" << std::endl;
65.             exit(1);
66.           }
67.
68.         timestamp.setFont(font);
69.         timestamp.setScale(0.5f, 0.5f);
70.         timestamp.setPosition(0.f, 0.f);
71.
72.         if(!background.loadFromFile("starfield.jpg"))
73.           return -1;
74.
75.         sf::Texture texture;
76.         texture.loadFromImage(background);
77.
78.         sf::Sprite sprite;
79.         sprite.setTexture(texture);
80.         sf::Vector2f size;
81.
82.         size.x = background.getSize().x;
83.         size.y = background.getSize().y;
84.
85.         std::cin >> planet_count >> univ_radius ;
86.
87.         for (int i = 0; i < planet_count; i++)
88.           {
89.           CelestialBody *planet = new CelestialBody (planet_count, univ_radius, size);
90.
91.           std::cin >> *planet;
92.           getline(std::cin, planet_filename);
93.           removeWhitespace(planet_filename);
94.           planet_name = planet_filename;
95.           planet->setName(planet_name);
96.           planet->setFile(planet_filename);
97.           planets.push_back(*planet);
98.           }
99.
100.        sf::RenderWindow window( sf::VideoMode(size.x, size.y), "ps2b");
101.
```

```
      while(window.isOpen()) {
        sf::Event event;
        while(window.pollEvent(event)) {
          if(event.type == sf::Event::Closed) {
           window.close();
           break;
          }
        }
        window.draw(sprite);

        for(std::vector<CelestialBody>::iterator iter = planets.begin(); iter != planets.end(); iter++)
          {
           iter->setUpPos();
           window.draw(*iter);
          }

        if (current < time)
          {
           for(std::vector<CelestialBody>::iterator iter1 = planets.begin(); iter1 != planets.end(); ++iter1) {
             sf::Vector2f Force_net;
             for(std::vector<CelestialBody>::iterator iter2 = planets.begin(); iter2 != planets.end(); ++iter2) {
                if (iter1->getName() != iter2->getName())
                  {
                   t_force = force(*iter1, *iter2);
                   Force_net.x = Force_net.x + t_force.x;
                   Force_net.y = Force_net.y + t_force.y;
                  }
             }
             iter1->setFnet(Force_net);
           }

           for (std::vector<CelestialBody>::iterator iter = planets.begin() ; iter != planets.end() ; iter++ ){
             iter->step(time_step);
             window.draw(*iter);
           }
           std::cout << current << std::endl;
           timestamp.setString(std::to_string(current));
           current += time_step;
          }
        window.draw(timestamp);
        window.display();
      }
      return 0 ;
    }
```

CelestialBody.cpp

```
1. /**
```

```
 * CelestialBody.cpp - PS2a
 * Date: 10/11/2021
 * created by: Wesley Gallo
 */

#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <iostream>
#include "CelestialBody.hpp"

CelestialBody::CelestialBody(int univ_size, double univ_radius, sf::Vector2f
window_size) {
  setUniverseSize(univ_size);
  setUniverseRadius(univ_radius);
  setWindowSize(window_size);
}

void CelestialBody::setName(sf::String name) {
  _planet_name = name;
}

void CelestialBody::setFile(sf::String file) {
  _filename = file;
}

void CelestialBody::setPosition(sf::Vector2f pos) {
  _pos = pos;
}

void CelestialBody::setVelocity(sf::Vector2f velo) {
  _velo = velo;
}

void CelestialBody::setMass(double mass) {
  _mass = mass;
}

void CelestialBody::setAccel(sf::Vector2f accel) {
  _accel = accel;
}

void CelestialBody::setUniverseSize(double univ_size) {
  _univ_size = univ_size;
}

void CelestialBody::setUniverseRadius(double univ_radius) {
  _univ_radius = univ_radius;
}

void CelestialBody::setWindowSize(sf::Vector2f window_size) {
```

```
  _window_size = window_size;
}

void CelestialBody::setnormalPos(sf::Vector2f normalPos) {
  _normalPos = normalPos;
}

void CelestialBody::setFnet(sf::Vector2f force) {
  _Fnet = force;
}

sf::Vector2f CelestialBody::getVelocity() {
  return _velo;
}

double CelestialBody::getMass() {
  return _mass;
}

sf::String CelestialBody::getName() {
  return _planet_name;
}

sf::String CelestialBody::getFile() const {
  return _filename;
}

double CelestialBody::getUniverseRadius() {
  return _univ_radius;
}

double CelestialBody::getUniverseSize() {
  return _univ_size;
}

sf::Vector2f CelestialBody::getWindowSize() {
  return _window_size;
}

sf::Vector2f CelestialBody::getPos() const {
  return _pos;
}

sf::Vector2f CelestialBody::getAccel() const {
  return _accel;
}

sf::Vector2f CelestialBody::getnormalPos() const {
  return _normalPos;
      }
```

```
101.
102.      sf::Vector2f CelestialBody::getFnet() const {
103.        return _Fnet;
104.      }
105.
106.      std::istream& operator >> (std::istream& in, CelestialBody& CelestialBody) {
107.        sf::Vector2f pos;
108.        sf::Vector2f vel;
109.        sf::Vector2f accel;
110.        double _mass;
111.        in >> (pos.x) >> (pos.y) >> (vel.x) >> (vel.y) >> _mass;
112.        CelestialBody.setPosition(pos);
113.        CelestialBody.setVelocity(vel);
114.        CelestialBody.setMass(_mass);
115.        return in;
116.      }
117.
118.      void CelestialBody::setUpPos() {
119.        sf::Vector2f velo;
120.        sf::Vector2f pos;
121.        (pos.x) = (((getWindowSize().x) / 2) +
122.                   ((((getPos()).x) / getUniverseRadius()) *
123.                    ((getWindowSize().x) / 2)));
124.        (pos.y) = (((getWindowSize().y) / 2) - (
125.                     (((getPos()).y) / getUniverseRadius()) *
126.                     ((getWindowSize().y) / 2)));
127.        setnormalPos(pos);
128.      }
129.
130.      void CelestialBody::draw(sf::RenderTarget &target, sf::RenderStates state) const
    {
131.        graphics(target, state);
132.      }
133.
134.      void CelestialBody::graphics(sf::RenderTarget &target, sf::RenderStates state)
    const {
135.        sf::Sprite sprite;
136.        sf::Image image;
137.        sf::Texture texture;
138.        image.loadFromFile(getFile());
139.        texture.loadFromImage(image);
140.        sprite.setTexture(texture);
141.        sprite.setPosition((getnormalPos()).x, (getnormalPos()).y);
142.        target.draw(sprite, state);
143.      }
144.
145.      void CelestialBody::setNewVelo(double seconds) {
146.        sf::Vector2f velo;
147.        velo.x = ((getVelocity()).x + (seconds * (getAccel()).x));
```

```
148.        velo.y = ((getVelocity()).y + (seconds * (getAccel()).y));
149.        setVelocity(velo);
150.      }
151.
152.      void CelestialBody::setNewPos(double seconds) {
153.        sf::Vector2f pos;
154.        pos.x = ((getPos()).x + (seconds * (getVelocity()).x));
155.        pos.y = ((getPos()).y + (seconds * (getVelocity()).y));
156.        setPosition(pos);
157.      }
158.
159.      void CelestialBody::setNewAccel() {
160.        sf::Vector2f accel;
161.        accel.x = ((getFnet()).x / (getMass()));
162.        accel.y = ((getFnet()).y / (getMass()));
163.        setAccel(accel);
164.      }
165.
166.      void CelestialBody::step(double seconds) {
167.        setNewAccel();
168.        setNewVelo(seconds);
169.        setNewPos(seconds);
170.        setUpPos();
171.      }
```

CelestialBody.hpp

```
1. /**
2. * CelestialBody.hpp - PS2a
3. * Date: 10/11/2021
4. * created by: Wesley Gallo
5. */
6. #ifndef N_BODY_HPP_
7. #define N_BODY_HPP_
8.
9. #include <SFML/Graphics.hpp>
10.      #include <SFML/Window.hpp>
11.
12.      class CelestialBody : public sf::Drawable {
13.       private:
14.        sf::Vector2f _window_size, _pos, _velo, _accel, _Fnet, _normalPos;
15.        double _univ_size, _univ_radius, _mass;
16.        sf::String _filename;
17.        sf::String _planet_name;
18.        virtual void draw( sf::RenderTarget &target, sf::RenderStates state ) const ;
19.
20.       public:
21.        CelestialBody(int univ_size, double univ_radius, sf::Vector2f window_size);
22.        void setName(sf::String name) ;
23.        void setFile(sf::String file);
```

```
24.         void setPosition(sf::Vector2f pos);
25.         void setVelocity(sf::Vector2f velo);
26.         void setMass(double mass);
27.         void setUniverseSize(double univ_size);
28.         void setUniverseRadius(double univ_radius);
29.         void setWindowSize(sf::Vector2f window_size);
30.         void setnormalPos(sf::Vector2f normalPos);
31.         void setAccel(sf::Vector2f accel);
32.         void setFnet(sf::Vector2f force);
33.
34.         double getUniverseSize();
35.         double getMass();
36.         double getUniverseRadius();
37.         sf::Vector2f getVelocity();
38.         sf::Vector2f getAccel() const;
39.         sf::Vector2f getWindowSize();
40.
41.         sf::Texture getTexture() const;
42.         sf::Sprite getSprite() const;
43.         sf::String getName();
44.         sf::String getFile() const;
45.
46.         sf::Vector2f getPos() const;
47.         sf::Vector2f getnormalPos() const;
48.         sf::Vector2f getFnet() const;
49.         friend std::istream& operator >> (std::istream& in, CelestialBody&
    CelestialBody);
50.
51.         void step(double seconds);
52.         void setNewAccel();
53.         void setNewVelo(double seconds);
54.         void setNewPos(double seconds);
55.         void setUpPos();
56.         void graphics( sf::RenderTarget &target, sf::RenderStates state ) const;
57.       };
58.       #endif
```

PS3: Recursive Graphics (Triangle Fractal)

Assignment Discussion:

In assignment PS3, we wrote a program that plotted Triangle Fractals, which was a variation of the Siepinski triangle. The program took two command line arguments, and using the length and depth of recursion is able to draw a triangle. The program was able to read the input and then opened a window with the final triangle. In addition to the assignment, the code was formatted to pass cpplint.

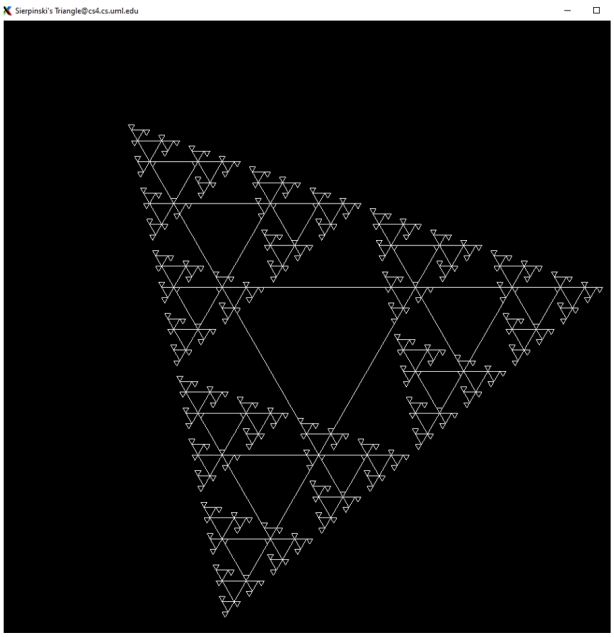
Key Algorithms, data structures, or OO designs that were central to the assignment:

The triangle was drawn from a central point, and then that was used to calculate the coordinates of the next three triangles. Ftree was used on a loop to draw the triangles.

What I learned:

In the assignment, I used the recursive function to create a window for the triangle and render the triangle. In addition the use of cpplint, was interesting in changing the format of our code to be compliant.

Screenshot(s) of Output:



Project Source Code:

Makefile

```
1. all: ps3
2.
3. ps3: TFractal.o Triangle.o
4.      g++ TFractal.o Triangle.o -o TFractal -std=c++11 -lsfml-graphics -lsfml-window -
   lsfml-system
5.
6. TFractal.cpp: TFractal.cpp
7.      g++ -c TFractal.cpp Triangle.h -std=c++11 -Wall -Werror -ansi -pedantic
8.
9. Triangle.o: Triangle.cpp Triangle.h
```

```
10.          g++ -c Triangle.cpp Triangle.h -std=c++11 -Wall -Werror -ansi -pedantic
11.
12.    clean:
13.          rm *.o TFractal *~
```

Tfractal.cpp

```
1. /**
2. * TFractal - PS3
3. * Date: 10/18/2021
4. * created by: Wesley Gallo
5. */
6.
7. #include "Triangle.h"
8.
9. void fTree(int depth, double x, double y,
10.                      double length, sf::RenderWindow &window);
11.      int main(int argc, char*argv[]) {
12.          double length = atof(argv[1]);
13.          int depth = atoi(argv[2]);
14.
15.          sf::RenderWindow window(sf::VideoMode(1000, 1000),
16.                                  "Sierpinski's Triangle");
17.          while (window.isOpen()) {
18.              sf::Event event;
19.              while (window.pollEvent(event)) {
20.                  if (event.type == sf::Event::Closed)
21.                  window.close();
22.              }
23.
24.              fTree(depth, 500, 500, length, window);
25.              window.display();
26.          }
27.
28.          return 0;
29.      }
30.
31.      void fTree(int depth, double x, double y,
32.                  double length, sf::RenderWindow &window) {
33.          if (depth < 0)
34.              return;
35.          double h = (sqrt(3)/2.0) * length;
36.
37.          sf::Vector2f xy1(x - (length/2.0), y - (h/3.0));
38.          sf::Vector2f xy2(x + (length/2.0), y - (h/3.0));
39.          sf::Vector2f xy3(x, y + (2.0 * (h/3.0)));
40.
41.          Triangle triangle(xy1, xy2, xy3);
42.          window.draw(triangle);
43.          fTree(depth - 1, x - (length/2.0), y - (2.0 * (h/3.0)), length/2.0, window);
44.          fTree(depth - 1, x + ((3 * length)/4.0), y - (h/6.0), length/2.0, window);
```

```
45.          fTree(depth - 1, x - (length/4.0), y + (5 * (h/6.0)), length/2.0, window);
46.      }
```

Triangle.cpp

```
1. /**
2. * Triangle.cpp - PS3
3. * Date: 10/18/2021
4. * created by: Wesley Gallo
5. */
6.
7. #include "Triangle.h"
8. Triangle::Triangle(const sf::Vector2f& p1,
9.                     const sf::Vector2f& p2, const sf::Vector2f& p3) {
10.          triangle = sf::VertexArray(sf::LineStrip, 4);
11.          triangle[0].position = p1;
12.          triangle[1].position = p2;
13.          triangle[2].position = p3;
14.          triangle[3].position = triangle[0].position;
15.      }
16.
17.      void Triangle::draw(sf::RenderTarget& target, sf::RenderStates states) const {
18.          target.draw(triangle);
19.      }
```

Triangle.h

```
1. /**
2. * Triangle.h - PS3
3. * Date: 10/18/2021
4. * created by: Wesley Gallo
5. */
6.
7. #include <iostream>
8. #include <cmath>
9. #include <SFML/Graphics.hpp>
10.
11.      class Triangle : public sf::Drawable {
12.       public:
13.        Triangle(const sf::Vector2f& p1,
14.                      const sf::Vector2f& p2, const sf::Vector2f& p3);
15.        void draw(sf::RenderTarget& target, sf::RenderStates states) const;
16.       private:
17.        sf::VertexArray triangle;
18.      };
```

PS4: Synthesizing a Plucked String Sound

Assignment Discussion:

Assignment PS4 e implemented a guitar string simulation, and generated a stream of string samples for audio playback under keyboard control. A Ring Buffer was implemented, which is a queue of fixed-size filled with random values. The queue works by deleting the value in front of the queue and put a new value equal to the average of the next value and the delete value, which is then multiplied by the decay factor. This simulates the guitar or string sound. The Boost unit test framework was used to, complete the CircularBuffer class. Using keys on the keyboard, the user is able to play different sounds.

Key Algorithms, data structures, or OO designs that were central to the assignment:

Assignment PS4, exceptions were used in the Boost test framework. The functions tested the date and returned a boolean value based on the argument or called function. The exceptions were thrown, and then the user could know where the error occurred. Lamba expressions were used to generate samples of the notes. I implemented exceptions to check StringSound as well.

What I learned:

PS4 was great for understanding how to program sound and how computers interpret them. While also generating sound data with algorithms.

Screenshot(s) of Output:

```
wgallo@cs3:~$ ./test
Running 1 test case...
Sample is: 0
Sample is: 1000
Sample is: 2000
Sample is: -5000
Sample is: 498
Sample is: 1494
Sample is: -1494
Sample is: -2241
Sample is: 992
Sample is: 0

*** No errors detected
```

```
wgallo@cs3:~$ ./KSGuitarSim
Setting vertical sync not supported
ja323key is: d
playing..
key is: f
playing..
key is: r
playing..
key is: g
playing..
key is: g
playing..
```

Project Source Code:

Makefile

```
CC= g++
CFLAGS= -g -Wall -Werror -pedantic -O3 -std=c++14
SFLAGS= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
Boost= -lboost_unit_test_framework

all:   KSGuitarSim test

KSGuitarSim:  KSGuitarSim.o StringSound.o CircularBuffer.o
	$(CC) KSGuitarSim.o StringSound.o CircularBuffer.o -o KSGuitarSim $(SFLAGS)

	test: test.o StringSound.o CircularBuffer.o
		$(CC) test.o StringSound.o CircularBuffer.o -o test $(Boost)

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

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

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

	test.o:	 test.cpp
		$(CC) -c test.cpp $(Boost)

	clean:
		rm *.o
		rm *.gch
		rm KSGuitarSim
	rm test
```

Test.cpp

```
/**
* test.cpp - ps4b
* Date: 11/2/2021
* created by: Wesley Gallo
*/

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

```
10.     #include <exception>
11.     #include <stdexcept>
12.     #include <vector>
13.     #include "StringSound.h"
14.
15.     BOOST_AUTO_TEST_CASE(SS) {
16.       std::vector<sf::Int16> v;
17.
18.       v.push_back(0);
19.       v.push_back(1000);
20.       v.push_back(2000);
21.       v.push_back(-5000);
22.
23.       BOOST_REQUIRE_NO_THROW(StringSound ss = StringSound(v));
24.
25.       StringSound ss = StringSound(v);
26.
27.       // SS is 0 1000 2000 -5000
28.       std::cout << "Sample is: " << ss.sample() << "\n";
29.       BOOST_REQUIRE(ss.sample() == 0);
30.
31.       ss.tic();
32.       //  now 1000 2000 -5000 498
33.       std::cout << "Sample is: " << ss.sample() << "\n";
34.       BOOST_REQUIRE(ss.sample() == 1000);
35.
36.       ss.tic();
37.       //  now 2000 -5000 498 1494
38.       std::cout << "Sample is: " << ss.sample() << "\n";
39.       BOOST_REQUIRE(ss.sample() == 2000);
40.
41.       ss.tic();
42.       //  now -5000 498 1494 -1494
43.       std::cout << "Sample is: " << ss.sample() << "\n";
44.       BOOST_REQUIRE(ss.sample() == -5000);
45.
46.       ss.tic();
47.       //  now 498 1494 -1494 -2241
48.       std::cout << "Sample is: " << ss.sample() << "\n";
49.       BOOST_REQUIRE(ss.sample() == 498);
50.
51.       ss.tic();
52.       //  now 1494 -1494 -2241 992
53.       std::cout << "Sample is: " << ss.sample() << "\n";
54.       BOOST_REQUIRE(ss.sample() == 1494);
55.
56.       ss.tic();
57.       //  now -1494 -2241 992 0
58.       std::cout << "Sample is: " << ss.sample() << "\n";
59.       BOOST_REQUIRE(ss.sample() == -1494);
60.
61.       ss.tic();
```

```
    std::cout << "Sample is: " << ss.sample() << "\n";
    BOOST_REQUIRE(ss.sample() == -2241);
    ss.tic();
    std::cout << "Sample is: " << ss.sample() << "\n";
    BOOST_REQUIRE(ss.sample() == 992);
    ss.tic();
    std::cout << "Sample is: " << ss.sample() << "\n";
    BOOST_REQUIRE(ss.sample() == 0);
  }
```

KSGuitarSim.cpp

```
/**
* KSGuitarSim.cpp - ps4b
* Date: 11/2/2021
* created by: Wesley Gallo
*/

#include <SFML/Audio.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
    #include <SFML/Window.hpp>
    #include <limits.h>
    #include <math.h>
    #include <exception>
    #include <iostream>
    #include <stdexcept>
    #include <string>
    #include <vector>
    #include "StringSound.h"

    #define CONCERT 440.0
    #define SAMPLE_RATE 44100
    const int key_size = 37;

    using std::string;

    vector<sf::Int16> makeSample(StringSound &sound) {
      vector<sf::Int16> samples;

      sound.pluck();

      int duration = 8;

      for (int i = 0; i < SAMPLE_RATE * duration; i++) {
        sound.tic();
        samples.push_back(sound.sample());
      }
      return samples;
    }

    int main() {
```

```
41.         // const int key_size = 37;
42.
43.         sf::RenderWindow window(sf::VideoMode(800, 800), "KSGuitarSim");
44.         sf::Event event;
45.
46.         double freq;
47.         vector<sf::Int16> sample;
48.         string keyboard =
49.             "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";  // keyboard arrangement
50.
51.         vector<std::vector<sf::Int16>> samples(
52.             key_size);  // holds audio sample stream generated by StringSound
53.         vector<sf::SoundBuffer> buffer(key_size);
54.         vector<sf::Sound> sounds(key_size);  // vector of sounds to be played
55.
56.         for (int i = 0; i < (signed)keyboard.size(); i++) {
57.           freq = CONCERT * pow(2, (i - 24) / 12.0);  // get frequency of each key
58.
59.           StringSound temp(freq);
60.           sample = makeSample(temp);
61.
62.           samples[i] = makeSample(temp);
63.
64.           if (i >= key_size)
65.             return -1;
66.           if (!buffer[i].loadFromSamples(&samples[i][0], samples[i].size(), 2,
67.                                          SAMPLE_RATE)) {
68.             throw std::runtime_error("failed to load");
69.           }
70.
71.           sounds[i].setBuffer(buffer[i]);
72.         }
73.
74.         while (window.isOpen()) {
75.           while (window.pollEvent(event)) {
76.             if (event.type == sf::Event::Closed) {
77.               window.close();
78.             } else if (event.type == sf::Event::TextEntered) {
79.               // make sure char is in ASCII range
80.               if (event.text.unicode < 128) {
81.                 char key = static_cast<char>(event.text.unicode);  // cast to a char
82.                 // loop through keyboard
83.                 for (int j = 0; j < (signed)keyboard.size(); j++) {
84.                   // matching key found
85.                   if (keyboard[j] == key) {
86.                     cout << "key is: " << keyboard[j] << endl;
87.                     cout << "playing.." << endl;
88.                     sounds[j].play();  // play the sound for that key
89.                     break;
90.                   }
91.                 }
92.               }
```

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

StringSound.cpp

```
/**
* StringSound.cpp - ps4b
* Date: 11/2/2021
* created by: Wesley Gallo
*/

#include "StringSound.h"

StringSound::StringSound(double frequency) {
        num = ceil(SAMPLE_RATE / frequency);  // size of buffer
        try {
          buff = new CircularBuffer(num);
        } catch (const std::bad_alloc &e) {
          cout << "Allocation failed" << e.what() << std::endl;
        }

        for (int i = 0; i < num; i++) {
          buff->enqueue((int16_t)0);
        }
        _time = 0;
      }

      StringSound::StringSound(vector<sf::Int16> init) {
        num = init.size();
        try {
          buff = new CircularBuffer(num);
        } catch (const std::bad_alloc &e) {
          std::cout << "Allocation failed" << e.what() << std::endl;
        }
        vector<sf::Int16>::iterator it;

        for (it = init.begin(); it < init.end(); it++) {
          buff->enqueue((int16_t)*it);
        }
        _time = 0;
      }

      StringSound::~StringSound() {
        // std::cout << "Destructing" << std::endl;
        delete buff;
      }

```

```
43.      void StringSound::pluck() {
44.        std::random_device rd;
45.        std::mt19937 gen(rd());
46.        std::uniform_int_distribution<int16_t> random(-32768, 32767);
47.        // for (int i = 0; i < num; i++) // first make sure buffer is empty
48.        // {
49.        // buff->dequeue();
50.        // }
51.        buff->empty();
52.        // add on random nums
53.        for (int i = 0; i < num; i++) {
54.          buff->enqueue(random(gen));
55.        }
56.      }
57.
58.      void StringSound::tic() {
59.        int16_t first = buff->dequeue();
60.        int16_t second = buff->peek();
61.
62.        int16_t avg = (first + second) / 2;
63.
64.        int16_t kar = avg * ENERGY_DECAY_FACTOR;  // karplus algorithm
65.
66.        buff->enqueue((sf::Int16)kar);
67.
68.        _time++;
69.      }
70.
71.      sf::Int16 StringSound::sample() {
72.        sf::Int16 samp = (sf::Int16)buff->peek();
73.        return samp;
74.      }
75.
76.      int StringSound::time() { return _time; }
```

StringSound.h

```
1. /**
2. * StringSound.h - ps4b
3. * Date: 11/2/2021
4. * created by: Wesley Gallo
5. */
6.
7. #ifndef STRINGSOUND_H
8. #define STRINGSOUND_H
9.
10.      #include "CircularBuffer.h"
11.      #include <SFML/Audio.hpp>
12.      #include <SFML/Graphics.hpp>
13.      #include <SFML/System.hpp>
14.      #include <SFML/Window.hpp>
15.      #include <cmath>
```

```
#include <iostream>
#include <random>
#include <string>
#include <vector>

const int SAMPLE_RATE = 44100;
const double ENERGY_DECAY_FACTOR = 0.996;

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

class StringSound {
 public:
  explicit StringSound(double frequency);
  explicit StringSound(vector<sf::Int16> init);
  StringSound(const StringSound &obj) {}
  ~StringSound();

  void pluck();
  void tic();
  sf::Int16 sample();
  int time();

 private:
  CircularBuffer *buff;
  int _time;
  int num;
};

#endif
```

CircularBuffer.h

```
/**
* CircularBuffer.h - ps4b
* Date: 11/2/2021
* created by: Wesley Gallo
*/


#ifndef CIRCULARBUFFER_H_
#define CIRCULARBUFFER_H_
#include <stdint.h>
#include <exception>
#include <iostream>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <algorithm>
using std::cout;
```

```
19.      using std::endl;
20.
21.      class CircularBuffer {
22.       public:
23.        explicit CircularBuffer(int capacity);
24.        int size();
25.        bool isEmpty();
26.        bool isFull();
27.        void enqueue(int16_t x);
28.        int16_t dequeue();
29.        int16_t peek();
30.
31.        void output();
32.        void empty();
33.
34.       private:
35.        std::vector<int16_t> buffer;
36.        int first;
37.        int last;
38.        int cap;
39.        int numItems;
40.      };
41.      #endif  // CIRCULARBUFFER_H_
```

CircularBuffer.cpp

```
1. /**
2. * CircularBuffer.cpp - ps4b
3. * Date: 11/2/2021
4. * created by: Wesley Gallo
5. */
6.
7. #include "CircularBuffer.h"
8.
9. CircularBuffer::CircularBuffer(int capacity) {
10.        if (capacity < 1) {
11.          throw std::invalid_argument(
12.              "CircularBuffer constructor: capacity must be greater than zero.");
13.        }
14.        first = 0;
15.        last = 0;
16.        cap = capacity;
17.        numItems = 0;
18.        buffer.resize(cap);
19.      }
20.
21.      int CircularBuffer::size() { return numItems; }
22.
23.      bool CircularBuffer::isEmpty() {
24.        if (numItems == 0) {
25.          return true;
26.        }
```

```
27.       return false;
28.     }
29.
30.     bool CircularBuffer::isFull() {
31.       if (numItems >= cap) {
32.         return true;
33.       }
34.       return false;
35.     }
36.
37.     void CircularBuffer::enqueue(int16_t x) {
38.       if (isFull()) {
39.         throw std::runtime_error("enqueue: can't enqueue to a full ring");
40.         return;
41.       }
42.       else{
43.       if (last >= cap) {
44.         last = 0;
45.       }
46.       buffer.at(last) = x;
47.
48.       last++;
49.       numItems++;
50.       }
51.     }
52.
53.     // delete front and return it
54.     int16_t CircularBuffer::dequeue() {
55.       if (isEmpty()) {
56.         throw std::runtime_error("dequeue: can't dequeue from empty string");
57.         return -1;
58.       }
59.       else{
60.       int16_t first16 = buffer.at(first);  // to remove from front
61.       buffer.at(first) = 0;
62.
63.       first++;
64.       numItems--;
65.
66.       if (first >= cap) {
67.         first = 0;
68.       }
69.       return first16;
70.       }
71.     }
72.
73.     int16_t CircularBuffer::peek() {
74.       if (isEmpty()) {
75.         throw std::runtime_error("peek: can't peek from empty string");
76.         return -1;
77.       }
78.       else{
```

```
79.         return buffer.at(first);
80.         }
81.       }
82.
83.       void CircularBuffer::output() {
84.         cout << "First: " << first << endl;
85.         cout << "Last: " << last << endl;
86.         cout << "Capacity: " << cap << endl;
87.         cout << "Size: " << numItems << endl;
88.         cout << "Vector capacity: " << buffer.capacity() << endl;
89.         cout << "Vector size: " << buffer.size() << endl;
90.         cout << "Printing out buffer" << endl;
91.         for_each(buffer.begin(), buffer.end(), [](int16_t i){cout << i << " ";});
92.         cout << endl << endl;
93.       }
94.
95.       void CircularBuffer::empty() {
96.         first = 0;
97.         last = 0;
98.         numItems = 0;
99.       }
```

PS5: DNA Sequence Alignment

Assignment Discussion:

The DNA Sequence Alignment assignment was to learn about dynamic programming. In the assignment we would create a class that could compute the distance between two strings, and then would weight each distance to simulate mutations in DNA sequence. The created class would then accept two strings, and calculate the distance between them, and compute the optimal set, so both strings would be equal. In the assignment itself, we used deletion and insertion.

Key Algorithms, data structures, or OO designs that were central to the assignment:

In the assignment I choose to fill in the table and then fill the outer column. Then I stored the results as sub problems from another function. Using dynamic programming I was able to use a multi step approach. Which avoids recomputing over and over again.

What I learned:

Assignment 5 was great for understanding Dynamic Programming, and how to structure a program to run more optimal and timely.

Screenshot(s) of Output:

Had trouble with screenshot

Project Source Code:

Makefile:

```
CC = g++
CFLAGS = -std=c++11 -c -g -O2 -Wall -Werror -pedantic
OBJ = main.o ED.o
DEPS = main.cpp ED.cpp
LIBS = -lsfml-system
EXE = ED

all: $(OBJ)
	$(CC) $(OBJ) -o $(EXE) $(LIBS)

	%.o: %.cpp $(DEPS)
		$(CC) $(CFLAGS) -o $@ $<

	clean:
	rm $(OBJ) $(EXE)
```

ED.cpp

```

/**
* ED.cpp - ps5
* Date: 11/8/2021
* created by: Wesley Gallo
*/

#include <iostream>
#include "ED.h"

	ED::ED(const std::string &a, const std::string &b)
	{
	    this->x = a;
	    this->y = b;
	    M = x.size();
	    N = y.size();
	    opt = new int *[M + 1]();
	    if(!opt)
	    {
	        std::cout << "memory not enough." << std::endl;
	        exit(1);
	    }
	    for (int i = 0; i < M + 1; i++)
	        opt[i] = new int[N + 1]();
	    for (int i = 0; i <= M; i++)
	    {
	        opt[i][N] = 2 * (M - i);
	    }
	    for (int j = 0; j <= N; j++)
	    {
```

```
31.             opt[M][j] = 2 * (N - j);
32.         }
33.     }
34.
35.     int ED::penalty(char a, char b)
36.     {
37.         return a == b ? 0 : 1;
38.     }
39.
40.     int ED::min(int a, int b, int c)
41.     {
42.         int minAB = a > b ? b : a;
43.         return minAB > c ? c : minAB;
44.     }
45.
46.     int ED::OptDistance()
47.     {
48.         for (int i = M - 1; i >= 0; i--)
49.             for (int j = N - 1; j >= 0; j--)
50.                 opt[i][j] = min(opt[i + 1][j + 1] + penalty(x[i], y[j]), opt[i +
   1][j] + 2, opt[i][j + 1] + 2);
51.     return opt[0][0];
52.     }
53.
54.     std::string ED::Alignment()
55.     {
56.         std::string result;
57.         int i = 0, j = 0;
58.         while (i < M || j < N) {
59.             if (i < M && j < N && x[i] == y[j] && opt[i][j] == opt[i + 1][j + 1])
60.             {
61.                 result += x[i];
62.                 result += " ";
63.                 result += y[j];
64.                 result += " 0";
65.                 i++;
66.                 j++;
67.             } else if (i < M && j < N && x[i] != y[j] && opt[i][j] == opt[i + 1][j +
   1] + 1)
68.             {
69.                 result += x[i];
70.                 result += " ";
71.                 result += y[j];
72.                 result += " 1";
73.                 i++;
74.                 j++;
75.             } else if (i < M && opt[i][j] == opt[i + 1][j] + 2)
76.             {
77.                 result += x[i];
78.                 result += " - 2";
79.                 i++;
80.             } else if (j < N && opt[i][j] == opt[i][j + 1] + 2)
```

```
            {
                result += "- ";
                result += y[j];
                result += " 2";
                j++;
            }
            if(i < M || j < N) result += "\n";
        }
        return result;
    }

    ED::~ED()
    {
        for(int i = 0; i <= M; i++)
            delete[] opt[i];
        delete[] opt;
    }
```

ED.h

```
/**
* ED.h - ps5
* Date: 11/8/2021
* created by: Wesley Gallo
*/

#ifndef ED_H
#define ED_H
#include <string>

    class ED
    {

    private:
        int **opt;
        std::string x;
        std::string y;
        int M, N;

    public:
        ED(const std::string &a, const std::string &b);
        ~ED();

        static int penalty(char a, char b);
        static int min(int a, int b, int c);

        int OptDistance();
        std::string Alignment();
    };

    #endif
```

main.cpp

```
/**
* main.cpp - ps5
* Date: 11/8/2021
* created by: Wesley Gallo
*/

#include <iostream>
#include "ED.h"
#include <SFML/System.hpp>

      using namespace std;

      int main() {
          sf::Clock clock;
          sf::Time t;
          string sequence1, sequence2;
          cin >> sequence1 >> sequence2;
          ED ed(sequence1, sequence2);


          auto distance = ed.OptDistance();
          auto alignment = ed.Alignment();

          t = clock.getElapsedTime();
          cout << "Edit distance = " << distance << endl;
          cout << alignment << endl;
          cout << "Execution time is " << t.asSeconds() << " seconds \n" << std::endl;
          return 0;
      }
```

PS6: Random Writer

Assignment Discussion:

The PS6 Random Writer assignment was to analyze an input text for transitions between k-grams, and thee following letter. This would then produce a realistic model of text. Using probability, the program was able to accomplish this. The length of the text generated based on the input text. Using Boost test framework to test the assignment, ensuring it was complete and correct.

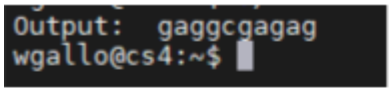
Key Algorithms, data structures, or OO designs that were central to the assignment:

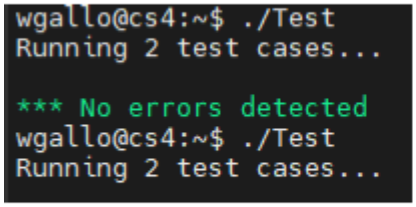
A created map would pair a K-gram string with a second map, containing each character that proceeded the k-gram, then sing an unsigned integer that represented the number of times a character would do this. The map allowed to avoid allocating the memory and provided a visual of the hierarchical structure.

What I learned:

Using an exception, both frequency functions kRand and Generate function were programmed to throw an error when it gets a string or character, who's length is not equal to the k variable. In addition the kRand function is also meant to throw a run time error when the argument could not be found in the kgrams variable.

Screenshot(s) of Output:





Project Source Code:

Makefile

```
1. CC=g++
2. CFLAGS=-g -O -Wall -Werror -std=c++11 -pedantic
3. Boost=-lboost_unit_test_framework
4.
```

```
all: TextWriter Test

TextWriter: TextWriter.o RandWriter.o
	$(CC) TextWriter.o RandWriter.o -o TextWriter

	Test:  Test.o RandWriter.o
		$(CC) Test.o RandWriter.o -o Test $(Boost)

	Test.o: Test.cpp
		$(CC) -c Test.cpp $(Boost)

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

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

	clean:
		rm  TextWriter Test Test.o RandWriter.o TextWriter.o
```

RandWriter.cpp

```
/**
* RandWriter.cpp - PS6
* Date: 11/16/2021
* created by: Wesley Gallo
*/

#include <utility>
#include <map>
#include <string>
	#include "RandWriter.h"

	RandWriter::RandWriter(string text, int k) {
	  srand(time(NULL));
	  this->k = k;
	  string texts = text;
	  for (int i = 0; i < k; i++)
	    texts.push_back(text[i]);
	  for (auto i : text)
	    if (ch.find(i) == string::npos)
	      ch.push_back(i);
	  for (int i = 0; i < 2; i++)
	    for (int ii = 0; ii < (signed)text.length(); ii++)
	      kgrams.insert(pair<string, int>(texts.substr(ii, k+i), 0));
	  for (int i = 0; i < 2; i++)
	    for (int ii = 0; ii < (signed)text.length(); ii++) {
	      map<string, int>::iterator itr = kgrams.find(texts.substr(ii, k+i));
	      kgrams[texts.substr(ii, k+i)] = itr->second+1;
	    }
	}

```

```
31.      int RandWriter::kOrder() {
32.        return k;
33.      }
34.
35.      int RandWriter::freq(string kgram) {
36.        if (kgram.length() != (unsigned)k)
37.          throw runtime_error("k-gram must be of length k.");
38.        map<string, int>::iterator itr = kgrams.find(kgram);
39.        if (itr == kgrams.end())
40.          return 0;
41.        return itr->second;
42.      }
43.
44.      int RandWriter::freq(string kgram, char c) {
45.        if (kgram.length() != (unsigned)k)
46.          throw runtime_error("k-gram must be of length k.");
47.        kgram.push_back(c);
48.        map<string, int>::iterator itr = kgrams.find(kgram);
49.        if (itr == kgrams.end())
50.          return 0;
51.        return itr->second;
52.      }
53.      char RandWriter::kRand(string kgram) {
54.        if (kgram.length() != (unsigned)k)
55.          throw runtime_error("kgram must be of length k.");
56.        map<string, int>::iterator itr = kgrams.find(kgram);
57.        if (itr == kgrams.end())
58.          throw runtime_error("The given kgram could not be found.");
59.        int r = rand()%freq(kgram); //NOLINT
60.        double x = 0;
61.        for (auto a : ch) {
62.          if((double)r/freq(kgram) < (double)freq(kgram,
   a)/freq(kgram)+x&&(double)freq(kgram, a)/freq(kgram) != 0) //NOLINT
63.            return a;
64.          x+=(double)freq(kgram, a)/freq(kgram); //NOLINT
65.        }
66.        return '-';
67.      }
68.      string RandWriter::generate(string kgram, int L) {
69.        if (kgram.length() != (unsigned)k)
70.          throw runtime_error("k-gram must be of length k.");
71.        string s = kgram;
72.        for (int a = 0; a < L-k; a++)
73.          s.push_back(kRand(s.substr(a, k)));
74.        return s;
75.      }
76.      ostream& operator<<(ostream &o, RandWriter &m) {
77.        o << "Order: " << m.k << endl;
78.        o << "Alphabet: " << m.ch << endl;
79.        o << "Frequencies of the k-grams and K+1-grams: " <<endl;
80.        map<string, int>::iterator itr;
81.        for (itr = m.kgrams.begin(); itr != m.kgrams.end(); itr++)
```

```
    o << itr->first << "\t" << itr->second << "\n";
  return o;
}
```

RandWriter.h

```
/**
* RandWriter.h - PS6
* Date: 11/16/2021
* created by: Wesley Gallo
*/

#ifndef RandWriter_HPP
#define RandWriter_HPP
#include <algorithm>
      #include <iostream>
      #include <map>
      #include <string>
      #include <stdexcept>
      #include <utility>
      #include <map>
      #include <string>
      using namespace std;
      class RandWriter {
       public:
        RandWriter(string text, int k);
        int kOrder();
        int freq(string kgram);
        int freq(string kgram, char c);
        char kRand(string kgram);
        string generate(string kgram, int L);
        friend ostream& operator<<(ostream &o, RandWriter &m);
       private:
        int k;
        map<string, int> kgrams;
        string ch;
      };
      #endif
```

test.cpp

```
/**
* Test.cpp - PS6
* Date: 11/16/2021
* created by: Wesley Gallo
*/

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

```
#include <string>
#include <exception>
#include <stdexcept>
#include "RandWriter.h"
BOOST_AUTO_TEST_CASE(test1) {
  BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 2));
  RandWriter rw("gagggagaggcgagaaa", 2);
  BOOST_REQUIRE(rw.kOrder() == 2);
  BOOST_REQUIRE(rw.freq("gg") == 3);
  BOOST_REQUIRE_THROW(rw.freq("y"), runtime_error);
  BOOST_REQUIRE(rw.freq("gg", 'g') == 1);
  BOOST_REQUIRE(rw.freq("gg", 'a') == 1);
  BOOST_REQUIRE(rw.freq("gg", 'c') == 1);
  BOOST_REQUIRE(rw.freq("gg", 'x') == 0);
}
BOOST_AUTO_TEST_CASE(test2) {
  BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 1));
  RandWriter rw("gagggagaggcgagaaa", 1);
  BOOST_REQUIRE(rw.kOrder() == 1);
  BOOST_REQUIRE_THROW(rw.freq(""), runtime_error);
  BOOST_REQUIRE_THROW(rw.freq("xy"), runtime_error);
  BOOST_REQUIRE(rw.freq("c") == 1);
  BOOST_REQUIRE(rw.freq("a", 'a') == 2);
  BOOST_REQUIRE(rw.freq("a", 'c') == 0);
  BOOST_REQUIRE(rw.freq("g", 'g') == 3);
  BOOST_REQUIRE_THROW(rw.freq(""), runtime_error);
  BOOST_REQUIRE_NO_THROW(rw.freq("y"));
  BOOST_REQUIRE_THROW(rw.freq("", 'g'), runtime_error);
  BOOST_REQUIRE_THROW(rw.freq("", 'x'), runtime_error);
  BOOST_REQUIRE_THROW(rw.freq("hwefw", 'g'), runtime_error);
  BOOST_REQUIRE_THROW(rw.kRand("j"), runtime_error);
  BOOST_REQUIRE_THROW(rw.kRand("o"), runtime_error);
  BOOST_REQUIRE_THROW(rw.kRand("tr"), runtime_error);
}
```

TextWriter.cpp

```
/**
* TextWriter.cpp - PS6
* Date: 11/16/2021
* created by: Wesley Gallo
*/

#include <string>
#include "RandWriter.h"
int main(int argc, const char* argv[]) {
        int k = stoi(argv[1]), l = stoi(argv[2]);
        string input = "", s;
        while (cin >> s) {
          input+=" "+s;
          s = "";
        }
```

```
16.         RandWriter rw(input, k);
17.         string output_string = rw.generate(input.substr(0, k), l);
18.         cout << "Output: ";
19.         for (int a = 0; a < l; a++)
20.           cout << output_string[a];
21.         cout << "\n";
22.         return 0;
23.       }
```

PS7: Kronos Time Clock or Introduction to Regular Expression Parsing

Assignment Discussion:

PS7 verified device boot up timing and analyzed the Kronos InTouch time clock log by using expressions to parse the file. The program also would note whether the startup was successful or not. This would gather information on what occurred in the device at the time of bootup. In the assignment we would scan logs given, and created a txt file that would be described each time the device was restarted, and giving the elapsed time.

Key Algorithms, data structures, or OO designs that were central to the assignment:

I used the Boost regex library to complete this assignment, and finding data in the device log. Expressions were used to find data, time, is the start up working.

What I learned:

The assignment taught me a lot about regex and using it in programming. It was interesting to enter a file and then parse it for data. Also while making the code more efficient in the process.

Screenshot(s) of Output:

device1_intouch.log.rpt

```
435369 (log.c.166) server started 2014-03-25 19:11:59 success 183000ms
436500 (log.c.166) server started 2014-03-25 19:29:59 success 165000ms
440719 (log.c.166) server started 2014-03-25 22:01:46 success 161000ms
440866 (log.c.166) server started 2014-03-26 12:47:42 success 167000ms
442094 (log.c.166) server started 2014-03-26 20:41:34 success 159000ms
443073 (log.c.166) server started 2014-03-27 14:09:01 success 161000ms
```


device1_intouch.log.rpt

```
498921 (log.c.166) server started 2014-03-11 15:42:26 success 162000ms
```

device3_intouch.log.rpt

```
31063 (log.c.166) server started 2014-01-26 09:55:07 success 177000ms
31274 (log.c.166) server started 2014-01-26 12:15:18 failure
31293 (log.c.166) server started 2014-01-26 14:02:39 success 165000ms
32623 (log.c.166) server started 2014-01-27 12:27:55 failure
32641 (log.c.166) server started 2014-01-27 12:30:23 failure
32656 (log.c.166) server started 2014-01-27 12:32:51 failure
32674 (log.c.166) server started 2014-01-27 12:35:19 failure
32693 (log.c.166) server started 2014-01-27 14:02:38 success 163000ms
33709 (log.c.166) server started 2014-01-28 12:44:17 failure
33725 (log.c.166) server started 2014-01-28 14:02:33 success 162000ms
34594 (log.c.166) server started 2014-01-29 12:43:07 failure
34613 (log.c.166) server started 2014-01-29 14:02:35 success 164000ms
37428 (log.c.166) server started 2014-01-30 12:43:05 failure
37447 (log.c.166) server started 2014-01-30 14:02:40 success 162000ms
38258 (log.c.166) server started 2014-01-31 14:02:33 success 163000ms
39150 (log.c.166) server started 2014-02-01 12:39:38 failure
39166 (log.c.166) server started 2014-02-01 12:42:07 failure
39182 (log.c.166) server started 2014-02-01 14:02:32 success 164000ms
40288 (log.c.166) server started 2014-02-02 14:02:39 success 172000ms
41615 (log.c.166) server started 2014-02-03 12:35:55 failure
41633 (log.c.166) server started 2014-02-03 12:38:22 failure
41648 (log.c.166) server started 2014-02-03 12:40:48 failure
41666 (log.c.166) server started 2014-02-03 12:43:17 failure
41684 (log.c.166) server started 2014-02-03 12:45:46 failure
41694 (log.c.166) server started 2014-02-03 14:02:34 success 164000ms
```

device4_intouch.log.rpt

```
4 (log.c.166) server started 2013-10-02 18:42:38 success 165000ms
747 (log.c.166) server started 2013-10-03 12:23:21 success 174000ms
1459 (log.c.166) server started 2013-10-04 16:20:03 success 183000ms
31848 (log.c.166) server started 2013-12-03 16:21:13 success 175000ms
32789 (log.c.166) server started 2013-12-04 21:50:27 success 150000ms
33145 (log.c.166) server started 2013-12-04 21:58:45 success 149000ms
33677 (log.c.166) server started 2013-12-04 22:21:03 success 148000ms
45295 (log.c.166) server started 2013-12-05 13:34:25 success 150000ms
45615 (log.c.166) server started 2013-12-05 14:12:25 success 148000ms
46117 (log.c.166) server started 2013-12-05 15:39:02 success 147000ms
46357 (log.c.166) server started 2013-12-05 20:20:24 success 150000ms
46792 (log.c.166) server started 2013-12-10 13:20:43 success 149000ms
47700 (log.c.166) server started 2013-12-10 19:40:58 success 177000ms
48100 (log.c.166) server started 2013-12-11 14:09:11 success 150000ms
48345 (log.c.166) server started 2013-12-11 14:17:49 success 177000ms
```

device5_intouch.log.rpt

```
31063 (log.c.166) server started 2014-01-26 09:55:07 success 177000ms
31274 (log.c.166) server started 2014-01-26 12:15:18 failure
31293 (log.c.166) server started 2014-01-26 14:02:39 success 165000ms
32623 (log.c.166) server started 2014-01-27 12:27:55 failure
32641 (log.c.166) server started 2014-01-27 12:30:23 failure
32656 (log.c.166) server started 2014-01-27 12:32:51 failure
32674 (log.c.166) server started 2014-01-27 12:35:19 failure
32693 (log.c.166) server started 2014-01-27 14:02:38 success 163000ms
33709 (log.c.166) server started 2014-01-28 12:44:17 failure
33725 (log.c.166) server started 2014-01-28 14:02:33 success 162000ms
34594 (log.c.166) server started 2014-01-29 12:43:07 failure
34613 (log.c.166) server started 2014-01-29 14:02:35 success 164000ms
37428 (log.c.166) server started 2014-01-30 12:43:05 failure
37447 (log.c.166) server started 2014-01-30 14:02:40 success 162000ms
38258 (log.c.166) server started 2014-01-31 14:02:33 success 163000ms
39150 (log.c.166) server started 2014-02-01 12:39:38 failure
39166 (log.c.166) server started 2014-02-01 12:42:07 failure
39182 (log.c.166) server started 2014-02-01 14:02:32 success 164000ms
40288 (log.c.166) server started 2014-02-02 14:02:39 success 172000ms
41615 (log.c.166) server started 2014-02-03 12:35:55 failure
41633 (log.c.166) server started 2014-02-03 12:38:22 failure
41648 (log.c.166) server started 2014-02-03 12:40:48 failure
41666 (log.c.166) server started 2014-02-03 12:43:17 failure
41684 (log.c.166) server started 2014-02-03 12:45:46 failure
41694 (log.c.166) server started 2014-02-03 14:02:34 success 164000ms
```

Project Source Code:

Makefile:

```
1. CC = g++
```

```
2. CFLAGS = -std=c++11 -c -g -Og -Wall -Werror -pedantic
3. OBJ = main.o
4. DEPS = main.cpp
5. LIBS = -lboost_regex -lboost_date_time
6. EXE = ps7
7.
8. all: $(OBJ)
9.       $(CC) $(OBJ) -o $(EXE) $(LIBS)
10.
11.      %.o: %.cpp $(DEPS)
12.            $(CC) $(CFLAGS) -o $@ $<
13.
14.      clean:
15.            rm $(OBJ) $(EXE)
```

main.cpp

```
1. /**
2. * Main.cpp - ps7
3. * Date: 12/3/2021
4. * created by: Wesley Gallo
5. */
6.
7.       #include <iostream>
8.       #include <string>
9.       #include <fstream>
10.      #include <boost/regex.hpp>
11.      #include "boost/date_time/posix_time/posix_time.hpp"
12.
13.      using std::cout;
14.      using std::cin;
15.      using std::endl;
16.      using std::string;
17.
18.      using boost::regex;
19.      using boost::smatch;
20.      using boost::regex_error;
21.
22.      using boost::gregorian::date;
23.      using boost::gregorian::from_simple_string;
24.      using boost::gregorian::date_period;
25.      using boost::gregorian::date_duration;
26.      using boost::posix_time::ptime;
27.      using boost::posix_time::time_duration;
28.
29.      int main(int argc, char **args)
30.      {
31.          if (argc != 2)
32.          {
33.              cout << "usage: ./ps7 [logfile]" << endl;
34.              exit(1);
35.          }
```

```
36.         string s, rs;
37.         regex e1;
38.         regex e2;
39.         bool flag = false;
40.         ptime t1, t2;
41.         string filename(args[1]);
42.         std::ifstream infile(filename);
43.         std::ofstream outfile(filename + ".rpt");
44.         if (!infile || !outfile)
45.         {
46.             cout << "open file error" << endl;
47.             exit(1);
48.         }
49.
50.         try
51.
52.         {
53.             e1 = regex(R"((.*): (\(log.c.166\) server started.*))");
54.             e2 = regex("(.*)\\.\\d*:INFO:oejs.AbstractConnector:Started "
55.                        "SelectChannelConnector@0.0.0.0:9080.*");
56.         }
57.         catch (regex_error &exc)
58.         {
59.             cout << "Regex constructor failed with code " << exc.code() << endl;
60.             exit(1);
61.         }
62.         int line_number = 1;
63.         string str;
64.         while (getline(infile, s))
65.         {
66.             if (regex_match(s, e1))
67.             {
68.                 smatch sm;
69.                 regex_match(s, sm, e1);
70.                 if (flag)
71.                 {
72.                     outfile << "failure" << endl;
73.                 }
74.                 flag = true;
75.                 t1 = ptime(boost::posix_time::time_from_string(sm[1]));
76.                 str = sm[2];
77.                 outfile << line_number << " (log.c.166) server started "
78.                         << sm[1] << " ";
79.             }
80.             if (regex_match(s, e2))
81.             {
82.                 smatch sm;
83.                 regex_match(s, sm, e2);
84.                 t2 = ptime(boost::posix_time::time_from_string(sm[1]));
85.                 outfile << "success " << (t2 - t1).total_milliseconds()
86.                         << "ms" << endl;
87.                 flag = false;
```

```
88.             }
89.             line_number++;
90.         }
91.         if (flag)
92.         {
93.             outfile << "failure" << endl;
94.         }
95.         infile.close();
96.         outfile.close();
97.         return 0;
98.     }
99.
```