Raja Ritika Reddy Buttreddy
COMP IV: Project Portfolio
Fall 2021

Contents:

Time to complete: 15 hours

PS0 Hello World with SFML

Description:

Our first Computing IV assignment was Hello World with SFML. This assignment's major purpose was to set up our Linux build environment and test the SFML audio/graphics library. This included installing Linux, either through a VirtualBox image or directly and executing some SFML sample programs to test SFML. To make the sample code perform anything interesting, we had to enhance it. In my implementation for this assignment, I imported a random picture I named "sprite.png" into the

program and made it respond to keystrokes. The picture would move up, left, down, and right in response to hitting the W, A, S, and D keys, and it would rotate with each input.

Key algorithms, Data structures and OO Designs used in this assignment:

Because this was my first project for Comp 4, I didn't utilize any algorithms to build the application. Mostly just basic programming concepts like loops, objects, and variables to keep the window open and the sprites moving to the keystrokes assigned to them. The utilization of SFML's window object and its render loop was a significant component of this project that required us to work with and use it. This loop would run if the window was active, checking every frame to see what events were detected by the built-in event handler. This event handler produces an Event object, which we can read and use to determine whether to update the sprite on the output screen. Following the completion of this check, the modified sprite is repainted, and the procedure is repeated. This was the main logic to the assignment, and it was easy to comprehend and implement.

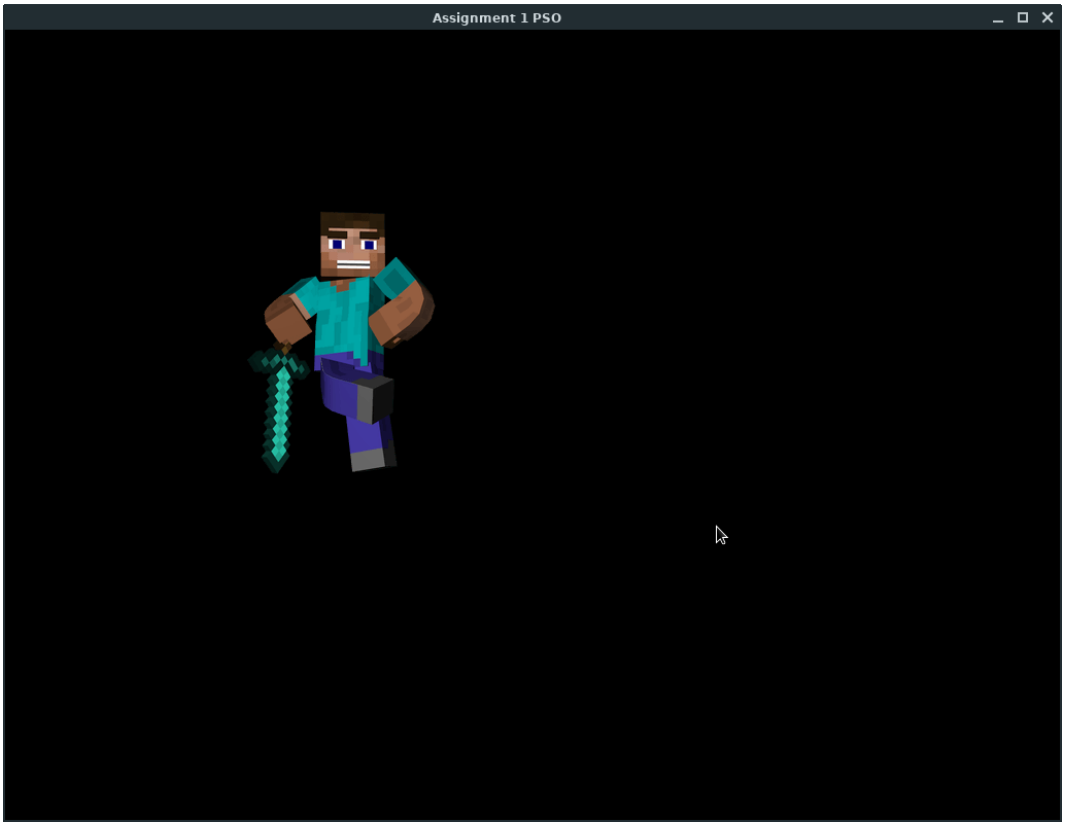
What I've Learned in this assignment:

I was new to SFML, so it was interesting to work with. I learnt how to utilize SFML on a basic level, how to show graphics in an SFML window, and even how to manipulate sprites using SFML's Keyboard module. Overall, it was a fun assignment to do.

Screenshots of Output:

Image 1 – Sprite.png

Image 2 – Screenshot.png



Source Code:

Note: I did not make a “Makefile” for this assignment. **main.cpp:**

```
1 // declaring
2 #include <SFML/Graphics.hpp> 3 class sprite {
4       private:
```

```
5       float x, y, velocity;
6       int screen_width, screen_height;
7       public:
8       float getPosX();
9       float getPosY();
10      void setPosX(float);
11      void setPosY(float);
12      void update();
13      sprite(float, float, int, int, int);
14      };
15
16      int main()
17      {
18      // max screen height
19      int max_height = 768, max_width = 1024;
20      int q = 0, p = 0;
21      bool ColourMode = false;
22
23      // creating a window
24      sf::RenderWindow window(sf::VideoMode(max_width, max_height), "Assignment 1 PSO");
25      window.setFramerateLimit(60);
26
27          // creating a texture
28          sf::Texture texture;
29          if (!texture.loadFromFile("sprite.png"))
30          return EXIT_FAILURE;
31
32      // creating a sprite from the texture
33      sf::Sprite sprite(texture);
34      sprite.setPosition(100.f, 100.f);
35
36      //creating arectangle for colour mode
37      sf::RectangleShape bcg(sf::Vector2f(max_width, max_height));
38      bcg.setPosition(0, 0);
39
40                  // Logic
41                  while (window.isOpen()){
42                  sf::Event event;
43                  while (window.pollEvent(event)){
44                  if (event.type == sf::Event::Closed)
45                  window.close();
46                  }
47
48              // Following Keystrokes
49              if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)){
50              if(!(sprite.getPosition().x <= 0))
51              sprite.move(sf::Vector2f(-10.f, 0.f));
52              }
53              else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)){
54              if(!(sprite.getPosition().x >= (max_width-sprite.getGlobalBounds().width)))
```

```
55              sprite.move(sf::Vector2f(10.f, 0.f));
56              }
57              else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up)){
58              if(!(sprite.getPosition().y <= 0))
59              sprite.move(sf::Vector2f(0.f, -10.f));
60
61             }
62             else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)){
63             if(!(sprite.getPosition().y >= (max_height-sprite.getGlobalBounds().height)))
64             sprite.move(sf::Vector2f(0.f, 10.f));
65             }
66             if (sf::Keyboard::isKeyPressed(sf::Keyboard::A))
67             ColourMode = true;
68             else if (sf::Keyboard::isKeyPressed(sf::Keyboard::B))
69             ColourMode = false;
70
71             // Colours Switch
72             window.clear();
73             if (ColourMode) {
74             if (p > 254) p = 0;
75             if (q > 254) q = 0;
76             bcg.setFillColor(sf::Color(p,q,q));
77             p += 10;
78             q++;
79             window.draw(bcg);
80             }
81
82         window.draw(sprite);
83         window.display();
84         }
85         return 0;
86         }
```

PS1 Linear Feedback Shift Register and Image Encoding

Description:

This assignment deals with creating a Linear Feedback Shift Register class to create pseudorandom bits and utilizing that register to provide encryption for digital photos. There were two parts to this assignment, i.e., Ps1(a) & Ps1(b). Ps1(a) included the first part of the assignment - This task requires us to use Linear Feedback Shift Register. This register shifts all bits left one position, then XORs the leftmost bit and the seed bit to fill the vacant space on the far-right side left after the shift left. The second part of this assignment Ps1(b) involved - constructing a program that reads a photo from the command line and then produces the same image, but encoded, using the LFSR class we constructed in

Ps1(a) (encrypted). The LFSR class was used to encode the picture by left shifting all the bits in the image, resulting in XOR encoding. We also needed to show the image in an SFML window and save the encrypted image to a file.

Key algorithms, Data structures and OO Designs used in this assignment:

The assignment demanded us to use vector of Booleans (data structure), with each Boolean representing a bit in the LFSR with a true/false, or 1/0 value. This was used to store the values of all the bits in the register, allowing me to perform vector operations like pop back and insert () to extract bits from the back of the register and put them in the front. We also used SFML components which were introduced in the first assignment like pictures, textures and sprites to read the file, encode it, and output the final encoded image to both the screen and disk.

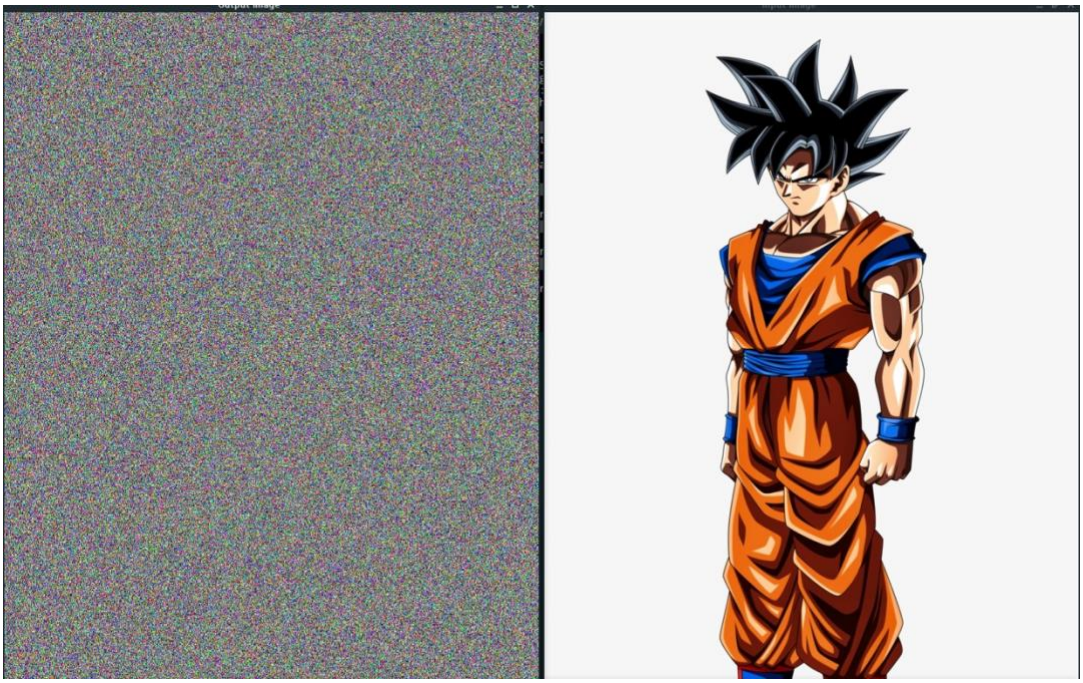
What I've Learned in this assignment:

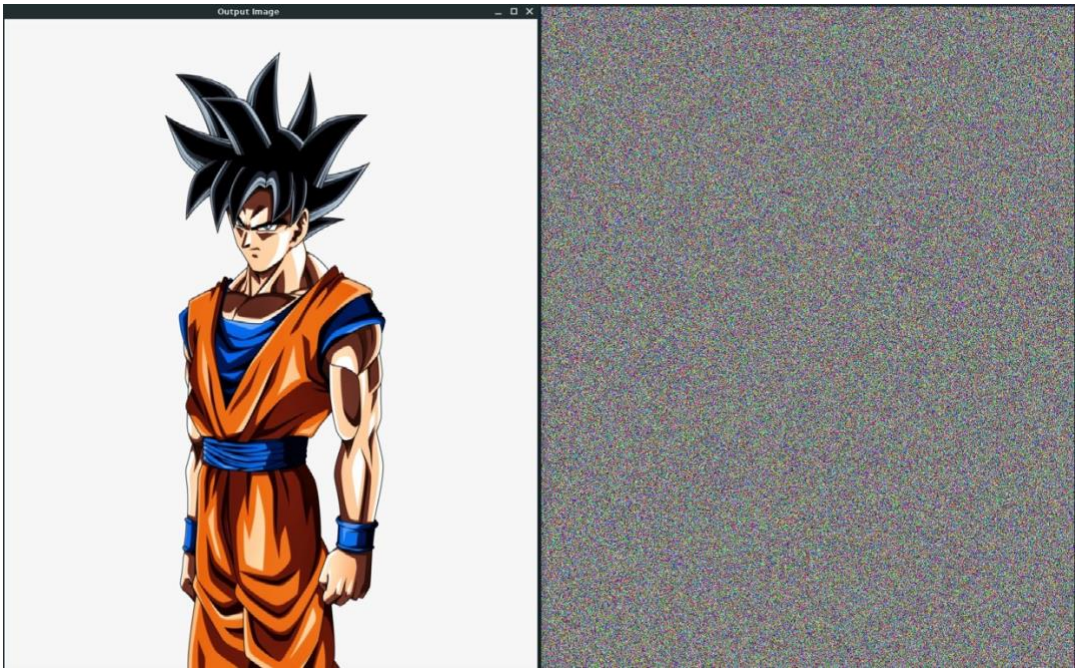
This assignment was a step above the first assignment we did which provided me with additional experience and gave me a broad understanding of bitwise operations, as well as the fundamentals of the Boost unit testing framework. Given that this project included a lot of bitwise shifting and the use of the XOR function, it helped me become more efficient when working with binary variables when coding. Finally, Ps1(a) & Ps1(b) required me to develop more familiarity with utilizing the command line to read input and output options. This assignment also helped me get more proficient with using SFML.

Screenshots of Output:

Image 1 – encoded.png Image

2 – decoded.png





Source Code:

Makefile:

```
1 # Makefile for ps1b
2
3 CC= g++
4 CFLAGS= -Wall -Werror -ansi -pedantic
5 SFMLFLAGS= -lsfml-graphics -lsfml-window -lsfml-system
6
7 # Make ps1b
```

```
8 all: PhotoMagic
9
10     # ps1b executable
11     PhotoMagic:  PhotoMagic.o FibLFSR.o
12     $(CC) PhotoMagic.o FibLFSR.o -o PhotoMagic $(SFMLFLAGS)
13
14     # object files
15     PhotoMagic.o: PhotoMagic.cpp FibLFSR.h
16     $(CC) -c PhotoMagic.cpp FibLFSR.h $(CFLAGS)
17
18     LFSR.o:       FibLFSR.cpp FibLFSR.h
19     $(CC) -c FibLFSR.cpp $(CFLAGS)
20
21 # Cleanup 22
clean:
23     rm *.o
24     rm PhotoMagic
```

test.cpp:

```
1 #include <iostream>
2 #include <string>
3
4 #include "FibLFSR.h"
5
6 #define BOOST_TEST_DYN_LINK
7 #define BOOST_TEST_MODULE Main
8 #include <boost/test/unit_test.hpp>
9
10 BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
11
12   FibLFSR l("1011011000110110");
13   BOOST_REQUIRE(l.step() == 0);
14   BOOST_REQUIRE(l.step() == 0);
15   BOOST_REQUIRE(l.step() == 0);
16   BOOST_REQUIRE(l.step() == 1);
17   BOOST_REQUIRE(l.step() == 1);
18   BOOST_REQUIRE(l.step() == 0);
19   BOOST_REQUIRE(l.step() == 0);
20   BOOST_REQUIRE(l.step() == 1);
21
22   FibLFSR l2("1011011000110110");
23   BOOST_REQUIRE(l2.generate(9) == 51);
24   }
25
26
27 BOOST_AUTO_TEST_CASE(TwentyBitsThreeTaps) {
28
29   FibLFSR l("10101001101010011010");
30   BOOST_REQUIRE(l.step() == 0);
31   BOOST_REQUIRE(l.step() == 1);
```

```
32   BOOST_REQUIRE(l.step() == 1);
33   BOOST_REQUIRE(l.step() == 1);
34
35   FibLFSR l2("10101001101010011010");
36   BOOST_REQUIRE(l2.generate(5) == 14);
37   }
38
39   BOOST_AUTO_TEST_CASE(SixteenBitsThreeTaps2) {
40   FibLFSR l("1011011000110111");
41   BOOST_REQUIRE(l.step() == 0);
42   BOOST_REQUIRE(l.step() == 0);
43   BOOST_REQUIRE(l.step() == 0);
44   BOOST_REQUIRE(l.step() == 1);
45   BOOST_REQUIRE(l.step() == 1);
46   BOOST_REQUIRE(l.step() == 0);
47
48   FibLFSR l2("1011011000110111");
49   BOOST_REQUIRE(l2.generate(5) == 3);
50   }
```

Photomagic.cpp

```
1 #include <iostream>
2 #include <string>
3 #include <sstream>
4 #include <SFML/System.hpp>
5 #include <SFML/Window.hpp>
6 #include <SFML/Graphics.hpp>
7 #include "FibLFSR.h"
8 std::string password(std::string st){
9
10           int convert = 53;
11           int len = st.length();
12           for(int i = 0; i < len; i++){
13           convert = convert ^ st[i];
14           convert *= convert;
15           }
16
17           std::string binary;
18           while(convert != 0) {
19           binary = (convert % 2 == 0 ? "0" : "1") + binary;
20           convert /= 2;
21           }
22
23     return binary;
24     }
25
26       void transform( sf::Image& img, FibLFSR* bit_generator) {
27       sf::Vector2u imgsize = img.getSize();
28       sf::Color p;
```

```
29      for(int x = 0; x < (signed)imgsize.x; x++)
30      {
31      for(int y = 0; y < (signed)imgsize.y; y++)
32      {
33      p = img.getPixel(x, y);
34      p.r = p.r ^ bit_generator -> generate(8);
35      p.g = p.g ^ bit_generator -> generate(8);
36      p.b = p.b ^ bit_generator -> generate(8);
37      img.setPixel(x, y, p);
38      }
39      }
40      }
41
42    int main(int argc, char* argv[]){
43    if(argc != 4){
44    std::cout << "Bad Input, Usage: ./PhotoMagic <inputfilename> <outputfilename>
      <seed>\n";
45    return -1;
46    }
47    std::string input_fname(argv[1]);
48    std::string output_fname(argv[2]);
49    std::string givenpassword(argv[3]);
50
51  std::string seed = password(givenpassword);
52
53    FibLFSR bit_generator(seed);
54    sf::Image input_image;
55    if (!input_image.loadFromFile(input_fname))
56    {
57    return -1;
58    }
59
60    sf::Image output_image;
61    if (!output_image.loadFromFile(input_fname))
62    {
63    return -1;
64    }
65    sf::Vector2u imgsize = input_image.getSize();
66    sf::RenderWindow input_window(sf::VideoMode(imgsize.x, imgsize.y), "Input Image"); 67
      sf::RenderWindow output_window(sf::VideoMode(imgsize.x, imgsize.y), "Output Image");
68  sf::Texture in_texture, out_texture;
69  in_texture.loadFromImage(input_image);
70
71  transform(input_image, &bit_generator);
72
73    out_texture.loadFromImage(input_image);
74    sf::Sprite in_sprite, out_sprite;
75    in_sprite.setTexture(in_texture);
76    out_sprite.setTexture(out_texture);
77    while (input_window.isOpen() && output_window.isOpen())
78    {
```

```
79     sf::Event event;
80
81         while (input_window.pollEvent(event))
82         {
83         if (event.type == sf::Event::Closed)
84         {
85         input_window.close();
86         }
87         }
88
89     while (output_window.pollEvent(event)) 90
{
91         if (event.type == sf::Event::Closed)
92         {
93         output_window.close();
94         }
95         }
96
97     input_window.clear();
98     input_window.draw(in_sprite);
99     input_window.display();
100
101     output_window.clear();
102     output_window.draw(out_sprite);
103     output_window.display();
104     }
105     if (!input_image.saveToFile(output_fname))
106     {
107     return -1;
108     }
109
110   return 0;
111   }
```

FibLFSR.h

```
1 #include <iostream>
2
3          class FibLFSR {
4          public:
5          FibLFSR(std::string seed);
6          int step();
7          int generate(int k);
8          friend std::ostream& operator<< (std::ostream& out, const FibLFSR& fibLFSR);
9          private:
10         std::string reg;
11         int getBit(char a);
12         int xorOP(int a, int b);
13
14 };
```

FibLFSR.cpp

```
1 #include "FibLFSR.h"
2 #include <string>
3 #include <math.h>
4
5     std::ostream& operator<< (std::ostream& out, const FibLFSR& fibLFSR) {
6     out << fibLFSR.reg;
7     return out;
8     }
9
10     int FibLFSR::getBit(char a) {
11     if (a == '1') return 1;
12     else if (a == '0') return 0;
13     else return -1;
14     }
15
16     FibLFSR::FibLFSR(std::string seed){
17     reg = seed;
18     }
19
20     int FibLFSR::xorOP(int a, int b){
21     return a != b;
22     }
23
24     // TAPS AT 13, 12, 10
25     int FibLFSR::step() {
26     // storage for the new register after shifting
27     std::string new_reg = reg.substr(1);
28
29     // taps, XORing
30     // NOT EQUAL = 1; EQUAL = 0
31     int tapval = xorOP(reg[0], reg[2]);
32     tapval = xorOP(tapval, getBit(reg[3]));
33     tapval = xorOP(tapval, getBit(reg[5]));
34
35     FibLFSR::reg = new_reg;
36     FibLFSR::reg += std::to_string(tapval);
37     return tapval;
38     }
39
40         int FibLFSR::generate(int k) {
41         int res = 0;
42         for(int i = 0; i < k; i++){
43         int x = step();
44         res = (res * 2) + x;
45         }
46         return res;
47         }
```

PS2 N-Body Simulation

Description:

The assignment was to simulate movement of celestial bodies in 2D plane. There were two parts of the assignment involved in this. The first part Ps2(a) was - mostly concerned with reading a file through standard I/O and utilizing the data from that file to populate sprites (displaying the numerous planets) in the proper position in an SFML window. The second part Ps2(b) involved generating a static world in the previous part, here we made the cosmos move and respond to Newton's law of universal gravity and Newton's second rule of motion.

Key algorithms, Data structures and OO Designs used in this assignment:

We used a few essential C++ concepts for this project. The first was to read a file into standard I/O using the command line operator. The CelestialBody class held the key data for each celestial body, such as its location and velocity, while the Universe class oversaw managing and updating the position of each celestial body during the simulation. The primary principles for this project are related to physics. They are as follows: - Newton's law of universal gravitation, The superposition principle and the second law of motion as stated by Newton.

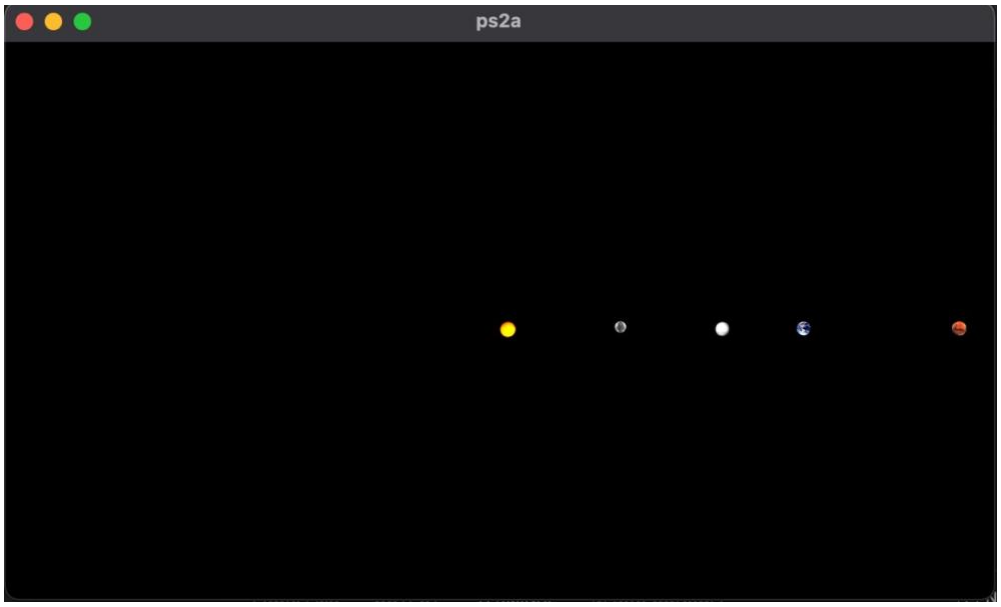
What I've Learned in this assignment:

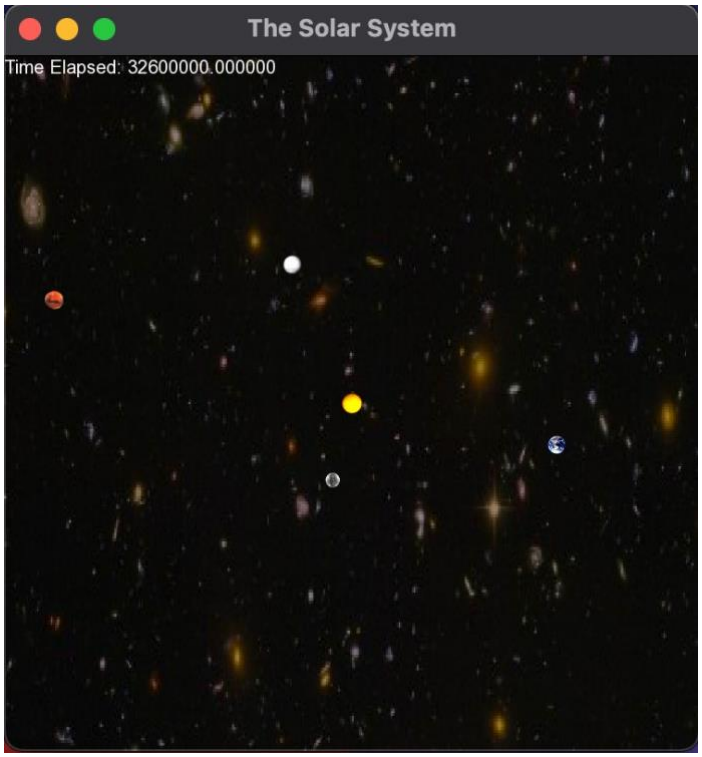
In this project, I learnt a little physics and how to apply various equations in a software. When you've successfully implemented them, you'll have a wonderful simulation of the cosmos. I also learned how to use smart pointers and it was a very helpful feature within the programming language. By utilizing static member variables to represent the universe's center and radius, just one copy of those values is shared across all CelestialBody instances, as opposed to having each CelestialBody retain a vector and float with the same value. The use of smart pointers in addition to developing the physics section, I learnt how to play music using SFML's audio library (extra point).

Screenshots of Output:

Image 1 – Ps2(a)screenshot.png

Image 2 – Ps2(b)screenshot.png





Source Code:

Makefile:

```
1      CC=g++
2      SFMLFLAGS=-lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system
3      CFLAGS=-g -Wall -ansi -pedantic -std=c++14
4      OBJ=main.o Universe.o CelestialBody.o
5      all: $(OBJ)
6      $(CC) $(CFLAGS) -o NBody $(OBJ) $(SFMLFLAGS)
7      main.o: main.cpp
8      $(CC) $(CFLAGS) -c main.cpp
9      Universe.o: Universe.cpp Universe.hpp
10     $(CC) $(CFLAGS) -c Universe.cpp
11     CelestialBody.o: CelestialBody.cpp CelestialBody.hpp 12  $(CC) $(CFLAGS) -c
       CelestialBody.cpp 13  clean:
14  -@rm -rf *.o 2>/dev/null || true
```

main.cpp:

```
1                          #include <iostream>
2                          #include <fstream>
3                          #include "Universe.hpp"
4                          #include "CelestialBody.hpp"
5                          #include <string>
6                          #include <exception>
7                          #include <vector>
8                          #include <SFML/Graphics.hpp>
9                          #include <SFML/Audio.hpp>
10                         #define widthwin 750
```

```
11                          #define fpswin 30
12                          #define heightwin 750
13                          using namespace std;
14                          int main(int argc, char* argv[])
15                          {
16                          if(argc < 2)
17                          {
18                          cout << "No arg" << endl;
19                          return 1;
20                          }
21                          float mtime, ctime;
22                          float time_change;
23                          ctime = 0;
24                          try {
25                          mtime = stod(argv[1]);
26                          time_change = stod(argv[2]);
27                          }
28                          catch(exception e)
29                          {
30                          cout << "Error" << endl;
31                          return 1;
32                          }
33                          sf::Vector2f centerUniverse{widthwin / 2, heightwin / 2};
34                          Universe solarSystem(centerUniverse);
35                          cin >> solarSystem;
36                          sf::RenderWindow window(sf::VideoMode(widthwin, heightwin), "The
                            Solar System");
37                          sf::Texture spaceTextures;
38                          sf::Font sFont;
39                          sf::SoundBuffer buffer;
40                          window.setFramerateLimit(fpswin);
41                          if(!spaceTextures.loadFromFile("spacebackground.png")) {
42                          throw FileNotFoundException();
43                          cout << "No background image selected" << endl;
44                          }
45                          sf::Sprite spaceBackground(spaceTextures);
46                          spaceBackground.setScale(static_cast<float>(widthwin) /
                            spaceTextures.get-
Size().x ,static_cast<float>(heightwin) / spaceTextures.getSize().y);
47                          if(!sFont.loadFromFile("font.ttf"))
48                          {
49                          throw FileNotFoundException();
50                          }
51                          if(!buffer.loadFromFile("pinkpanther.wav"))
52                          {
53                          throw FileNotFoundException();
54                          }
55                          sf::Sound sound(buffer);
56                          sf::Text timeElapsed{"Time Elapsed: " + to_string(ctime), sFont};
57                          timeElapsed.setPosition(0,0);
58                          timeElapsed.setCharacterSize(20);
```

```
59                          timeElapsed.setOutlineColor(sf::Color::White);
60                          while(window.isOpen())
61                          {
62                          sf::Event event;
63                          while(window.pollEvent(event))
64                          {
65                          sound.play();
66                          if (event.type == sf::Event::Closed ||
67                          sf::Keyboard::isKeyPressed(sf::Keyboard::Escape)) 68 {
69                                  ofstream result;
70                                  result.open("output.txt");
71                                  result << solarSystem;
72                                  result.close();
73                                  window.close();
74                                  }
75                                  }
76                                  if (ctime < mtime) {
77                                  window.clear();
78                                  window.draw(spaceBackground);
79                                  for ( const auto &p : solarSystem.getBodies() )
80                                  window.draw(*p);
81                                  window.draw(timeElapsed);
82                                  window.display();
83                                  solarSystem.step(time_change);
84                                  ctime += time_change;
85                                  timeElapsed.setString("Time Elapsed: " +
                                    to_string(ctime)); 86                  }
87                             else
88                             {
89                             window.setFramerateLimit(0);
90                             } 91          }
92          return 0;
93          }
```

CelestialBody.hpp:

```
1           #ifndef CELESTIAL_BODY_HPP_
2           #define CELESTIAL_BODY_HPP_
3           #include <iostream>
4           #include <string>
5           #include <exception>
6           #include <iomanip>
7           #include <SFML/Graphics.hpp>
8           using namespace std;
9           struct FileNotFoundException : public exception {
10          const char * what() const noexcept {
11          return "Can't find file!";
12          }
13          };
14          class CelestialBody : public sf::Drawable {
```

```
15          public:
16          CelestialBody() {}
17          CelestialBody(sf::Vector2f iPosition, sf::Vector2f iVelocity,float iMass,
            std::string iImageRef);
18          inline float getMass() { return mass; }
19          inline sf::Vector2f getPosition() { return position; }
20          inline void setPosition(sf::Vector2f nPosition) { position = nPosition; } 21
            inline sf::Vector2f getVelocity() { return velocity; }
22          inline void setVelocity(sf::Vector2f nVelocity) { velocity = nVelocity; }
23          static void createUniverse(sf::Vector2f iCenterUniverse, float iRadiusUniverse);
24          void spriteUpdate();
25          ~CelestialBody() { delete sprite_textures; };
26          friend ostream& operator<<(ostream &out, const CelestialBody& cb);
27          private:
28          sf::Vector2f position;
29          sf::Vector2f velocity;
30          float mass;
31          string imageRef;
32          sf::Sprite sprite;
33          sf::Texture* sprite_textures;
34          virtual void draw(sf::RenderTarget& rendTarget,
35          sf::RenderStates rendStates) const;
36          static sf::Vector2f centerUniverse;
37          static float radiusUniverse;
38          };
39          #endif
```

CelestialBody.cpp:

```
1           #include "CelestialBody.hpp"
2           using namespace std;
3           sf::Vector2f CelestialBody::centerUniverse{0,0};
4           float CelestialBody::radiusUniverse = 0;
5           CelestialBody::CelestialBody(sf::Vector2f iPosition, sf::Vector2f iVelocity,
            float iMass, string iImageRef) {
6           mass = iMass;
7           position = iPosition;
8           velocity = iVelocity;
9           imageRef = iImageRef;
10          sprite_textures = new sf::Texture;
11          if(!sprite_textures->loadFromFile(imageRef)) {
12          throw FileNotFoundException(); 13          }
14                  sprite = sf::Sprite(*sprite_textures);
15                  sprite.setOrigin(sprite_textures->getSize().x / 2,
16                  sprite_textures->getSize().y / 2);
17                  spriteUpdate();
18                  }
19                  void CelestialBody::createUniverse(sf::Vector2f iCenterUniverse,
20                  float iRadiusUniverse) {
21                  centerUniverse = iCenterUniverse;
```

```
22                  radiusUniverse = iRadiusUniverse;
23                  }
24                  void CelestialBody::spriteUpdate() {
25                  sf::Vector2f sprite_position{position.x / radiusUniverse
                    *centerUniverse.x + centerUniverse.x,position.y / radiusUniverse *
                    centerUniverse.y + centerUniverse.y};
26                  sprite.setPosition(sprite_position);
27                  }
28                  void CelestialBody::draw(sf::RenderTarget& rendTarget, sf::RenderStates
                    rendStates) const {
29                  rendTarget.draw(sprite, rendStates);
30                  }
31                  ostream& operator<<(ostream &out, const CelestialBody& cb) {
32                  out.setf(ios_base::scientific);
33                  out << setprecision(4) << left;
34                  out << setw(12) << cb.position.x << setw(12) << cb.position.y << setw(12)
35                  << cb.velocity.x << setw(12) << cb.velocity.y << setw(12)
36                  << cb.mass << right << setw(12) << cb.imageRef;
37                  out.unsetf(ios_base::scientific);
38                  return out;
39                  }
```

Universe.hpp:

```
1  #ifndef UNIVERSE_HPP_
2  #define UNIVERSE_HPP_
3  #include <iostream>
4  #include <cmath>
5  #include "CelestialBody.hpp"
6  #include <SFML/Graphics.hpp>
7  #include <string>
8  #include <vector>
9  using namespace std; 10  class Universe {
11          public:
12          Universe() {}
13          Universe(sf::Vector2f iCenter) : center(iCenter) {}
14          inline const vector<unique_ptr<CelestialBody>>&getBodies() const { return
            celBodies; }
15          friend istream& operator>>(istream& in, Universe& u);
16          friend ostream& operator<<(ostream& out, const Universe& u); 17          void
            step(float seconds);
18          private:
19          sf::Vector2f center;
20          float radius;
21          vector<unique_ptr<CelestialBody>> celBodies;
22          };
23          # endif
```

Universe.cpp:

```
1                                   #include "Universe.hpp"
2                                   using namespace std;
3                                   std::istream& operator>>(std::istream& in, Universe& u)
4                                   {
5                                   int numBodies;
6                                   in >> numBodies >> u.radius;
7                                   CelestialBody::createUniverse(u.center, u.radius);
8                                   for(int i = 0; i < numBodies; i++) {
9                                   float xPosition, yPosition, xVelocity, yVelocity, mass;
10                                  string imageRef;
11                                  in >> xPosition >> yPosition >> xVelocity >> yVelocity >>
                                    mass >>imageRef;
12                                  u.celBodies.push_back(
13                                  make_unique<CelestialBody>(sf::Vector2f(xPosition,
                                    yPosition),
14                                  sf::Vector2f(xVelocity, yVelocity),mass, imageRef));
15                                  }
16                                  return in;
17                                  }
18                                  ostream& operator<<(ostream& out, const Universe& u)
19                                  {
20                                  out << u.celBodies.size() << endl;
21                                  out << u.radius << endl;
22                                  for(const auto &b : u.celBodies) out << (*b) << endl;
23                                  return out;
24                                  }
25                                  void Universe::step(float seconds)
26                                  {
27                                  auto getNetForce = [&](size_t planetIndex) ->
                                    sf::Vector2f
28                                  {
29                                  sf::Vector2f netForce;
30                                  for(size_t i = 0; i < celBodies.size(); i++)
31                                  {
32                                  if(i != planetIndex)
33                                  {
34                                  sf::Vector2f position_change = celBodies[i]-
                                    >getPosition() - celBodies[planetIndex]->getPosition();
35                                  float planetDistance = hypot(position_change.x,
                                    position_change.y);
36                                  float scaleForce =(6.67430e-11 * celBodies[planetIndex]-
>getMass() * celBodies[i]->getMass()) /pow(planetDistance, 2);
37                                  sf::Vector2f force_xy =
38                                  {
39                                  scaleForce * (position_change.x /planetDistance), 40
                                    scaleForce * (position_change.y /planetDistance)
41                                  };
42                                  netForce += force_xy;
43                                  }
44                                  }
45                                  return netForce;
```

```
46                                  };
47                                  vector<sf::Vector2f> rPosition, rVelocity;
48                                  for(size_t i = 0; i < celBodies.size(); i++)  49
                                    {
50                  sf::Vector2f netForce = getNetForce(i);
51                  sf::Vector2f planetAccel =
52                  {
53                  netForce.x / celBodies[i]->getMass(), 54
                    netForce.y / celBodies[i]->getMass()
55                  };
56                  sf::Vector2f velocity =
57                  {
58                  celBodies[i]->getVelocity().x + seconds * planetAccel.x, 59
                    celBodies[i]->getVelocity().y + seconds * planetAccel.y
60                          };
61                          sf::Vector2f position =
62                          {
63                          celBodies[i]->getPosition().x + seconds * velocity.x,
64                          celBodies[i]->getPosition().y + seconds * velocity.y
65                          };
66                          rVelocity.push_back(velocity);
67                          rPosition.push_back(position);
68                          }
69                          for(size_t i = 0; i < celBodies.size(); i++) {
70                          celBodies[i]->setVelocity(rVelocity[i]);
71                          celBodies[i]->setPosition(rPosition[i]);
72                          celBodies[i]->spriteUpdate();
73                          }
74                          }
```

PS3 Recursive Graphics

Description:

In this assignment we write a program that plots a Triangle Fractal (Serpinski triangle). The goal of this assignment was to recursively draw a Serpinski triangle using the SFML library. A Serpinski triangle is created by drawing a base triangle, and then drawing one additional triangle on each side. The process is then repeated until the triangle is complete. The use of recursive functions and data structures are implemented, also some math to calculate the coordinates for the triangles and then used "fTree" to draw the triangles. Our job in this assignment is to create a program TFractal.cpp that includes a recursive function fTree () as well as a main () method that invokes the recursive function. Our program must accept the following command-line arguments: L and N: L length of the base equilateral triangle's side (double) and N is the recursion depth (int).

Key algorithms, Data structures and OO Designs used in this assignment:

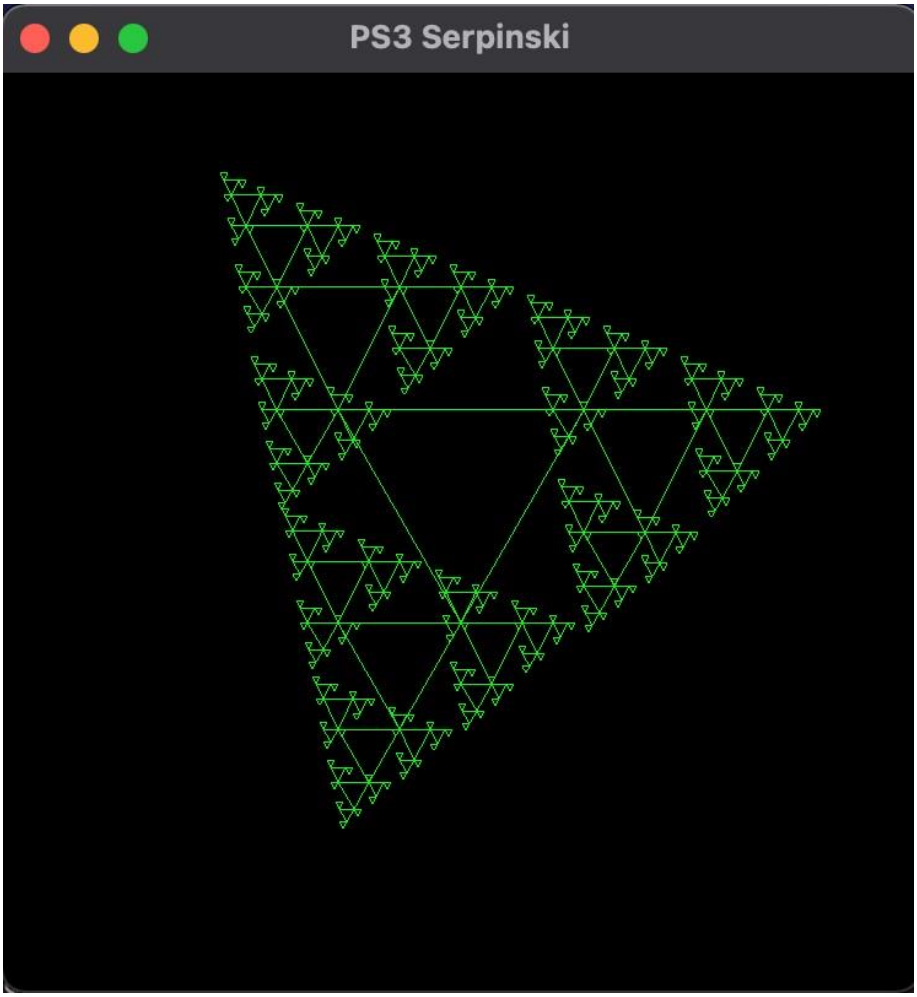
Basic OO concepts, classes, linked lists, and many other base concepts are used. The use of LinkedList is very crucial as the dependency of the triangles are on it to iterate 3 times and recursively repeat it and so on. I implemented green with the RGB values (3,253,53) to create the triangle (extra point).

What I've Learned in this assignment:

The use of Pythagoras theorem in software and how recursive functions work in an effective way. I also learned how to implement recursion in SFML and a better understanding of pointers. I understood why and how cpplint is useful in programming languages as it was asked in this assignment. How to implement SFML graphics with programs in an efficient way. This was an overall challenging assignment compared to the starting two. I learned how to draw shapes using the SFML library and gained exposure to an advanced application of recursion too.

Screenshots of Output:

Image 1 – Screenshot.png



Source code:

Makefile:

```
1 #ps3 Make file
2
3 compiler= g++
4 cppFlags= -Wall -Werror -std=c++0x -pedantic
5 SFMLFlags= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
6
7
8 all: main
9
10     main:  TFractal.o Triangle.o
11     $(compiler) TFractal.o Triangle.o -o main $(SFMLFlags)
12
13     TFractal.o:   TFractal.cpp Triangle.h Triangle.h
14     $(compiler) -c TFractal.cpp Triangle.h $(cppFlags)
15
16 main.o: Triangle.cpp Triangle.h
17 $(compiler) -c Triangle.cpp Triangle.h $(cppFlags) 18  19 clean:
20     rm *.o
21     rm main 22  rm *.gch
```

Tfractal.cpp:

```
1 #include "Triangle.h"
2 #include <vector>
3 #include <iostream>
4 #include <random>
5
6 #define height_of_screen 750
7 #define width_of_screen 750
8 #define DEF 0.867
9
10         void fTree(Triangle triangle, int depth, int length, sf::RenderWindow *window) {
11         if(depth <= 0)
12         return;
13
14     std::cout << "calculating - depth: " << depth << std::endl;
15
16     triangle.bottom = new Triangle(sf::Vector2f(triangle.p3.x - length, triangle.p3.y),
triangle.p3, sf::Vector2f(triangle.p3.x - length/2, triangle.p3.y + (length * DEF))); 17
triangle.left = new Triangle(sf::Vector2f(triangle.p1.x - length/2, triangle.p1.y -
length), sf::Vector2f(triangle.p1.x + length/2, triangle.p1.y - length), triangle.p1);
18     triangle.right = new Triangle(triangle.p2, sf::Vector2f(triangle.p2.x + length,
triangle.p2.y), sf::Vector2f(triangle.p2.x + length/2, triangle.p2.y + length)); 19
20     window -> draw(*triangle.bottom);
```

```
21      window -> draw(*triangle.left);
22      window -> draw(*triangle.right);
23
24      depth--;
25
26      fTree(*triangle.left, depth, length/2, window);
27      fTree(*triangle.right, depth, length/2, window);
28      fTree(*triangle.bottom, depth, length/2, window);
29      }
30
31      int main(int argc, char* argv[]) {
32
33          if (argc!= 3) {
34          std::cout << "[length] [depth]\n";
35          return -1;
36          } 37
38      sf::RenderWindow window(sf::VideoMode(width_of_screen, height_of_screen), "PS3
        Serpinski");
39      window.setFramerateLimit(60);
40
41      int length = atoi(argv[1]);
42      int depth = atoi(argv[2]);
43
44      float x = width_of_screen/2 - length/2;
45      float y = height_of_screen/2 - length/2;
46
47
48      Triangle initialTriangle(sf::Vector2f(x, y), sf::Vector2f(x + length, y),
sf::Vector2f(x + length/2, y + (length * DEF)));
49
50      window.draw(initialTriangle);
51      fTree(initialTriangle, depth, length/2, &window);
52
53
54      while (window.isOpen()){
55
56                  sf::Event event;
57                  while (window.pollEvent(event)){
58                  if (event.type == sf::Event::Closed)
59                  window.close();
60                  }
61
62          window.display();
63          }
64          }
```

Triangle.cpp:

```
1 #include "Triangle.h"
2
3      Triangle::Triangle(sf::Vector2f p, sf::Vector2f q, sf::Vector2f r) {
4      p1 = p;
```

```
5     p2 = q;
6     p3 = r;
7
8      bottom = NULL;
9      left = NULL;
10     right = NULL;
11     } 12
13         void Triangle::draw(sf::RenderTarget &target, sf::RenderStates states) const {
14         sf::Vertex line[] = {
15         sf::Vertex(p1, sf::Color(3,253,53)),
16         sf::Vertex(p2, sf::Color(3,253,53)),
17         };
18
19     target.draw(line, 2, sf::Lines);
20
21     line[0] = sf::Vertex(p2, sf::Color(3,253,53));
22     line[1] = sf::Vertex(p3, sf::Color(3,253,53));
23     target.draw(line, 2, sf::Lines);
24
25     line[0] = sf::Vertex(p1, sf::Color(3,253,53));
26     line[1] = sf::Vertex(p3, sf::Color(3,253,53));
27     target.draw(line, 2, sf::Lines);
28     }
```

Triangle.h:

```
1 #include <SFML/System.hpp>
2 #include <SFML/Window.hpp>
3 #include <SFML/Graphics.hpp>
4
5      class Triangle: public sf::Drawable {
6      public:
7      Triangle(sf::Vector2f, sf::Vector2f, sf::Vector2f);
8      sf::Vector2f p1, p2, p3;
9      Triangle *left, *right, *bottom;
10     int length;
11
12     private:
13     void draw (sf::RenderTarget &target, sf::RenderStates states) const; 14 };
```

PS4 Synthesizing a Plucked String Sound

Description:

The assignment was to create a software that uses the Karplus-Strong algorithm to imitate plucking a guitar string. This method was pivotal in the development of physically modeled sound synthesis (the use of a physical description of a musical instrument to generate sound electronically). The

assignment is split into two parts Ps4(a) and Ps4(b). The main task was to create a program that simulated the generated sound. A ring buffer has been implemented for this purpose (Circular Buffer) is a fixed size queue that can be filled with random values. Boost Unit Testing Framework Made sure that the Circular Buffer class created is complete, correct, and complete. The simulation itself supports 37 keys and can be played with the keys on your computer keyboard.

Key algorithms, Data structures and OO Designs used in this assignment:

This assignment is solved using a circular buffer to generate a plucked string and it is also tested using unit and boost tests. In the implementation, we use unit testing and exceptions. Key elements such as capacity, enqueue and dequeue are used to implement the queue. The main data structure used is a "QUEUE". First in first out mechanism is implemented where the first entered element is removed first unlike stack. The use of vectors and, I implemented lambda expression in the code.

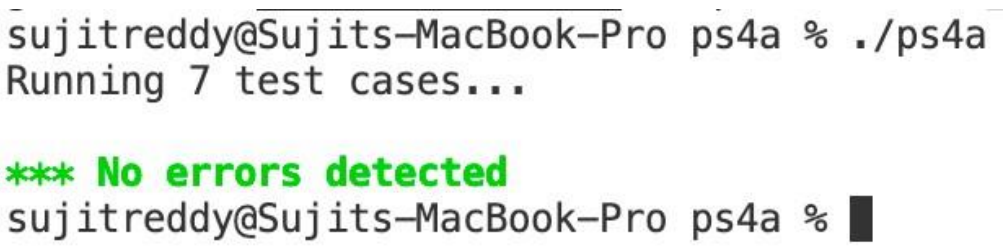
What I've Learned in this assignment:

Learned about cpplint. A deeper understanding of space and time complexity. In addition, I developed a better grasp of how program store and represent sound, as well as how algorithms may be used to construct the data that represents that sound. I also understood the two primary components that make the Karplus-Strong algorithm work - the ring buffer feedback mechanism and the averaging operation. The implementation of queue in a advance level.

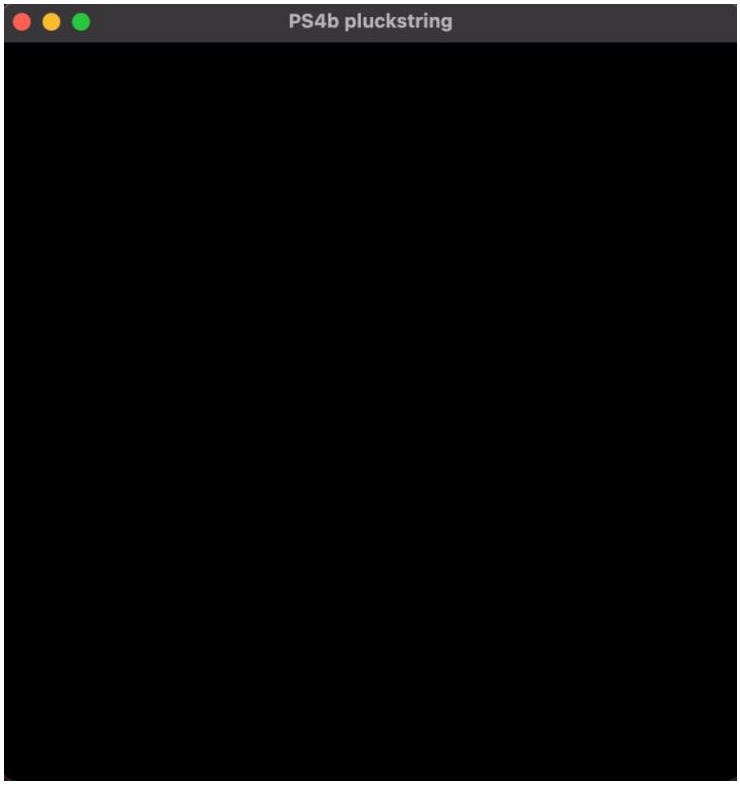
Screenshots of Output:

Image 1 – Running of ps4(a) screenshot

Image 2 – Ps4b screenshot – It plays sound responding to the keystrokes



It is outputting sound



Source code:

Makefile:

```
1 # ps4 makefile
2 compiler= g++
3 flags= -g -std=c++11
4 SFMLFlags= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
5
6 all: KSGuitarSim
7
8 KSGuitarSim: CircularBuffer.o StringSound.o KSGuitarSim.o
9 $(compiler) KSGuitarSim.o CircularBuffer.o StringSound.o -o KSGuitarSim $(SFMLFlags) 10
11     KSGuitarSim.o: KSGuitarSim.cpp
12     $(compiler) -c KSGuitarSim.cpp $(flags)
13
14 StringSound.o: StringSound.h StringSound.cpp
15 $(compiler) -c StringSound.h StringSound.cpp $(flags) 16
17     CircularBuffer.o: CircularBuffer.cpp CircularBuffer.h
18     $(compiler) -c CircularBuffer.cpp CircularBuffer.h $(flags)
19
20 test.o: test.cpp CircularBuffer.h
21 $(compiler) -c test.cpp CircularBuffer.h $(flags) 22  23 clean:
```

```
24     rm *.o
25     rm *.gch
26     rm KSGuitarSim
27     rm *.out
```

test.cpp:

```
1 /*
2 * Copyright 2021 Raja Ritika Reddy Buttreddy(01987338) 3  * All rights
  reserved.
4  */
5  #define BOOST_TEST_DYN_LINK
6  #define BOOST_TEST_MODULE Main
7  #include <boost/test/unit_test.hpp>
8
9 #include "CircularBuffer.h"
10
11     BOOST_AUTO_TEST_CASE(constructor) {
12     BOOST_REQUIRE_THROW(CircularBuffer(0), std::exception);
13     BOOST_REQUIRE_THROW(CircularBuffer(0), std::invalid_argument); 14
       BOOST_REQUIRE_THROW(CircularBuffer(-1), std::invalid_argument);
15     BOOST_REQUIRE_NO_THROW(CircularBuffer(20));
16     }
17
18 BOOST_AUTO_TEST_CASE(size) {
19
20     CircularBuffer testBuffer(10);
21
22     BOOST_REQUIRE(testBuffer.size() == 0);
23
24     testBuffer.enqueue(5);
25     testBuffer.enqueue(5);
26
27     BOOST_REQUIRE(testBuffer.size() == 2);
28
29     testBuffer.dequeue();
30     BOOST_REQUIRE(testBuffer.size() == 1);
31
32     testBuffer.dequeue();
33     BOOST_REQUIRE(testBuffer.size() == 0);
34
35     testBuffer.enqueue(5);
36     testBuffer.dequeue();
37     BOOST_REQUIRE(testBuffer.size() == 0);
38     }
39
40     BOOST_AUTO_TEST_CASE(isEmpty) {
41     CircularBuffer testBuffer(5);
42     BOOST_REQUIRE(testBuffer.isEmpty() == true);
43
44     testBuffer.enqueue(5);
```

```
45      BOOST_REQUIRE(testBuffer.isEmpty() == false);
46      }
47
48      BOOST_AUTO_TEST_CASE(isFull) {
49      CircularBuffer testBuffer(1);
50      BOOST_REQUIRE(testBuffer.isFull() == false);
51
52      testBuffer.enqueue(5);
53      BOOST_REQUIRE(testBuffer.isFull() == true);
54      }
55
56
57      BOOST_AUTO_TEST_CASE(Enqueue) {
58      CircularBuffer testBuffer(1);
59      BOOST_REQUIRE_NO_THROW(testBuffer.enqueue(1));
60      BOOST_REQUIRE(testBuffer.dequeue() == 1);
61
62      testBuffer.enqueue(1);
63      BOOST_REQUIRE_THROW(testBuffer.enqueue(1), std::runtime_error);
64      }
65
66      BOOST_AUTO_TEST_CASE(Dequeue) {
67      CircularBuffer testBuffer(5);
68
69      testBuffer.enqueue(0);
70      testBuffer.enqueue(1);
71      testBuffer.enqueue(2);
72
73      BOOST_REQUIRE(testBuffer.dequeue() == 0);
74      BOOST_REQUIRE(testBuffer.dequeue() == 1);
75      BOOST_REQUIRE(testBuffer.dequeue() == 2);
76      BOOST_REQUIRE_THROW(testBuffer.dequeue(), std::runtime_error);
77      }
78
79      BOOST_AUTO_TEST_CASE(peek) {
80      CircularBuffer testBuffer(1);
81
82      BOOST_REQUIRE_THROW(testBuffer.peek(), std::runtime_error);
83
84      testBuffer.enqueue(1);
85      BOOST_REQUIRE(testBuffer.peek() == 1);
86      }
```

KSGuitarSim.cpp:

```
1 /*
2 * Copyright 2021 Raja Ritika Reddy Buttreddy 3  * All rights
  reserved.
4  */
5  #include <math.h>
```

```
6  #include <limits.h>
7  #include <iostream>
8  #include <string>
9  #include <exception>
10 #include <stdexcept>
11 #include <vector>
12 #include "StringSound.h" 13 #define ConA 160.0
14 #define Samppersec 44100
15
16     std::vector<sf::Int16> makeSamples(StringSound *gs) {
17     std::vector<sf::Int16> samples;
18     gs -> pluck();
19     int duration = 8;
20     int i;
21     for (i= 0; i < Samppersec * duration; i++) {
22     gs -> tic();
23     samples.push_back(gs -> sample());
24     }
25
26   return samples;
27   }
28
29   int main() {
30   sf::RenderWindow window(sf::VideoMode(1000, 1000), "PS4b pluckstring");
31   window.setFramerateLimit(60);
32
33   sf::Event event;
34   std::vector<sf::Int16> sample;
35   double freq = ConA;
36
37   std::string keyboardString = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
38   std::vector<sf::Sound> sounds(123);
39   std::vector<sf::SoundBuffer> buffers(keyboardString.size());
40
41         for (int i = 0; i < keyboardString.size(); i++) {
42         sounds[static_cast<int>(keyboardString[i])] = sf::Sound();
43         freq = ConA * pow(2, (i-24)/12.0);
44         StringSound gs = StringSound(freq);
45         sample = makeSamples(&gs);
46         if (!buffers[i].loadFromSamples(&sample[0],
47         sample.size(), 2, Samppersec))
48         throw std::runtime_error("sf::SoundBuffer: failed");
49         sounds[static_cast<int>(keyboardString[i])].setBuffer(buffers[i]);
50         }
51
52     while (window.isOpen()) {
53     while (window.pollEvent(event)) {
54     switch (event.type) { 55       case sf::Event::Closed:
56         window.close();
57         break;
```

```
58
59          case sf::Event::TextEntered:
60          sounds[event.text.unicode].play();
61
62          window.clear();
63          window.display();
64          }
65          }
66          }
67          return 0;
68          }
```

CircleBuffer.h:

```
1 /*
2 * Copyright 2021 Raja Ritika Reddy Buttreddy 3  * All rights
  reserved. 4  */
5  #ifndef CIRCULARBUFFER_H_
6  #define CIRCULARBUFFER_H_
7  #include <stdint.h>
8  #include <vector>
9  #include <exception>
10 #include <stdexcept>
11 #include <iostream>
12
13     class CircularBuffer {
14     public:
15     explicit CircularBuffer(int capacity);
16
17     void prettyPrint();
18
19     // assignment
20     bool isEmpty();
21     bool isFull();
22     void empty();
23     void enqueue(int16_t x);
24     int16_t dequeue();
25     int16_t peek();
26     int size();
27
28
29          private:
30          int len;
31          int capacity;
32          int head;
33          int tail;
34          int16_t* buffer;
35          };
36          #endif  // CIRCULARBUFFER_H_
```

CircleBuffer.cpp:

```
1               /*
2               * Copyright 2021 Raja Ritika Reddy Buttreddy
3               * All rights reserved.
4               * MIT Licensed - see http://opensource.org/licenses/MIT for details.
5               */
6               #include "CircularBuffer.h"
7               CircularBuffer::CircularBuffer(int cap) {
8               if (cap < 1) {
9               throw
10              std::invalid_argument
11              ("CircularBuffer constructor: capacity should be >0");
12              }
13
14    capacity = cap;
15    buffer = new int16_t[capacity];
16
17    head = 0;
18    tail = 0;
19    len = 0;
20    }
21
22    int CircularBuffer::size() {
23    return len;
24    }
25
26        bool CircularBuffer::isEmpty() {
27        auto x = [](int len) {
28        return len == 0;
29        };
30        return x(len);
31        }
32
33  void CircularBuffer::empty() {
34  head = 0;
35  tail = 0;
36  len = 0;
37  }
38
39    bool CircularBuffer::isFull() {
40    return (len == capacity) ? true : false;
41    }
42
43      void CircularBuffer::enqueue(int16_t x) {
44      if (isFull()) {
45      throw
46      std::runtime_error("enqueue: can't enqueue");
47      }
48
```

```
49     if (tail >= capacity) {
50     tail = 0;
51     }
52
53   buffer[tail] = x;
54
55   tail++;
56   len++;
57   }
58
59       int16_t CircularBuffer::dequeue() {
60       if (isEmpty()) {
61       throw
62       std::runtime_error("dequeue: can't dequeue");
63       }
64
65   int16_t first = buffer[head];
66   buffer[head] = 0;
67
68   head++;
69   len--;
70
71     if (head >= capacity) {
72     head = 0;
73     }
74
75   return first;
76   }
77
78       int16_t CircularBuffer::peek() {
79       if (isEmpty()) {
80       throw
81       std::runtime_error("peek: can't peek");
82       }
83       return buffer[head];
84       }
85
86   void CircularBuffer::prettyPrint() {
87   std::cout << "Buffer: capacity " << capacity << " ";
88   std::cout << ", tail " << tail;
89   std::cout << ", head " << head;
90   std::cout << ", current length " << len << "\n";
91   std::cout << "Buffer: ";
92
93   int front = 0;
94   int back = head;
95
96       while (front < len) {
97       if (back >= capacity) {
98       back = 0;
```

```
99       }
100
101     std::cout << buffer[back] << " ";
102     back++;
103     front++;
104     }
105     std::cout << std::endl;
106     }
```

StringSound.h:

```
1 /*
2 * Copyright 2021 Raja Ritika Reddy Buttreddy 3  * All rights
  reserved.
4  */
5  #include <iostream>
6  #include "CircularBuffer.h"
7  #include <vector>
8  #include <SFML/System.hpp>
9  #include <SFML/Window.hpp>
10 #include <SFML/Graphics.hpp>
11 #include <SFML/Audio.hpp>
12
13     class StringSound {
14     public:
15     explicit StringSound(double frequency);
16     explicit StringSound(std::vector<sf::Int16> init);
17     StringSound(const StringSound &obj) {}
18     ~StringSound();
19     void pluck();
20     void tic();
21     sf::Int16 sample();
22     int time(); 23  24  private:
25     CircularBuffer *buffer{};
26     int _time;
27     int blen;
28     };
```

StringSound.cpp:

```
1 /*
2 * Copyright 2021 Raja Ritika Reddy Buttreddy 3  * All rights
  reserved. 4  */
5  #include "StringSound.h"
6  #include <math.h>
7  #include <random>
8  #define Samprate 44100
9  #define DecayFac 0.996
10 #define PI 3.14159265358979323846
```

```
11
12          StringSound::StringSound(double frequency) {
13          blen = ceil(Samprate/frequency);
14          buffer = new CircularBuffer(blen);
15          for (int i = 0; i < blen; i++)
16          buffer -> enqueue(0);
17          _time = 0;
18          }
19          void StringSound::pluck() {
20          buffer -> empty();
21          int16_t val = 0;
22          for (int i = 1; i <= blen; i++) {
23          val = rand();
24          buffer->enqueue(val);
25          }
26          }
27          void StringSound::tic() {
28          int16_t val = buffer -> dequeue();
29          int16_t ns = (0.1 * val + 0.9 * buffer -> peek()) / 2 * DecayFac;
30          // drums sound
31          buffer -> enqueue(int16_t(ns));
32          _time++;
33          }
34          int StringSound::time() {
35          return _time;
36          }
37
38      sf::Int16 StringSound::sample() {
39      return buffer -> peek();
40      }
41      StringSound::~StringSound() {}
```

PS5 DNA Sequence Alignment

Description:

This assignment required them to solve a fundamental problem in computational biology and to learn about a powerful programming paradigm known as dynamic programming. To do such a task, a class must be constructed that can receive two strings, determine the edit distance between them, and compute the ideal set of actions that might be executed to make both strings equal. Insertion, deletion, and replacement are examples of such operations.

Key algorithms, Data structures and OO Designs used in this assignment:

The assignment required the usage of a two-dimensional array to hold and contain each possible edit distance between two strings for each set of operations. This dynamic programming approach was implemented with a vector, containing another vector of fast unsigned 32-bit integers.
I used this logic to find the alignment:
Starting from opt [0][0], there are four situations:

1. opt[i][j] == opt[i+1] [j+1] and x[i] == y[j], skip
2. opt[i][j] == opt[i+1] [j+1] and x[i] != y[j], modified
3. opt[i][j] == opt[i+1] [j]+2, x insert a gap
4. opt[i][j] == opt[i][j+1] +2, y insert a gap Until opt[M][N] ceases.

I chose this approach as it was the most effective way, I could think off to solve this problem from all the above approaches. I also used lambda expression in this assignment.

What I've Learned in this assignment:

PS5 helped me better grasp alternative techniques to optimizing code when performance is one of the most important considerations. Furthermore, working with two-dimensional arrays helped me develop knowledge and experience with them in general. We also used lambda expression in this program, and it helped me understand better on lambda expression implementation.

Screenshots of Output:

Image 1 – Running file example10.txt screenshot

Image 2 – Running file filelist.txt screenshot

```
sujitreddy@Sujits-MacBook-Pro ps5_SujitReddy_Anireddy % ./EDistance < example10.txt
Edit distance = 7
A T 1
A A 0
C - 2
A A 0
G G 0
T G 1
T T 0
A - 2
C C 0
C A 1
Execution time = 0.000101 seconds
```

```
sujitreddy@Sujits-MacBook-Pro ps5_SujitReddy_Anireddy % ./EDistance < filelist.txt
Edit distance = 4
A o 1
l f 1
l - 2
Execution time = 0.000147 seconds
```

Source Code:

Makefile:

```
1 #Makefile for Ps5
2 CC = g++
3 CFLAGS = -std=c++11 -c -g -O2 -Wall -Werror -pedantic
4 OBJ = main.o EDistance.o
5 DEPS = main.cpp EDistance.cpp
6 LIBS = -lsfml-system
7 EXE = EDistance
8
9      all: $(OBJ)
10     $(CC) $(OBJ) -o $(EXE) $(LIBS)
11
12 %.o: %.cpp $(DEPS)
13 $(CC) $(CFLAGS) -o $@ $< 14
15     clean:
16     rm $(OBJ) $(EXE)
```

main.cpp:

```
1 #include <iostream>
2 #include "EDistance.h"
3 #include <SFML/System.hpp>
4
5 using namespace std;
6
7      int main() {
8      sf::Clock clock;
9      sf::Time t;
10     string sequence1, sequence2;
11     cin >> sequence1 >> sequence2;
12     EDistance ed(sequence1, sequence2);
13     auto distance = ed.OptDistance();
14     auto alignment = ed.Alignment();
15     t = clock.getElapsedTime();
16     cout << "Edit distance = " << distance << endl;
17     cout << alignment << endl;
18     cout << "Execution time = " << t.asSeconds() << " seconds \n" << std::endl;
19     return 0;
20     }
```

EditDistance.h:

```
1 #ifndef EDistance_H
2 #define EDistance_H
3 #include <string>
4
5      class EDistance
6      {
7      private:
8      int **opt;
9      std::string x;
10     std::string y;
11     int M, N; 12  13 public:
```

```
14      EDistance(const std::string &a, const std::string &b);
15      ~EDistance();
16      static int penalty(char a, char b);
17      static int min(int a, int b, int c);
18      int OptDistance();
19      std::string Alignment();
20      };
21      #endif
```

EditDistance.cpp:

```
1 #include <iostream>
2 #include "EDistance.h"
3
4           EDistance::EDistance(const std::string &a, const std::string &b)
5           {
6           this->x = a;
7           this->y = b;
8           M = x.size();
9           N = y.size();
10          opt = new int *[M + 1]();
11          if(!opt)
12          {
13          std::cout << "NO memory to be allottEDistance" << std::endl;
14          exit(1);
15          }
16          for (int i = 0; i < M + 1; i++)
17          opt[i] = new int[N + 1]();
18          for (int i = 0; i <= M; i++) 19      {
20          opt[i][N] = 2 * (M - i); 21
}
22      for (int j = 0; j <= N; j++) 23
{
24          opt[M][j] = 2 * (N - j);
25          }
26          }
27
28      int EDistance::penalty(char a, char b)
29      {
30      return [](char a,char b){
31      return a == b ? 0 : 1;
32      }(a,b);
33      }
34
35      int EDistance::min(int a, int b, int c)
36      {
37      int minAB = a > b ? b : a;
38      return minAB > c ? c : minAB;
39      }
40
```

```
41          int EDistance::OptDistance()
42          {
43          for (int i = M - 1; i >= 0; i--)
44          for (int j = N - 1; j >= 0; j--)
45          opt[i][j] = min(opt[i + 1][j + 1] + penalty(x[i], y[j]), opt[i + 1][j] + 2,
            opt[i][j + 1] + 2);
46          return opt[0][0];
47          }
48
49      std::string EDistance::Alignment()
50      {
51      std::string result;
52      int i = 0, j = 0;
53      while (i < M || j < N) {
54      if (i < M && j < N && x[i] == y[j] && opt[i][j] == opt[i + 1][j + 1]) 55          {
56              result += x[i]; 57
result += " ";
58              result += y[j];
59              result += " 0";
60              i++;
61              j++;
62              } else if (i < M && j < N && x[i] != y[j] && opt[i][j] == opt[i + 1][j + 1] +
                1) 63          {
64              result += x[i]; 65
result += " ";
66              result += y[j];
67              result += " 1";
68              i++;
69              j++;
70              } else if (i < M && opt[i][j] == opt[i + 1][j] + 2)
71              {
72              result += x[i];
73              result += " - 2";
74              i++;
75              } else if (j < N && opt[i][j] == opt[i][j + 1] + 2)
76              {
77              result += "- ";
78              result += y[j];
79              result += " 2";
80              j++;
81              }
82              if(i < M || j < N) result += "\n";
83              }
84              return result;
85              }
86              EDistance::~EDistance()
87              {
88              for(int i = 0; i <= M; i++)
89              delete[] opt[i];
90              delete[] opt;
```

```
91              }
```

PS6 Random Writer

Description:

In this assignment we examine an input text for transitions between k-grams and the letter that follows it and create a textual probabilistic model with any possible text. Implementing a Markov model to assess transitions between units of k characters, or k-grams, inside a given text was the task for this project. PS5 instructed us to produce text probabilistically depending on what was provided to the model

itself using this model. I finished all the tasks given in this assignment and the code generates random text for a given input (K-series) which satisfies the logic mentioned in the PDF

Key algorithms, Data structures and OO Designs used in this assignment:

The use of constructors, datatypes, strings, and other C++ concepts are used in this assignment. The C++ map object was critical to the assignments and the Markov Model class's operation. A map with a key-value pair of a string and another nested map with a key-value pair of a character and an unsigned integer were utilized specifically. The resulting map would couple the k-gram string with a second map that has an entry for each character that follows the k-gram, as well as an unsigned integer representing the number of times that character would do so.

What I've Learned in this assignment:

This assignment has increased my grasp of key-value data structures in general, as well as complicated data structures in general. This improved comprehension includes how they can be traversed, and more experience with accessing the data within them. I tried using Lambda expression in this but could not. I finished all the tasks given in this assignment and the code generates random text for a given input (K-series) which satisfies the logic mentioned.

Screenshots of Output:

Image 1 – Running of test file screenshot

Image 2 – When input17.txt file is executed screenshot (Random Writer)

Image 3 – When monalisa.txt file is executed screenshot (Random Writer)

```
sujitreddy@Sujits-MacBook-Pro ps6_SujitReddy_Anireddy % ./test
Running 3 test cases...

*** No errors detected
```

```
sujitreddy@Sujits-MacBook-Pro ps6_SujitReddy_Anireddy % ./TextWriter 2 11 < input17.txt
ORIGINAL INPUT TEXT BELOW THIS LINE.

 gagggagagg

FINAL OUTPUT TEXT BELOW THIS LINE.

 gagagaa ga
```

```
sujitreddy@Sujits-MacBook-Pro ps6_SujitReddy_Anireddy % ./TextWriter 1 56 < monalisa.txt
ORIGINAL INPUT TEXT BELOW THIS LINE.

 _______ _,,ad8888888888bba,_ ,ad88888I888888888888888ba

FINAL OUTPUT TEXT BELOW THIS LINE.

 88888888888 ;;'(@>ZZZ88PZZZb,,I88;;l";;' Illl; _,;lZZZZ
```

Source Code:

Makefile:

```
1 #Makefile
2 CC = g++
3 CFLAGS = -g -Wall -Werror -std=c++0x -pedantic
4 SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
5 Boost = -lboost_unit_test_framework
6
7      all: TextWriter test
8      TextWriter: TextWriter.o RandWriter.o
```

```
9      $(CC) TextWriter.o RandWriter.o -o TextWriter
10     test:  test.o RandWriter.o
11     $(CC) test.o RandWriter.o -o test $(Boost)
12     TextWriter.o:TextWriter.cpp RandWriter.hpp
13     $(CC) -c TextWriter.cpp RandWriter.hpp $(CFLAGS)
14     RandWriter.o:RandWriter.cpp RandWriter.hpp
15     $(CC) -c RandWriter.cpp  RandWriter.hpp $(CFLAGS)
16     test.o:test.cpp
17     $(CC) -c test.cpp $(Boost) 18 clean:
19     rm *.o
20     rm *.gch
21     rm TextWriter
22     rm test
```

test.cpp:

```
1 #include <iostream>
2 #include <string>
3 #include <exception>
4 #include <stdexcept>
5 #include "RandWriter.hpp"
6 #define BOOST_TEST_DYN_LINK
7 #define BOOST_TEST_MODULE Main
8 #include <boost/test/unit_test.hpp>
9
10 using namespace std;
11
12   BOOST_AUTO_TEST_CASE(order0)
13   {
14   BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 0));
15
16   RandWriter mm("gagggagaggcgagaaa", 0);
17
18   BOOST_REQUIRE(mm.order() == 0);
19   BOOST_REQUIRE(mm.freq("") == 17);
20   BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
21
22   BOOST_REQUIRE(mm.freq("", 'g') == 9);
23   BOOST_REQUIRE(mm.freq("", 'a') == 7);
24   BOOST_REQUIRE(mm.freq("", 'c') == 1);
25   BOOST_REQUIRE(mm.freq("", 'x') == 0);
26
27 }
28
29   BOOST_AUTO_TEST_CASE(order1)
30   {
31   BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 1));
32
33   RandWriter mm("gagggagaggcgagaaa", 1);
34
35   BOOST_REQUIRE(mm.order() == 1);
36   BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
37   BOOST_REQUIRE_THROW(mm.freq("xx"), std::runtime_error);
38
39   BOOST_REQUIRE(mm.freq("a") == 7);
40   BOOST_REQUIRE(mm.freq("g") == 9);
41   BOOST_REQUIRE(mm.freq("c") == 1);
42
43   BOOST_REQUIRE(mm.freq("a", 'a') == 2);
```

```
44   BOOST_REQUIRE(mm.freq("a", 'c') == 0);
45   BOOST_REQUIRE(mm.freq("a", 'g') == 5);
46
47   BOOST_REQUIRE(mm.freq("c", 'a') == 0);
48   BOOST_REQUIRE(mm.freq("c", 'c') == 0);
49   BOOST_REQUIRE(mm.freq("c", 'g') == 1);
50
51   BOOST_REQUIRE(mm.freq("g", 'a') == 5);
52   BOOST_REQUIRE(mm.freq("g", 'c') == 1);
53   BOOST_REQUIRE(mm.freq("g", 'g') == 3);
54
55   BOOST_REQUIRE_NO_THROW(mm.randk("a"));
56   BOOST_REQUIRE_NO_THROW(mm.randk("c"));
57   BOOST_REQUIRE_NO_THROW(mm.randk("g"));
58
59   BOOST_REQUIRE_THROW(mm.randk("x"), std::runtime_error);
60
61   BOOST_REQUIRE_THROW(mm.randk("xx"), std::runtime_error);
62
63   }
64   BOOST_AUTO_TEST_CASE(order2)
65   {
66   BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 2));
67
68   RandWriter mm("gagggagaggcgagaaa", 2);
69
70   BOOST_REQUIRE(mm.order() == 2);
71
72   BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
73   BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
74   BOOST_REQUIRE_NO_THROW(mm.freq("xx"));
75   BOOST_REQUIRE_THROW(mm.freq("", 'g'), std::runtime_error);
76   BOOST_REQUIRE_THROW(mm.freq("x", 'g'), std::runtime_error);
77   BOOST_REQUIRE_THROW(mm.freq("xxx", 'g'), std::runtime_error);
78
79   BOOST_REQUIRE(mm.freq("aa") == 2);
80   BOOST_REQUIRE(mm.freq("aa", 'a') == 1);
81   BOOST_REQUIRE(mm.freq("aa", 'c') == 0);
82   BOOST_REQUIRE(mm.freq("aa", 'g') == 1);
83
84   BOOST_REQUIRE(mm.freq("ag") == 5);
85   BOOST_REQUIRE(mm.freq("ag", 'a') == 3);
86   BOOST_REQUIRE(mm.freq("ag", 'c') == 0);
87   BOOST_REQUIRE(mm.freq("ag", 'g') == 2);
88
89   BOOST_REQUIRE(mm.freq("cg") == 1);
90   BOOST_REQUIRE(mm.freq("cg", 'a') == 1);
91   BOOST_REQUIRE(mm.freq("cg", 'c') == 0);
92   BOOST_REQUIRE(mm.freq("cg", 'g') == 0);
93
94   BOOST_REQUIRE(mm.freq("ga") == 5);
95   BOOST_REQUIRE(mm.freq("ga", 'a') == 1);
96   BOOST_REQUIRE(mm.freq("ga", 'c') == 0);
97   BOOST_REQUIRE(mm.freq("ga", 'g') == 4);
98
99    BOOST_REQUIRE(mm.freq("gc") == 1);
100   BOOST_REQUIRE(mm.freq("gc", 'a') == 0);
101   BOOST_REQUIRE(mm.freq("gc", 'c') == 0);
102   BOOST_REQUIRE(mm.freq("gc", 'g') == 1);
103
104   BOOST_REQUIRE(mm.freq("gg") == 3);
105   BOOST_REQUIRE(mm.freq("gg", 'a') == 1);
```

```
106   BOOST_REQUIRE(mm.freq("gg", 'c') == 1);
107   BOOST_REQUIRE(mm.freq("gg", 'g') == 1);
108
109 }
```

RandWriter.hpp

```
1  /*
2  * Copyright 2021 Raja Ritika Reddy Buttreddy
3  * All rights reserved.
4  * MIT Licensed - see http://opensource.org/licenses/MIT for details.
5  *
6  */
7  #ifndef RandWriter_HPP
8  #define RandWriter_HPP
9  #include <algorithm>
10 #include <iostream>
11 #include <map>
12 #include <string>
13 #include <stdexcept> 14 class RandWriter {
15   public:
16   RandWriter(std::string text, int k);
17   int order();
18   int freq(std::string kgram);
19   int freq(std::string kgram, char c);
20   char randk(std::string kgram);
21   std::string gen(std::string kgram, int T);
22   friend std::ostream& operator<< (std::ostream &out, RandWriter &mm);
23   private:
24   int _order;
25   std::map <std::string, int> _kgrams;
26   std::string _alphabet;
27   };
28   #endif
```

Randwriter.cpp

```
1  /*
2  * Copyright 2021 Raja Ritika Reddy Buttreddy
3  * All rights reserved.
4  * MIT Licensed - see http://opensource.org/licenses/MIT for details.
5  *
6  */
7  #include "RandWriter.hpp"
8  #include <algorithm>
9  #include <map>
10 #include <string>
11 #include <stdexcept>
12 #include <vector>
13 #include <utility>
14
```

```
15        RandWriter::RandWriter(std::string text, int k)
16        {
17        _order = k;
18        srand((int)time(NULL));
19        std::string text_circular = text;
20        for (int a = 0; a < _order; a++)
21        {
22        text_circular.push_back(text[a]);
23        }
24        int text_len = text.length();
25        char tmp;
26        bool inAlpha = false;
27        for (int i = 0; i < text_len; i++)
28        {
29        tmp = text.at(i);
30        inAlpha = false;
31        for (unsigned int y = 0; y < _alphabet.length(); y++)
32        {
33        if (_alphabet.at(y) == tmp)
34        {
35        inAlpha = true;
36        }
37        }
38        if (!inAlpha) {
39        _alphabet.push_back(tmp);
40        }
41        }
42        std::sort(_alphabet.begin(), _alphabet.end());
43        std::string tmp_str;
44        int x, y;
45        for (x = _order; x <= _order + 1; x++)
46        {
47        for (y = 0; y < text_len; y++)
48        {
49        tmp_str.clear();
50        tmp_str = text_circular.substr(y, x);
51        _kgrams.insert(std::pair<std::string, int>(tmp_str, 0)); 52     }
53      }
54      std::map<std::string, int>::iterator it;
55      int count_tmp = 0;
56      for (x = _order; x <= _order + 1; x++)
57      {
58      for (y = 0; y < text_len; y++)
59      {
60      tmp_str.clear();
61      tmp_str = text_circular.substr(y, x);
62      it = _kgrams.find(tmp_str);
63      count_tmp = it->second;
64      count_tmp++;
65      _kgrams[tmp_str] = count_tmp;
```

```
66      }
67      }
68      }
69      int RandWriter::order()
70      {
71      return _order;
72      }
73      int RandWriter::freq(std::string kgram)
74      {
75      if (kgram.length() != (unsigned)_order)
76      {
77      throw
78      std::runtime_error("Error - kgram not of length k.");
79      }
80      std::map<std::string, int>::iterator it;
81      it = _kgrams.find(kgram);
82      if (it == _kgrams.end())
83      {
84      return 0;
85      }
86      return it->second;
87      }
88      int RandWriter::freq(std::string kgram, char c)
89      {
90      if (kgram.length() != (unsigned)_order)
91      {
92      throw
93      std::runtime_error("Error - kgram not of length k.");
94      }
95      std::map<std::string, int>::iterator it;
96      kgram.push_back(c);
97      it = _kgrams.find(kgram);
98      if (it == _kgrams.end())
99      {
100     return 0;
101     }
102     return it->second;
103     }
104     char RandWriter::randk(std::string kgram)
105     {
106     if (kgram.length() != (unsigned)_order)
107     {
108     throw std::runtime_error("Error - kgram not of length k (randk)");
109     }
110     std::map<std::string, int>::iterator it;
111     it = _kgrams.find(kgram);
112     if (it == _kgrams.end())
113     {
114     throw std::runtime_error("Error - Could not find the given kgram! (randk)"); 115   }
116      int kgram_freq = freq(kgram);
```

```
117       int random_value = rand() % kgram_freq;
118       double test_freq = 0;
119       double random_num = static_cast<double>(random_value) / kgram_freq;
120       double last_values = 0;
121       for (unsigned int a = 0; a < _alphabet.length(); a++) {
122       test_freq =  static_cast<double>(freq(kgram, _alphabet[a])) / kgram_freq;
123       if (random_num < test_freq + last_values && test_freq != 0) {
124       return _alphabet[a];
125       }
126       last_values += test_freq;
127       }
128       return '-';
129       }
130       std::string RandWriter::gen(std::string kgram, int T)
131       {
132       if (kgram.length() != (unsigned)_order)
133       {
134       throw std::runtime_error("Error - kgram not of length k. (gen)");
135       }
136       std::string final_string = "";
137       char return_char;
138       final_string += "" + kgram;
139       for (unsigned int a = 0; a < (T - (unsigned)_order); a++)  140   {
141     return_char = randk(final_string.substr(a, _order));
142     final_string.push_back(return_char);
143     }
144     return final_string;
145     }
146     std::ostream& operator<< (std::ostream &out, RandWriter &mm)
147     {
148     out << "\n_Order: " << mm._order << "\n";
149     out << "Alphabet: "<< mm._alphabet << "\n";
150     out << "Kgrams map: \n\n";
151     std::map<std::string, int>::iterator it;
152     for (it = mm._kgrams.begin(); it != mm._kgrams.end(); it++)
153     {
154     out << it->first << "\t" << it->second << "\n";
155     }
156     return out;
157     }
```

Textwriter.cpp

```
1        /*
2        * Copyright 2021 Raja Ritika Reddy Buttreddy
3        * All rights reserved.
4        * MIT Licensed - see http://opensource.org/licenses/MIT for details.
5        *
6        */
7        #include <string>
```

```
8       #include "RandWriter.hpp"
9       int main(int argc, const char* argv[])
10      {
11      if (argc != 3)
12      {
13      std::cout << "Usage: ./TextGenerator (int K) (int T)\n";
14      return 0;
15      }
16      std::string str_k(argv[1]);
17      std::string str_t(argv[2]);
18      int k = std::stoi(str_k);
19      int t = std::stoi(str_t);
20      std::string input = "";
21      std::string current_txt = "";
22      while (std::cin >> current_txt)
23      {
24      input += " " + current_txt;
25      current_txt = "";
26      }
27      std::cout << "ORIGINAL INPUT TEXT BELOW THIS LINE.\n\n";
28      for (int a = 0; a < t; a++)
29      {
30      std::cout << input[a];
31      if (input[a] == '.' || input[a] == '!')
32      {
33      std::cout << "\n";
34      }
35      }
36      std::string output_string = "";
37      RandWriter amazing(input, k);
38      output_string += "" + amazing.gen(input.substr(0, k), t);
39      std::cout << "\n\nFINAL OUTPUT TEXT BELOW THIS LINE.\n\n";
40      for (int a = 0; a < t; a++) {
41      std::cout << output_string[a];
42      if (output_string[a] == '.' || output_string[a] == '!')
43      {
44      std::cout << "\n";
45      }
46      }
47      std::cout << "\n";
48      return 0;
49      }
```

PS7 Kronos Time Clock

Description:

This assignment involved parsing files of different Kronos InTouch time clock logs to examine them, checking the device's boot up timing, and recording whether these starts were totally successful. Scan the complete log file and create a text file report chronologically describing each time the device was restarted.

Key algorithms, Data structures and OO Designs used in this assignment:

The Regex I used: (. *): (\ (log.c.166\) server started. *). The starter code given with all the other information was a good start for me. Eventually I used these two expressions (regex) to figure out what the current line is, I used these parameters to track my progress through the project -

If it’s started - I recorded the time to "t1"

If it's finished, I recorded the time to "t2"

To calculate the value of (t2 - t1) - I used a flag to determine if the startup failed or not

If the current flag is true - expect the next one to be "completion row" If

the next row is still at "start line" - the last start failed.

This is the general idea of the approach I used to solve this problem,

What I've Learned in this assignment:

This project provided a thorough introduction to regular expressions inside the C++ language and increased my comfort level with the Boost regex library, as well as other regex libraries in general. The assignment's goal was to assist me improve my ability to output complete files while also providing an easy approach to effectively parse a file. Finally, the Boost date and time methods that were built were able to make my code more efficient in computing the elapsed time based on the text inputs from the log file. I did not implement lambda expression in this assignment.

Screenshots of Output:

Image 1 - device1_intouch.log.rpt (output screenshot)

Image 2 – device2_intouch.log.rpt
Image 3 – device3_intouch.log.rpt
Image 4 – device4_intouch.log.rpt
Image 5 – device5_intouch.log.rpt
Image 6 – device6_intouch.log.rpt

ps7_SujitReddy_Anireddy > device1_intouch.log.rpt

```
1   435369 (log.c.166) server started 2014-03-25 19:11:59 success 183000ms
2   436500 (log.c.166) server started 2014-03-25 19:29:59 success 165000ms
3   440719 (log.c.166) server started 2014-03-25 22:01:46 success 161000ms
4   440866 (log.c.166) server started 2014-03-26 12:47:42 success 167000ms
5   442094 (log.c.166) server started 2014-03-26 20:41:34 success 159000ms
6   443073 (log.c.166) server started 2014-03-27 14:09:01 success 161000ms
```

ps7_SujitReddy_Anireddy > device2_intouch.log.rpt

```
1   498921 (log.c.166) server started 2014-03-11 15:42:26 success 162000ms
```

ps7_SujitReddy_Anireddy > 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
```

ps7_SujitReddy_Anireddy > 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
```

ps7_SujitReddy_Anireddy > 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
```

ps7_SujitReddy_Anireddy > device6_intouch.log.rpt

```
2 (log.c.166) server started 2014-04-03 20:27:48 success 193000ms
82079 (log.c.166) server started 2014-04-09 14:51:15 success 204000ms
85398 (log.c.166) server started 2014-04-10 18:13:13 success 204000ms
85957 (log.c.166) server started 2014-04-10 19:11:05 success 199000ms
86127 (log.c.166) server started 2014-04-10 19:18:36 success 200000ms
86568 (log.c.166) server started 2014-04-10 19:32:16 success 200000ms
86750 (log.c.166) server started 2014-04-10 20:06:27 success 160000ms
86939 (log.c.166) server started 2014-04-11 00:15:56 success 173000ms
87116 (log.c.166) server started 2014-04-11 13:28:25 success 167000ms
87836 (log.c.166) server started 2014-04-11 13:58:02 success 167000ms
88983 (log.c.166) server started 2014-04-11 14:23:42 success 169000ms
90112 (log.c.166) server started 2014-04-14 12:13:59 failure
90135 (log.c.166) server started 2014-04-14 12:16:13 failure
90176 (log.c.166) server started 2014-04-14 12:18:44 success 161000ms
```

Source Code:

Makefile:

```
#Makefile for ps7
CC = g++
CFLAGS = -std=c++11 -c -g -Og -Wall -Werror -pedantic
OBJ = main.o
DEPS = main.cpp
LIBS = -lboost_regex -lboost_date_time
EXE = ps7

     all: $(OBJ)
```

```
10     $(CC) $(OBJ) -o $(EXE) $(LIBS)
11
12 %.o: %.cpp $(DEPS)
13 $(CC) $(CFLAGS) -o $@ $< 14  15 clean:
16  rm $(OBJ) $(EXE)
```

main.cpp:

```
1 #include <iostream>
2 #include <string>
3 #include <fstream>
4 #include <boost/regex.hpp>
5 #include "boost/date_time/posix_time/posix_time.hpp"
6
7
8  using std::cout;
9  using std::cin;
10 using std::endl;
11 using std::string;
12 using boost::regex;
13 using boost::smatch;
14 using boost::regex_error;
15 using boost::gregorian::date;
16 using boost::gregorian::from_simple_string;
17 using boost::gregorian::date_period;
18 using boost::gregorian::date_duration;
19 using boost::posix_time::ptime;
20 using boost::posix_time::time_duration;
21
22
23                    int main(int argc, char **args)
24                    {
25                    if (argc != 2)
26                    {
27                    cout << "usage: ./ps7 [logfile]" << endl;
28                    exit(1);
29                    }
30                    string s, rs;
31                    regex e1;
32                    regex e2;
33                    bool flag = false;
34                    ptime t1, t2;
35                    string filename(args[1]);
36                    std::ifstream infile(filename);
37                    std::ofstream outfile(filename + ".rpt");
38                    if (!infile || !outfile)
39                    {
40                    cout << "open file error" << endl;
41                    exit(1);
42                    }
```

```
43                      try
44                      {
45                      e1 = regex(R"((.*): (\(log.c.166\) server started.*))");
46                      e2 = regex("(.*)\\.\\d*:INFO:oejs.AbstractConnector:Started "
47                      "SelectChannelConnector@0.0.0.0:9080.*");
48                      }
49                      catch (regex_error &exc)
50                      {
51                      cout << "Regex constructor failed with code " << exc.code() << endl;
52                      exit(1);
53                      }
54                      int line_number = 1;
55                      string str;
56                      while (getline(infile, s))
57                      {
58                      if (regex_match(s, e1))
59                      {
60                      smatch sm;
61                      regex_match(s, sm, e1);
62                      if (flag)
63                      {
64                      outfile << "failure" << endl;
65                      }
66                      flag = true;
67                      t1 = ptime(boost::posix_time::time_from_string(sm[1]));
68                      str = sm[2];
69                      outfile << line_number << " (log.c.166) server started "
70                      << sm[1] << " ";
71                      }
72                      if (regex_match(s, e2))
73                      {
74                      smatch sm;
75                      regex_match(s, sm, e2);
76                      t2 = ptime(boost::posix_time::time_from_string(sm[1]));
77                      outfile << "success " << (t2 - t1).total_milliseconds()
78                      << "ms" << endl;
79                      flag = false;
80                      }
81                      line_number++;
82                      }
83                      if (flag)
84                      {
85                      outfile << "failure" << endl;
86                      }
87                      infile.close();
88                      outfile.close();
89                      return 0;
90                      }
```