

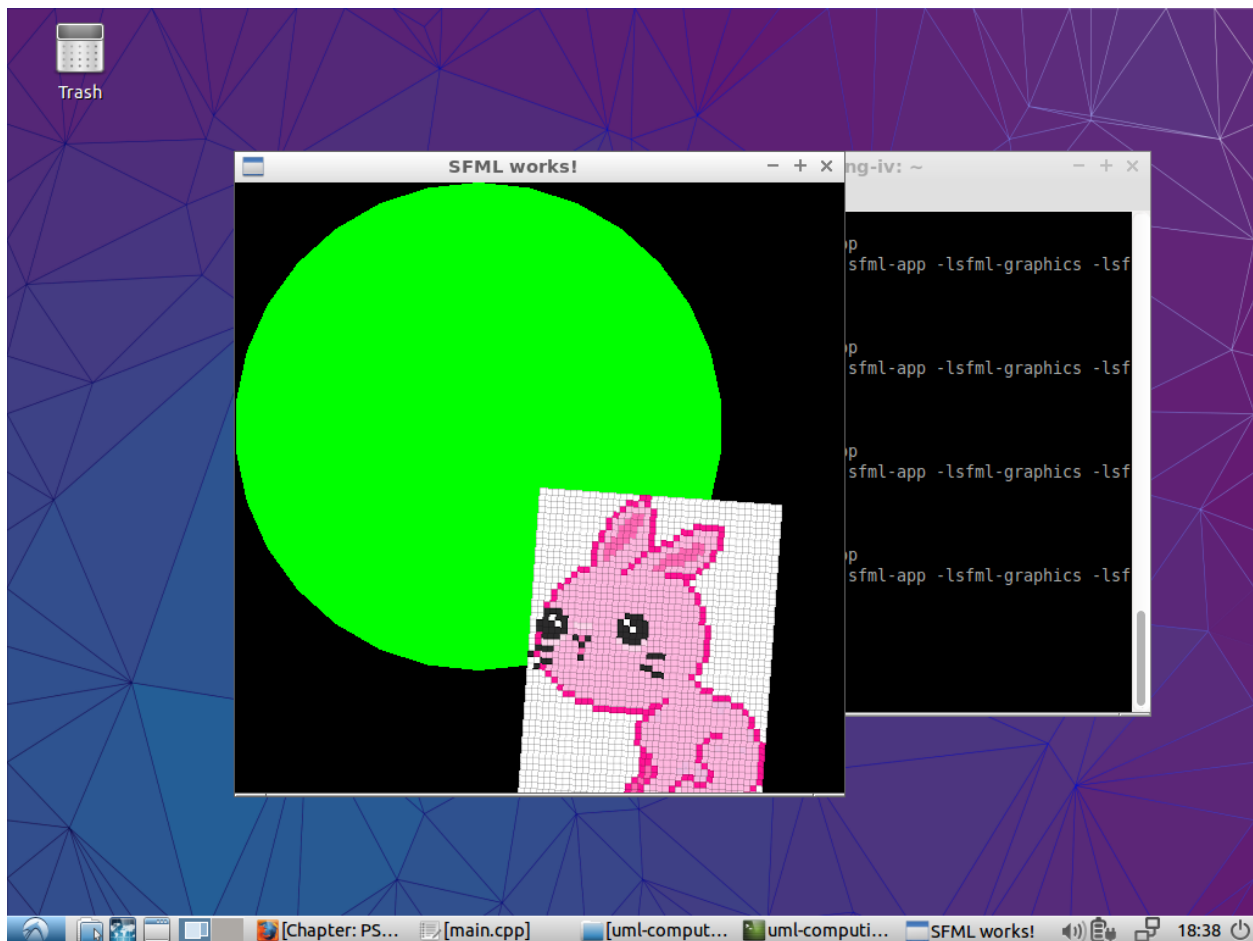
Thiago Lima
COMP IV: Project Portfolio
Fall 2018

Contents:

- PS0** Hello World with SFML
- PS1** Recursive Graphic
- PS2** Linear Feedback Shift Register and Image Encoding
 - PS2a** Linear Feedback Shift Register and Unit Testing
 - PS2b** Encoding images with LFSR
- PS3** N-Body Simulation
 - PS3a** Design a program that loads and displays a static universe
 - PS3b** Using Newton's laws of physics, animate the universe
- PS4** Edit Distance
- PS5** Ring Buffer and Guitar Hero
 - PS5a** Ring Buffer with cpplint, testing, and exceptions
 - PS5b** GuitarHero
- PS6** Markov Model of Natural Language
- PS** Kronos Intouch Parsing
 - PS7a**
 - PS7b**

PS0 Hello World with SFML:

This assignment asked me to create an SFML window with a shape and a sprite that could be moved. This was our first assignment and taught us to work with the SFML library and other important features like creating shape and working with windows and events as well as drawable tools. One of the big algorithms I implemented in this project was the event people function and how they worked with sprites As well as using the window function to draw shapes and pictures on a window.




```
62:         }
63:     }
64:
65:     if (left) x-= 1;
66:     if (right) x+= 1;
67:     if (up) y-= 1;
68:     if (down) y+= 1;
69:
70:
71:     if (x<0) x=0;
72:     if (x>(int>window.getSize().x)
73: x>window.getSize().x;
74:
75:
76:     if (y<0) y=0;
77:     if (y>(int>window.getSize().y)
78: y>window.getSize().y;
79:
80:     sprite.setPosition(x,y);
81:     sprite.setRotation( timer.getElapsedTime().asSeconds() / M_PI );
82:     window.draw(sprite);
83:
84:     sprite.setPosition(x,y);
85:     window.clear();
86:     window.draw(shape);
87:     window.draw(sprite);
88:     window.display();
89: }
90:
91: return 0;
92: }
```

PS1: Recursive Graphics (Pythagoras tree):

The purpose of this assignment was to create a Pythagoras tree which is a Greek mathematical fractal are repeating squares that are created on top of each other and smaller in a rotational pattern. In this assignment I created rectangles but was not able to perfectly create the fractal shape using the recursive implementation. my square shape was created and I was able to get a few implementations done after the base but not as far as the program required. But throughout this program I did learn a good amount from it like how would you create more complex shapes through the SFML library and how to work with convex and rectangular shapes and how to work with drawable classes and create a class for a shape.

```
1: CC= g++
2: CFLAGS= -Wall -Werror -ansi -pedantic
3: LFLAGS= -lsfml-graphics -lsfml-window -lsfml-system
4:
5: all:    pTree
6:
7: pTree:  main.o pTree.o
8:         $(CC) main.o pTree.o -o pTree $(LFLAGS)
9:
10: main.o: main.cpp pTree.hpp
11:         $(CC) -c main.cpp pTree.hpp $(CFLAGS)
12:
13: pTree.o:      pTree.cpp pTree.hpp
14:         $(CC) -c pTree.cpp pTree.hpp $(CFLAGS)
15:
16:
17:
18: clean:
19:         rm *.o
20:         rm pTree
```

```
1:
2: #include "pTree.hpp"
3:
4: int main(int argc, char const *argv[]) {
5:
6:     if (argc != 2) {
7:         cout << "Error invalid number of arg";
8:         return 0;
9:     }
10:
11:     int l = atoi(argv[1]);
12:     int n = atoi(argv[2]);
13:
14:     sf::RenderWindow window(sf::VideoMode(500, 500), "Original Recursive Image
");
15:
16:     window.setFramerateLimit(60);
17:
18:     sf::Texture texture;
19:
20:
21:     pTree square(l, n);
22:
23:
24:
25:     while (window.isOpen()) {
26:
27:         sf::Event event;
28:
29:         while (window.pollEvent(event)) {
30:
31:             if (event.type == sf::Event::Closed) {
32:                 window.close();
33:             }
34:         }
35:
36:         window.clear();
37:         window.draw(square);
38:         window.display();
39:     }
40:
41:     return 0;
42: }
```

```
1: #ifndef pTree_HPP
2: #define pTree_HPP
3:
4:
5: #include <stdlib.h>
6: #include <stdint.h>
7: #include <SFML/Graphics.hpp>
8: #include <SFML/System.hpp>
9: #include <SFML/Window.hpp>
10: #include <iostream>
11: #include <cmath>
12: #include <vector>
13:
14: using namespace std;
15:
16: class pTree : public sf::Drawable
17: {
18: public:
19:
20:     pTree(double l, int n);
21:
22:
23: private:
24:
25:     void virtual draw(sf::RenderTarget& target, sf::RenderStates states) const;
26:
27:     int _l;
28:     int _n;
29:     sf::Sprite _sprite;
30:     sf::Texture _texture;
31:     vector<RectangleShape> square;
32:
33:
34: };
35:
36: #endif
```



```
1: // Thiago
2:
3: #include "pTree.hpp"
4:
5: pTree::pTree(double l, int n){
6:
7:     _l = l;
8:     _n = n;
9:
10: }
11:
12:
13:
14: void pTree::draw(sf::RenderTarget&target, sf::RenderStates states) const {
15:     sf::RectangleShape squarebase;
16:     squarebase.setSize(sf::Vector2f(_l, _l));
17:     squarebase.setOutlineColor(sf::Color::Red);
18:     squarebase.setOutlineThickness(1);
19:     squarebase.setPosition(250, 250);
20:
21:     int newl = sqrt((pow(_l, 2) / 2));
22:     int rotation = 225;
23:
24:     for (size_t i = 1; i < ; i++) {
25:         square.push_back();
26:     }
27:
28:     for (size_t i = 1; i < (_n ); i++) {
29:         square.push_back();
30:     }
31:
32:     sf::RectangleShape square;
33:     square[0].setSize(sf::Vector2f(newl, newl));
34:     square[0].setOutlineColor(sf::Color::Blue);
35:     square[0].setRotation(rotation);
36:     square[0].setOutlineThickness(1);
37:     square[0].setPosition((250), (250));
38:
39:     sf::RectangleShape square3;
40:     square3.setSize(sf::Vector2f(newl, newl));
41:     square3.setOutlineColor(sf::Color::Green);
42:     square3.setRotation(rotation);
43:     square3.setOutlineThickness(1);
44:     square3.setPosition((250 + _l), (250));
45:
46:
47:     target.draw(square);
48:     target.draw(square2);
49:     target.draw(square3);
50: }
```

PS2: Linear Feedback Shift Register (part A):

In the first part of PS 2 I was supposed to create a linear feedback shift register which takes a number of bits and a position in those bits, and shift the bit to the left while creating a new bit by using an exclusive or with the last and position, this will create a new bit then this process is repeated a number of times you will get your original number bits back. I was able to successfully create this program using linear feedback shift registers. One of the most important algorithms that were used in this assignment was the creating the feedback register and I learn how to work with this algorithm through this program.

PS2: Linear Feedback Shift Register (part B):

In part 2 of this assignment we worked off of the linear feedback shift register that we created in the part one to manipulate an image given to us and shifting the bits of the images RGB color palette. This would create a scrambled image and we would be able to use the linear feedback shift register to unscramble the image original picture back. This assignment worked perfectly i was able to change the RGB color palette of our image and create a perfectly encoded and decoded image. In this assignment I learned how to work with images in SFML And how to manipulate images I also learned how to work with more than one window and how to find the important fundamentals of images like there RGB color palette and how to work with multiple drawable windows.



```
1: # Compiler
2: CC=g++
3:
4: # Compiler options
5: CFLAGS=-c -Wall -Werror -ansi -pedantic
6:
7: # Linker options
8: LDFLAGS=
9:
10: # Libraries
11: LIBS=-lsfml-graphics -lsfml-window -lsfml-system
12:
13: # Build executable PhotoMagic
14: all: PhotoMagic
15:
16: # Build PhotoMagic
17: PhotoMagic: LFSR.o PhotoMagic.o
18:     $(CC) $(LDFLAGS) -o PhotoMagic LFSR.o PhotoMagic.o $(LIBS)
19:
20: LFSR.o: LFSR.cpp LFSR.hpp
21:     $(CC) $(CFLAGS) LFSR.cpp
22:
23: PhotoMagic.o: PhotoMagic.cpp LFSR.hpp
24:     $(CC) $(CFLAGS) PhotoMagic.cpp
25:
26: # Clean all temporary files
27: clean:
28:     rm *.o PhotoMagic
```

```
1: // pixels.cpp:
2: // using SFML to load a file, manipulate its pixels, write it to disk
3: // Fred Martin, fredm@cs.uml.edu, Sun Mar 2 15:57:08 2014
4:
5: // g++ -o pixels pixels.cpp -lsfml-graphics -lsfml-window
6:
7: #include <SFML/System.hpp>
8: #include <SFML/Window.hpp>
9: #include <SFML/Graphics.hpp>
10:
11: #include "LFSR.hpp"
12: using namespace std;
13:
14: sf::Image code(sf::Image image, LFSR lfsr, int k){
15:     sf::Color p;
16:     for (unsigned int x = 0; x < image.getSize().x; x++) {
17:         for (unsigned int y = 0; y < image.getSize().y; y++)
18:             {
19:                 p = image.getPixel(x, y);
20:                 p.r = lfsr.generate(k) ^ p.r;
21:                 p.g = lfsr.generate(k) ^ p.g;
22:                 p.b = lfsr.generate(k) ^ p.b;
23:                 image.setPixel(x, y, p);
24:             }
25:     return image;
26: }
27:
28: int main(int argc, char* argv[]) {
29:
30:     string Input = argv[1];
31:     string Output = argv[2];
32:     string string = argv[3];
33:     int t = atoi(argv[4]);
34:
35:
36:     sf::Image image;
37:     sf::Texture texture;
38:     sf::Sprite sprite;
39:
40:
41:
42:     image.loadFromFile(Input);
43:     LFSR l(string, t);
44:     image = code(image, l, t);
45:     image.saveToFile(Output);
46:     image.loadFromFile(Input);
47:
48:     sf::Image image2;
49:     sf::Texture texture2;
50:     sf::Sprite sprite2;
51:
52:     image2.loadFromFile(Output);
53:     texture.loadFromImage(image);
54:     sprite.setTexture(texture);
55:     texture2.loadFromImage(image2);
56:     sprite2.setTexture(texture2);
57:     //sprite2.setPosition(0, image.getSize().y);
58:
59:     sf::RenderWindow window(sf::VideoMode(image.getSize().x, image.getSi
ze().y), "Work");
```

```
60:         sf::RenderWindow window2(sf::VideoMode(image.getSize().x, image.getS
ize().y), "Work");
61:
62:         while (window.isOpen() && window2.isOpen()) {
63:             sf::Event event;
64:
65:             while (window.pollEvent(event)) {
66:                 if (event.type == sf::Event::Closed)
67:                     window.close();
68:             }
69:
70:             while (window2.pollEvent(event)) {
71:                 if (event.type == sf::Event::Closed)
72:                     window2.close();
73:             }
74:             window.clear();
75:             window.draw(sprite);
76:             window.display();
77:             //window2.clear();
78:             window2.draw(sprite2);
79:             window2.display();
80:
81:         }
82:
83:
84:
85:
86:
87:
88:         return 0;
89: }
```

```
1: #ifndef LFSR_HPP
2: #define LFSR_HPP
3:
4: #include <iostream>
5: #include <string>
6: #include <sstream>
7:
8: using namespace std;
9:
10: class LFSR{
11: public:
12:
13: LFSR(string seed, int t);
14:
15: int step();
16:
17: int generate(int k);
18:
19:
20: friend ostream& operator<< (ostream &out, LFSR &cLFSR);
21:
22: private:
23: string seed;
24: int t;
25: };
26:
27: #endif
```

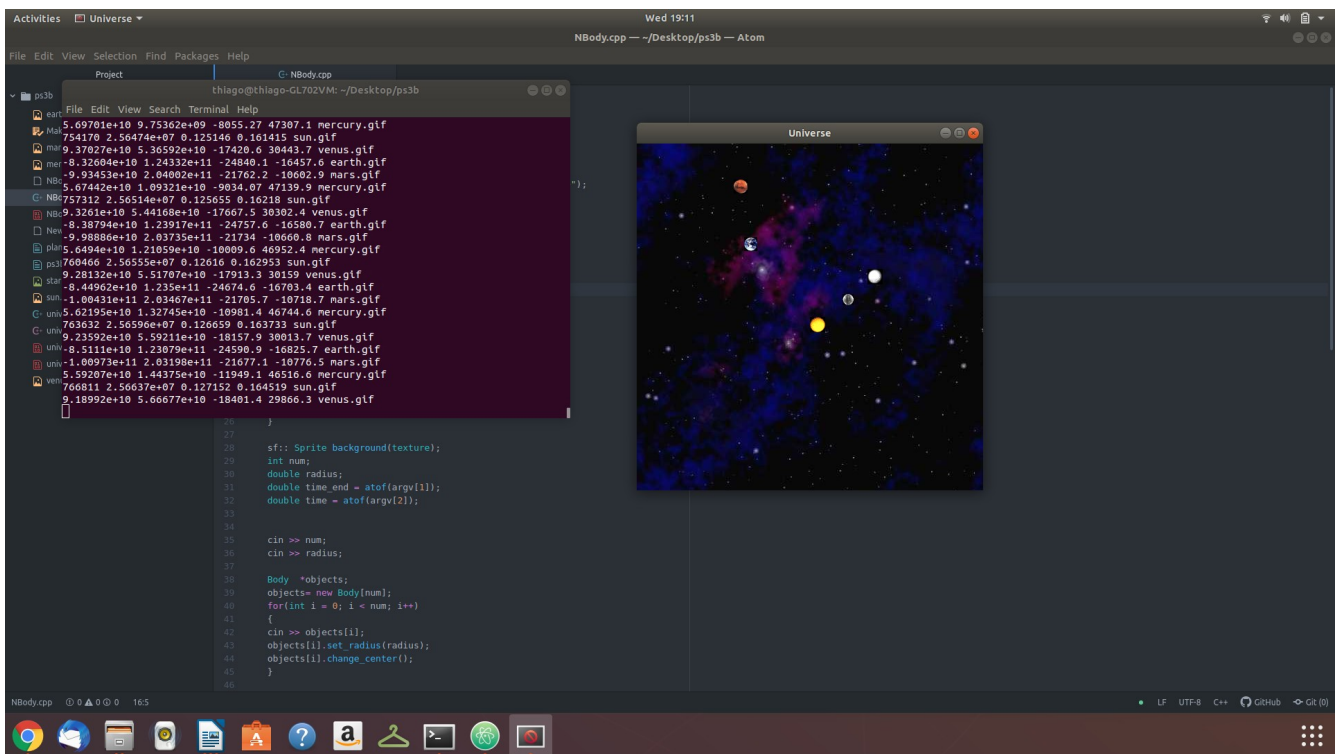
```
1: #include "LFSR.hpp"
2:
3: LFSR::LFSR(string seed, int t):seed(seed), t(t) {
4: }
5:
6:
7:
8:
9: int LFSR::step()
10: {
11:     int last = seed.length() - 1;
12:     int reverse = last - t;
13:
14:     char tap = seed.at(reverse);
15:     char left = seed.at(0);
16:
17:     for(int i = 0; i < last; i++)
18:     {
19:         seed.at(i) = seed.at(i+1);
20:     }
21:
22:     if(left == tap)
23:     {
24:         seed.at(last) = '0';
25:         return 0;
26:     }
27:
28:     else
29:     {
30:         seed.at(last) = '1';
31:         return 1;
32:     }
33: }
34: }
35:
36: int LFSR::generate(int k)
37: {
38:     int gen = 0;
39:     for(int i = 0; i < k; i++)
40:     {
41:         int temp = step();
42:         gen *= 2;
43:         gen += temp;
44:     }
45:     return gen;
46: }
47:
48: ostream& operator<< (ostream &out, LFSR &cLFSR)
49: {
50:     out << cLFSR.seed;
51:     return out;
52: }
53:
54:
55:
56:
57:
```

PS3: N-Body Simulation:

In the first part of my assignment I was tasked with creating a universe using planet and a sun shaped sprite, my program would also accept a file of specific requirements for the spacing in positioning of each sprite which I would have to calculate and display in my universe. I created a planet class which had variables that would accept everything inside of the file including the Filename, then it would calculate all of the information in setting it to scale in my Universe and displaying out each in the universe I created. This programming assignment helped me understand how to implement drawable classes that would take many sprites and create them in a window, I also learned about the origin in center the SFML class windows and worked with overloading the inputs and outputs operators to work with the classes.

PS3: N-Body Simulation(Part B):

In the second part of my assignment I was tasked with implementing a moving universe that was to scale with our solar system, I would take the information given to me in the first assignment and use force and gravity values to figure out how my sprites would move in a active solar system. In this assignment I learned a good amount of how to use arguments from command line and I'm learn how to work with multiple sprites and implement a moving system which was something that I didn't know I could do using SFML. I created multiple functions that would work with each variable I was given in mathematical figure out how each universe would move but one of the most important things I learned was how to make and updating drawable function in the SFML window. The key algorithm that was implemented in this program was the algorithm too take invariables calculate their new position based on their old positions and how it was all related in the system and to output that to the drawable window which was extremely difficult but I successfully implemented.




```
1: CC= g++ -std=c++14
2: CFLAGS= -Wall -Werror -ansi -pedantic
3: SFMLFLAGS= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
4: all: NBody
5:
6:
7: NBody: NBody.o universe.o
8:     $(CC) NBody.o universe.o -o NBody $(SFMLFLAGS)
9:
10: NBody.o: NBody.cpp universe.hpp
11:     $(CC) -c NBody.cpp $(CFLAGS)
12:
13: universe.o: universe.cpp universe.hpp
14:     $(CC) -c universe.cpp $(CFLAGS)
15:
16: clean:
17:     rm universe.o NBody.o NBody *~
```

```
1: #include <iostream>
2: #include <sstream>
3: #include <cmath>
4: #include <memory>
5: #include <SFML/Audio.hpp>
6: #include "universe.hpp"
7: using namespace std;
8: sf::RenderWindow window(sf::VideoMode(500, 500), "Universe");
9:
10:
11: int main(int argc, char* argv[])
12: {
13:
14:     sf::SoundBuffer buffer;
15:     //if(!buffer.loadFromFile("2001.wav"))
16:         //return -1;
17:
18:     sf::Sound sound;
19:     sound.setBuffer(buffer);
20:     sound.setVolume(100.f);
21:
22:     sf::Texture texture;
23:     if(!texture.loadFromFile("starfield.jpg"))
24:     {
25:         exit (1);
26:     }
27:
28:     sf::Sprite background(texture);
29:     int num;
30:     double radius;
31:     double time_end = atof(argv[1]);
32:     double time = atof(argv[2]);
33:
34:
35:     cin >> num;
36:     cin >> radius;
37:
38:     Body *objects;
39:     objects= new Body[num];
40:     for(int i = 0; i < num; i++)
41:     {
42:         cin >> objects[i];
43:         objects[i].set_radius(radius);
44:         objects[i].change_center();
45:     }
46:
47:     while(window.isOpen())
48:     {
49:         sf::Event event;
50:         while (window.pollEvent(event))
51:         {
52:             if (event.type == sf::Event::Closed)
53:             {
54:                 window.close();
55:             }
56:         }
57:         double time_start = 0;
58:         if(time_start < time_end)
59:         {
60:             for(int a = 0; a < num; a++)
61:             {
```

```
62:
63:                                objects[a].setforce(0); // set the force per
objects
64:                                for(int b = 0; b < num; b++)
65:                                {
66:
67:                                    if(a != b)
68:                                    {
69:                                        objects[a].Xdistance(objects
[b]);
70:                                        objects[a].Ydistance(objects
[b]); // find the distance between the planets
71:                                        objects[a].math(objects[b]);
// do the math for each planet to another
72:                                }
73:
74:                                }
75:                                objects[a].acc();
76:                                objects[a].vel(time);
77:                                objects[a].nextPos(time);
78:
79:                                }
80:                                time_start += time;
81:
82:                                }
83:
84:                                window.clear();
85:                                sound.play();
86:                                window.draw(background);
87:                                for(int i = 0; i < num; i++)
88:                                {
89:                                    objects[i].change_center(); // set the image center.
90:                                    window.draw(objects[i]);
91:                                    window.display();
92:                                }
93:                                for(int i = 0; i < num; i++)
94:                                {
95:                                    objects[i].print();
96:                                }
97:
98:                                }
99:
100:                                return 0;
101: }
```

```
1: #ifndef UNIVERSE_HPP_INCLUDED
2: #define UNIVERSE_HPP_INCLUDED
3:
4: #include <SFML/Graphics/Sprite.hpp>
5: #include <SFML/Graphics.hpp>
6: #include <iostream>
7: #include <string>
8: #include <fstream>
9: using namespace std;
10:
11: const double gravity = 6.67e-11;
12:
13: class Body : public sf::Drawable{
14: public:
15:
16: Body();
17: Body(double Xpos, double Ypos, double Xvel, double Yvel,
18: double mass, string Image_data);
19:
20: void set_radius(double radius);
21: void change_center();
22:
23: double Xdistance(Body &Body2);
24: double Ydistance(Body &Body2);
25: void math(Body &Body2);
26:
27: double getXforce();
28: double getYforce();
29: void setforce(double x);
30: void setdistance(double x);
31:
32: friend istream& operator>> (istream &inputStream, Body &x);
33:
34: void acc();
35: void vel(double seconds);
36: void nextPos(double seconds);
37:
38: void print();
39:
40: virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const;
41:
42: double _Xpos;
43: double _Ypos;
44: double _SpriteXpos;
45: double _SpriteYpos;
46: double _Xvel;
47: double _Yvel;
48: double _mass;
49: double _radius;
50: string _Image_data;
51:
52: double _Xdis;
53: double _Ydis;
54: double _Xacc;
55: double _Yacc;
56: double _Xfor;
57: double _Yfor;
58:
59:
60: sf::Sprite _sprite;
61: sf::Texture _texture;
```

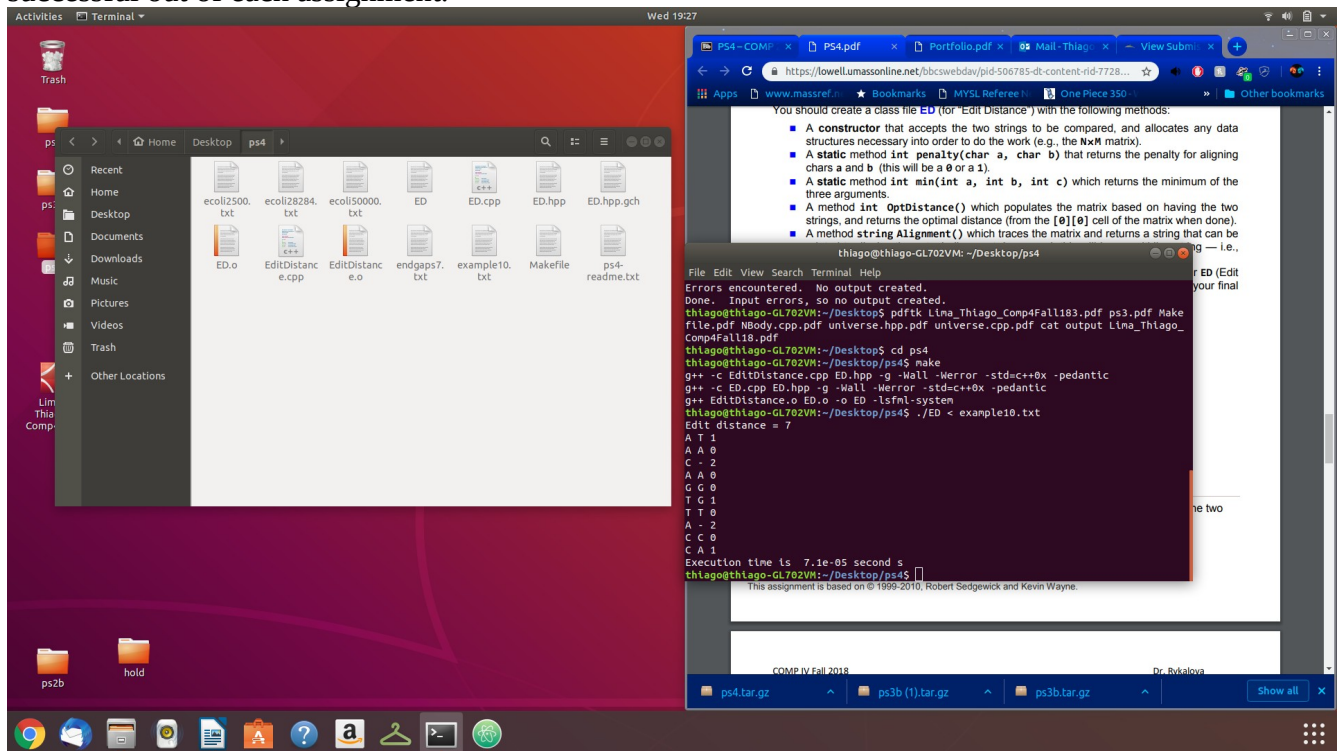
```
62: };  
63:  
64: #endif
```

```
1: #include <SFML/Graphics/Sprite.hpp>
2: #include <SFML/Graphics.hpp>
3: #include <iostream>
4: #include <string>
5: #include <cmath>
6: #include "universe.hpp"
7: using namespace std;
8:
9: Body::Body() {
10: return;
11: }
12:
13: Body::Body(double Xpos, double Ypos, double Xvel, double Yvel,
14: double mass, string Image_data){
15: _Xpos = Xpos;
16: _Ypos=Ypos;
17: _Xvel=Xvel;
18: _Yvel=Yvel;
19: _mass=mass;
20: _Image_data=Image_data;
21: }
22:
23: istream& operator >> (istream &in, Body &x){
24: in >> x._Xpos;
25: in >> x._Ypos;
26: in >> x._Xvel;
27: in >> x._Yvel;
28: in >> x._mass;
29: in >> x._Image_data;
30: x._texture.loadFromFile(x._Image_data);
31: x._sprite.setPosition(x._Xpos, x._Ypos);
32: x._sprite.setTexture(x._texture);
33: return in;
34: }
35:
36: void Body::set_radius(double radius){
37: _radius = radius;
38: }
39:
40: void Body::change_center(){
41: _SpriteXpos = ( (_Xpos / _radius) * (500 / 2) ) + (500 / 2);
42: _SpriteYpos = ( ((_Ypos * -1) / _radius) * (500 / 2) ) + (500 / 2);
43: _sprite.setPosition(_SpriteXpos, _SpriteYpos);
44: }
45:
46: void Body::draw(sf::RenderTarget& target, sf::RenderStates states) const{
47: target.draw(_sprite);
48: }
49:
50: double Body::Xdistance(Body &Body2){
51: _Xdis = Body2._Xpos - _Xpos;
52: return _Xdis;
53: }
54:
55: double Body::Ydistance(Body &Body2){
56: _Ydis = Body2._Ypos - _Ypos;
57: return _Ydis;
58: }
59:
60: double Body::getXforce(){
61: return _Xfor;
```

```
62: }
63: double Body::getYforce() {
64: return _Yfor;
65: }
66: void Body::setforce(double x) {
67: _Xfor = x;
68: _Yfor = x;
69: }
70:
71: void Body::setdistance(double x) {
72: _Xdis = x;
73: _Ydis = x;
74: }
75:
76: void Body::math(Body &Body2) {
77: double force;
78: force = (gravity * _mass * Body2._mass) / (sqrt(pow(_Xdis, 2) + pow(_Ydis,
2)));
79: _Xfor += force * (_Xdis / (pow(_Xdis, 2) + pow(_Ydis, 2)) );
80: _Yfor += force * (_Ydis / (pow(_Xdis, 2) + pow(_Ydis, 2)) );
81: }
82:
83: void Body::acc() {
84: _Xacc = _Xfor / _mass ;
85: _Yacc = _Yfor / _mass ;
86: }
87: void Body::vel(double seconds) {
88: _Xvel = _Xvel + (_Xacc * seconds) ;
89: _Yvel = _Yvel + (_Yacc * seconds) ;
90: }
91: void Body::nextPos(double seconds) {
92: _Xpos = _Xpos + (_Xvel * seconds) ;
93: _Ypos = _Ypos + (_Yvel * seconds);
94: }
95:
96: void Body::print() {
97: cout << _Xpos << " " << _Ypos << " " ;
98: cout << _Xvel << " " << _Yvel << " " << _Image_data << endl;
99: }
100:
```

PS4: DNA Sequence Alignment:

In this programming assignment i was asked for creating a DNA Sequencing alignment program which we take 2 sets of strings and compare the sets of strings together to find the most optimal editing distance, which is the total cost it take to edit one string to most look like the second string. We were giving many tools to help us with this, like sequencing grid in which we understood how editing distances found and it was a basic diagram for how we were supposed to implement our program. I create an implementation using the 2 dimensional matrix that would find the editing distance most like how we worked on it in class using the grid and I implemented an algorithm that would search the grid in adjacent boxes and find the most optimal path to take to have the smallest editing distance. I not only learn how to work with that algorithm but in this assignment I learned how to better work with matrixes and Vectors which are very useful variable implementations. this program will work extremely well and i understood how to implement this program very quickly which I think make this program most successful out of each assignment.



The screenshot displays a Linux desktop environment. On the left, a file manager window shows the contents of a directory named 'ps4' on the desktop. The files include 'ecoli2500.txt', 'ecoli28284.txt', 'ecoli5000.txt', 'ED', 'ED.cpp', 'ED.hpp', 'ED.hpp.gch', 'ED.o', 'EditDistance.cpp', 'EditDistance.o', 'endgaps7.txt', 'example10.txt', 'Makefile', and 'ps4-readme.txt'. In the center, a web browser window is open to a page titled 'PS4 - COM' with a URL starting with 'https://lowellumassonline.net'. The page contains instructions for creating a class file 'ED' and lists several methods to implement: a constructor, a static method 'penalty', a static method 'min', a method 'OptDistance', and a method 'string Alignment'. On the right, a terminal window shows the execution of the program. The user navigates to the 'ps4' directory, compiles the files with 'g++', and runs the program with './ED < example10.txt'. The output shows the editing distance for a specific sequence alignment, which is 7. The terminal also displays the execution time as 7.1e-05 seconds.

```
thiago@thiago-GL702VM: ~/Desktop/ps4
thiago@thiago-GL702VM:~/Desktop/ps4$ g++ -c EditDistance.cpp ED.hpp -g -Wall -Werror -std=c++0x -pedantic
g++ -c ED.cpp ED.hpp -g -Wall -Werror -std=c++0x -pedantic
g++ EditDistance.o ED.o -o ED -lsfml-system
thiago@thiago-GL702VM:~/Desktop/ps4$ ./ED < example10.txt
Edit distance = 7
A T 1
A A 0
C - 2
A A 0
G G 0
T G 1
T T 0
A - 2
C C 0
C A 1
Execution time is 7.1e-05 second s
thiago@thiago-GL702VM:~/Desktop/ps4$
```



```
1: CC= g++
2: CFLAGS= -g -Wall -Werror -std=c++0x -pedantic
3: SFLAGS= -lsfml-system
4:
5: all:          ED
6:
7: ED: EditDistance.o ED.o
8:      $(CC) EditDistance.o ED.o -o ED $(SFLAGS)
9:
10: EditDistance.o: EditDistance.cpp ED.hpp
11:      $(CC) -c EditDistance.cpp ED.hpp $(CFLAGS)
12:
13: ED.o: ED.cpp ED.hpp
14:      $(CC) -c ED.cpp ED.hpp $(CFLAGS)
15:
16: clean:
17:      rm *.o
18:      rm *.gch
19:      rm ED
```

```
1: #include "ED.hpp"
2:
3: using namespace std;
4:
5: int main(int argc, char* argv[]){
6:
7:     sf::Time t;
8:     sf::Clock clock;
9:
10:    string x, y;
11:    cin >> x >> y;
12:    ED ed(x, y);
13:    t = clock.getElapsedTime();
14:
15:    int distance = ed.Optdistance();
16:    cout << "Edit distance = " << distance ;
17:    string last;
18:    last = ed.Alignment();
19:    int j = last.length();
20:    int h = j / 3 ;
21:    for(int i = 0; i < h; i++){
22:
23:        cout << last[i]<<' ' << last[h + i]<<' ' << last[(h *2) + i ] <<endl;
24:    }
25:    cout << "Execution time is  " << t.asSeconds() << " second s \n";
26:
27:    return 0;
28: }
```

```
1: #ifndef ED_HPP
2: #define ED_HPP
3:
4: #include <iostream>
5: #include <string>
6: #include <vector>
7: #include <SFML/System.hpp>
8:
9:
10: using namespace std;
11:
12: class ED{
13: public:
14: ED(string one, string two);
15: static int penalty(char a, char b);
16: static int min(int a, int b, int c);
17: int Optdistance();
18: string Alignment();
19:
20: string _x;
21: string _y;
22: vector <vector<int>> matrix;
23: int n, m;
24: };
25:
26: #endif
```

```
1: #include "ED.hpp"
2:
3: using namespace std;
4:
5: ED::ED(string one, string two){
6:     _x = one;
7:     _y = two;
8: }
9:
10: int ED::penalty(char a, char b){
11:     if(a==b)
12:     {
13:         return 0;
14:     }
15:     return 1;
16: }
17:
18: int ED::min(int a, int b, int c){
19:
20:     if(b < a ){
21:         if(b<c){
22:             a = b;
23:         }
24:     }
25:
26:     if(c < a ){
27:         if(c<b){
28:             a = c;
29:         }
30:     }
31:
32:     return a;
33: }
34:
35: int ED::Optdistance(){
36:
37:     m = _x.length();
38:     n = _y.length();
39:
40:     vector<int> temp;
41:     for(int i = 0; i <= m; i++)
42:     {
43:         matrix.push_back(temp);
44:         for(int j = 0; j <= n; j++)
45:         {
46:             matrix.at(i).push_back(-1);
47:         }
48:     }
49: }
50:
51: for(int i = 0; i <= n; i++)
52: {
53:     matrix[m][i] = 2 * (n - i);
54: }
55:
56: for(int i = 0; i <= m; i++)
57: {
58:     matrix[i][n] = 2 * (m - i);
59: }
60:
61: for(int j = n-1; j >= 0; j--)
```

```
62: {
63:     for(int i = m-1; i >= 0; i--)
64:     {
65:         int a = matrix[i+1][j+1] + penalty(_x[i], _y[j]);
66:         int b = matrix[i+1][j] + 2;
67:         int c = matrix[i][j+1] + 2;
68:         matrix[i][j] = min(a, b, c);
69:     }
70: }
71: int ans = matrix[0][0];
72: return ans;
73: }
74:
75: string ED::Alignment() {
76:     cout << endl;
77:     string x;
78:     string y;
79:     string z;
80:     int i=0, j=0;
81:     while(i<m || j<n)
82:     {
83:         if(i == m)
84:         {
85:             y += _x[j-1];
86:             x += '-';
87:             z += '2';
88:             j++;
89:         }
90:
91:         else if(j == n)
92:         {
93:             y += '-';
94:             x += _y[i-1];
95:             z += '2';
96:             i++;
97:         }
98:
99:         else
100:         {
101:             if( (matrix[i][j]) == (matrix.at(i+1).at(j+1) + penalty(_x[i
], _y[j]) ) )
102:             {
103:                 y += _y[j];
104:                 x += _x[i];
105:                 if(_x[i] == _y[j])
106:                 {
107:                     z += '0';
108:
109:                 }
110:                 else
111:                 {
112:                     z += '1';
113:                 }
114:                 i++, j++;
115:             }
116:
117:             else if( (matrix[i][j]) == (matrix.at(i+1).at(j) + 2 ) )
118:             {
119:                 y += '-';
120:                 x += _x[i];
121:                 z += '2';
```

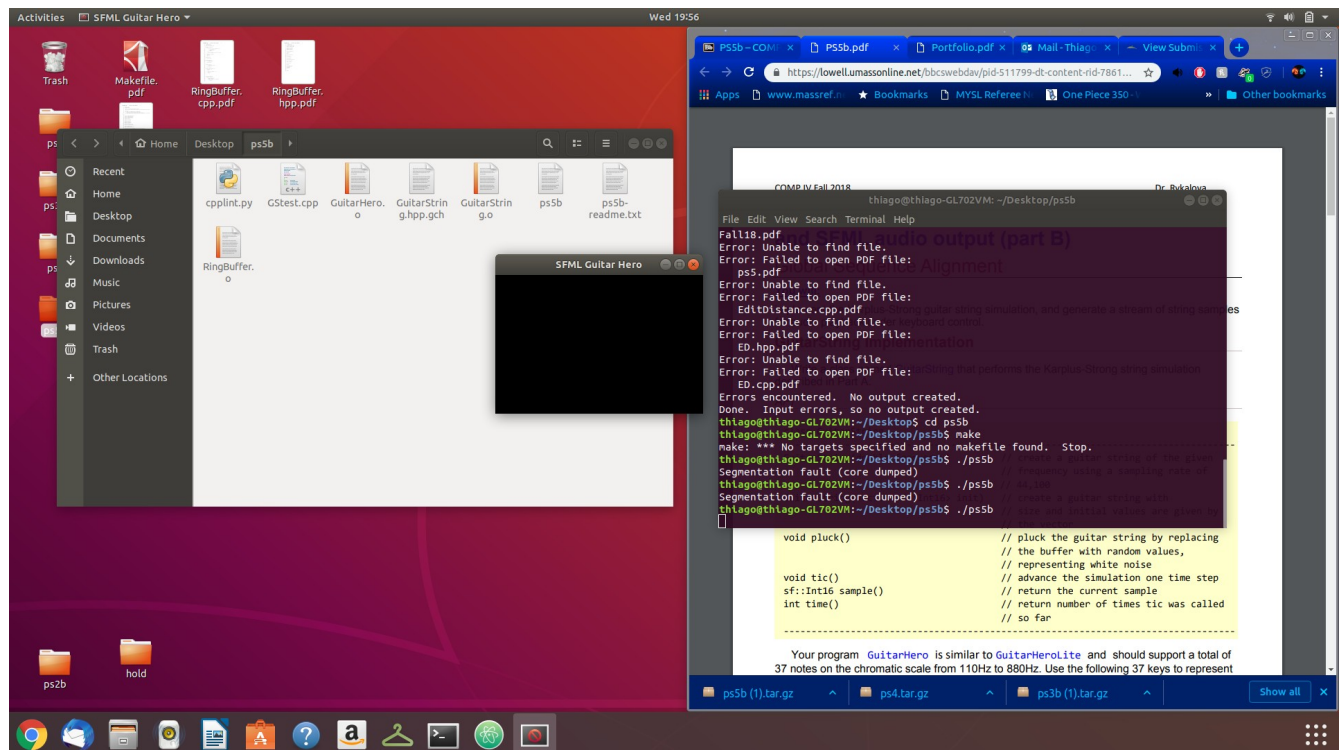
```
122:                i++;
123:            }
124:
125:            else if( (matrix[i][j]) == (matrix.at(i).at(j+1) + 2 ) )
126:            {
127:                y += _y[j];
128:                x += '-';
129:                z += '2';
130:                j++;
131:            }
132:        }
133:    }
134:    return x + y + z;
135: }
```

PS5: Guitar Hero: RingBuffer implementation with unit tests and exceptions (Part A):

In the first part of the guitar hero assignment I implemented multiple functions at word help with creating a fully functional guitar hero program. these functions were made me to work with the ring buffer that i had created and were made to check certain things about the ring buffer like to check it size want to add and remove certain elements inside the ring buffer. This program worked successfully and I learn how to work with ring buffer functions for it. The ring buffer was implemented as a implementation Q or priority queue and the most difficult algorithm to create were the algorithms to add or delete items in the queue these work divine basically popping off an element or adding a new element into the queue from one that I was given and use the first and last positions in the ring buffer.

PS5: Guitar Hero: GuitarString implementation and SFML audio output (part B):

In the second part of the guitar hero assignment I worked with the functions from the guitar string class to create the sounds that each key would create, I have to create the guitar string class and the functions within it like plug that would generate each noise, tar hero program would work with 37 keys on my computer and create noises from 110 Hertz to 880 Hertz following closely to what A piano would sound like. When are the hardest and most important algorithms that were implemented in this program was the clock function which would fill up the ring buffer that I had created with a random numbers. When good luck function activate it would fill up the ring buffer and give the guitar string class a ring buffer to work With and create the sounds and frequencies I would need. I learn how to work with sound functions in this program and How to create audio outputs in classes that would work with these other parts as well as a better understanding of priorities queues.



```
1: CC=g++
2:
3: CFLAGS=-c -g -Wall -Werror -ansi -pedantic
4:
5: LIBS= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio -lboost_unit_
test_framework
6:
7: all: ps5b
8:
9: ps5b: GuitarHero.o GuitarString.o RingBuffer.o GStest.o
10:      $(CC) -o ps5b GuitarHero.o GuitarString.o RingBuffer.o $(LIBS)
11:
12: GuitarHero.o: GuitarHero.cpp GuitarString.hpp
13:      $(CC) $(CFLAGS) GuitarHero.cpp
14:
15: GuitarString.o: GuitarString.cpp GuitarString.hpp
16:      $(CC) $(CFLAGS) GuitarString.cpp
17:
18: RingBuffer.o: RingBuffer.cpp RingBuffer.hpp
19:      $(CC) $(CFLAGS) RingBuffer.cpp
20:
21: clean:
22:      rm *.o ps5b
```



```
1: /*
2:   Thiago Lima
3:   Copyright 2018
4:
5:   based on Princeton's GuitarHeroLite.java
6:   www.cs.princeton.edu/courses/archive/fall13/cos126/assignments/guitar.html
7:
8:   build with
9:   g++ -Wall -c GuitarHeroLite.cpp -lsfml-system \
10:      -lsfml-audio -lsfml-graphics -lsfml-window
11:   g++ -Wall GuitarHeroLite.o RingBuffer.o GuitarString.o \
12:      -o GuitarHeroLite -lsfml-system -lsfml-audio -lsfml-graphics -lsfml-windo
w
13: */
14:
15: #include <SFML/Graphics.hpp>
16: #include <SFML/System.hpp>
17: #include <SFML/Audio.hpp>
18: #include <SFML/Window.hpp>
19:
20: #include <math.h>
21: #include <limits.h>
22:
23: #include <iostream>
24: #include <string>
25: #include <exception>
26: #include <stdexcept>
27: #include <vector>
28:
29: #include "RingBuffer.hpp"
30: #include "GuitarString.hpp"
31:
32: #define CONCERT_A 220.0
33: #define SAMPLES_PER_SEC 44100
34:
35: std::vector<sf::Int16> makeSamplesFromString(GuitarString gs) {
36:     std::vector<sf::Int16> samples;
37:
38:
39:     gs.pluck();
40:     int duration = 8; // seconds
41:     int i;
42:     for (i= 0; i < SAMPLES_PER_SEC * duration; i++) {
43:         gs.tic();
44:         samples.push_back(gs.sample());
45:     }
46:
47:     return samples;
48: }
49:
50:
51: int main() {
52:     sf::RenderWindow window(sf::VideoMode(300, 200), "SFML Guitar Hero");
53:     sf::Event event;
54:     double freq;
55:     unsigned int index;
56:     std::string Keyboard = "q2we4r5ty7u8i9op-=[zxdcfvghbnjmk,.;/' ";
57:     vector <vector <sf::Int16> > samples(37);
58:     vector<sf::SoundBuffer> sound_samples(37);
59:     vector <sf::Sound> sound(37);
60:
```

```
61:
62: // we're reusing the freq and samples vars, but
63: // there are separate copies of GuitarString, SoundBuffer, and Sound
64: // for each note
65: //
66: // GuitarString is based on freq
67: // samples are generated from GuitarString
68: // SoundBuffer is loaded from samples
69: // Sound is set to SoundBuffer
70:
71: for(int i = 0; i < 37; i++) {
72:     freq = (i - 24.0) / 12.0;
73:     freq = pow(2.0, freq);
74:     freq *= CONCERT_A;
75:     samples[i] = makeSamplesFromString(GuitarString(freq));
76:     if (!(sound_samples[i]).loadFromSamples(&samples[i][0]
77:         , samples[i].size(), 2, SAMPLES_PER_SEC))
78:         throw std::runtime_error("failed to load from samples");
79:     (sound[i]).setBuffer(sound_samples[i]);
80: }
81:
82: while (window.isOpen()) {
83:     while (window.pollEvent(event)) {
84:         switch (event.type) {
85:             case sf::Event::Closed:
86:                 window.close();
87:                 break;
88:
89:             case sf::Event::TextEntered:
90:                 index = Keyboard.find(event.text.unicode);
91:                 if (index != std::string::npos)
92:                     sound[index].play();
93:                 break;
94:             default:
95:                 break;
96:         }
97:
98:         window.clear();
99:         window.display();
100:     }
101: }
102: return 0;
103: }
```

```
1: // Thiago Lima
2: // Copyright 2018
3:
4: #ifndef GuitarString_HPP
5: #define GuitarString_HPP
6:
7: #include "RingBuffer.hpp"
8:
9:
10: #include <stdlib.h>
11: #include <iostream>
12: #include <string>
13: #include <vector>
14: #include <stdint.h>
15: #include <math.h>
16: #include <vector>
17:
18:
19: using namespace std;
20:
21: class GuitarString {
22: public:
23:
24:   GuitarString(double frequency);
25:
26:   GuitarString(vector<int16_t> init);
27:
28:   ~GuitarString();
29:
30:   void pluck();
31:
32:   void tic();
33:
34:   double sample();
35:
36:   int time();
37:
38: private:
39:
40:   int step;
41:   int frequency;
42:   RB *RingBuffer;
43:
44: };
45:
46: #endif
```

```
1: // Thiago Lima
2: // Copyright 2018
3:
4: #include <vector>
5: #include "GuitarString.hpp"
6: #include "RingBuffer.hpp"
7:
8:
9:
10: GuitarString::GuitarString(double frequency) {
11:     step = 0;
12:     frequency = ceil(44100 / frequency);
13:     RingBuffer = new RB(frequency);
14:
15:     for (int i = 0; i < 44100; i++) {
16:         RingBuffer->enqueue(0);
17:     }
18: }
19:
20: GuitarString::GuitarString(vector<int16_t> init) {
21:     step = 0;
22:     RingBuffer = new RB(init.size());
23:
24:     for (unsigned int i = 0; i < init.size(); i++) {
25:         RingBuffer->enqueue(init[i]);
26:     }
27: }
28:
29: GuitarString::~GuitarString() {
30:     delete RingBuffer;
31: }
32:
33: void GuitarString::pluck() {
34:     while (!RingBuffer->isEmpty()) {
35:         RingBuffer->dequeue();
36:     }
37:
38:     while (!RingBuffer->isFull()) {
39:         unsigned int next;
40:         RingBuffer->enqueue((int16_t)(rand_r(&next) & 0xffff));
41:     }
42: }
43:
44: void GuitarString::tic() {
45:     int16_t x = RingBuffer->dequeue();
46:     int16_t y = RingBuffer->peek();
47:
48:     int16_t next = x + y;
49:     next /= 2;
50:     next *= 0.996;
51:
52:     RingBuffer->enqueue(next);
53:     step++;
54: }
55:
56: double GuitarString::sample() {
57:     return RingBuffer->peek();
58: }
59:
60: int GuitarString::time() {
61:     return step;
```

```
62: }
```

```
1: // Thiago Lima
2: // Copyright 2018
3:
4: #ifndef RingBuffer_HPP
5: #define RingBuffer_HPP
6:
7: #include <iostream>
8: #include <string>
9: #include <vector>
10: #include <stdint.h>
11:
12:
13: using namespace std;
14:
15: class RB {
16: public:
17:
18:     RB (int capacity);
19:
20:     int size();
21:
22:     bool isEmpty();
23:
24:     bool isFull();
25:
26:     void enqueue(int16_t x);
27:
28:     int16_t dequeue();
29:
30:     int16_t peek();
31:
32:     void print();
33:
34: private:
35:
36:     vector<int> ring;
37:     int _capacity;
38:     int _size;
39:     int first;
40:     int last;
41: };
42:
43:
44: #endif
```

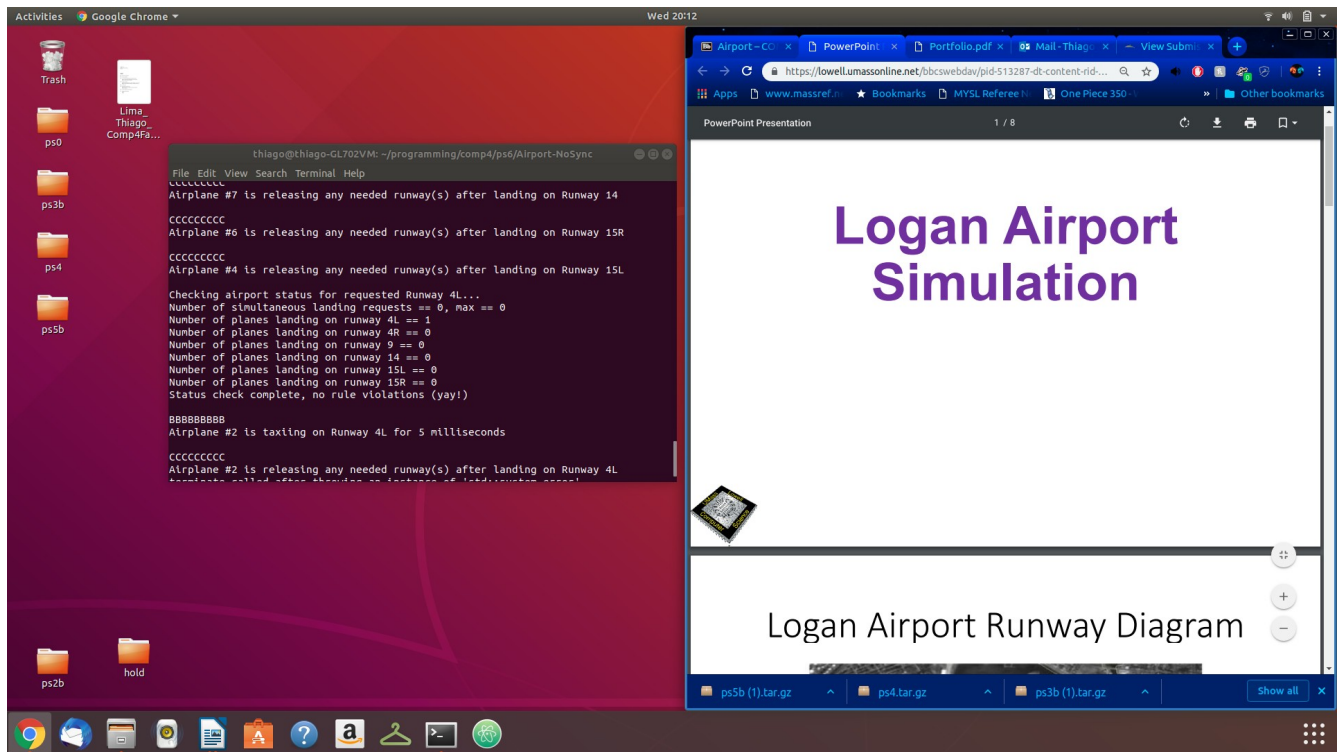
```
1: // Thiago Lima
2: // Copyright 2018
3:
4: #include "RingBuffer.hpp"
5:
6: RB::RB(int capacity) {
7:     try {
8:         for (int i = 0; i < capacity; i++) {
9:             ring.push_back(-1);
10:        }
11:        if (capacity == 0) {
12:            throw 1;
13:        }
14:    } catch (int x) {
15:        cout << "capacity can't be 0" << endl;
16:    }
17:    _capacity = capacity;
18:    first = 0;
19:    last = 0;
20:    _size = 0;
21: }
22:
23: int RB::size() {
24:     return _size;
25: }
26:
27: bool RB::isEmpty() {
28:     if (size() == 0) {
29:         return true;
30:     }
31:     return false;
32: }
33:
34: bool RB::isFull() {
35:     if (_capacity == size()) {
36:         return true;
37:     }
38:     return false;
39: }
40:
41: void RB::enqueue(int16_t x) {
42:     ring[last] = x;
43:     _size++;
44:
45:     if (last == (_capacity - 1)) {
46:         last = 0;
47:     } else {
48:         last++;
49:     }
50: }
51:
52:
53: int16_t RB::dequeue() {
54:     int16_t temp = -1;
55:     try {
56:         temp = ring[first];
57:         ring[first] = 0;
58:         first = (first + 1) % _capacity;
59:         if (size() == 0) {
60:             throw 2;
61:         }

```

```
62:     }catch (int x) {
63:         cout << "can't dequeue with a 0 size" << endl;
64:     }
65:     if (size() == 0) {
66:         _size = 0;
67:     } else {
68:         _size--;
69:     }
70:     return temp;
71: }
72:
73: int16_t RB::peek() {
74:     int16_t temp;
75:     try {
76:         temp = ring[first];
77:         if (size() == 0) {
78:             throw 3;
79:         }
80:     }catch (int x) {
81:         cout << "can't peek with size 0" << endl;
82:     }
83:     return temp;
84: }
85:
86: void RB::print() {
87:     for (int i = 0; i <= _capacity - 1; i++) {
88:         cout << ring[i] << endl;
89:     }
90:     cout << "first: " << first << endl;
91:     cout << "last: " << last << endl;
92:     cout << endl;
93: }
```


Airport Simulation Project:

In this program I have to use mutex locks and concurrency to create a system that would allow certain claims to land and space in a airport run and not crash this program was to run for 15 minutes and to create a file called airport server that would handle all of the landing request. This program was extremely frustrating and very confusing but I learned a lot on how to work with threads and processes as well as concurrency. To solve this program I created variables of mutex locks and conditions that requested permission for landing and switch statements that were tell me which airport runways were in use and which one of the runways would overlap with other runways. The most important algorithm that I had to implement was the switch case statement that checked which runway was working and blocked off every other runway that was in use and conflicting runways too not allowed them to crash my program it worked very well and did not create any crashes.



```

1: /**
2: *   AirportServer.h
3: *   This class defines the methods called by the Airplanes
4: */
5:
6: #ifndef AIRPORT_SERVER_H
7: #define AIRPORT_SERVER_H
8:
9: #include <mutex>
10: #include <random>
11: #include <condition_variable>
12:
13:
14: #include "AirportRunways.h"
15:
16:
17:
18: class AirportServer
19: {
20: public:
21:
22:     /**
23:     *   Default constructor for AirportServer class
24:     */
25:     AirportServer()
26:     {
27:         // ***** Initialize any Locks and/or Condition Variables her
e as necessary *****
28:
29:     } // end AirportServer default constructor
30:
31:
32:     /**
33:     *   Called by an Airplane when it wishes to land on a runway
34:     */
35:     void reserveRunway(int airplaneNum, AirportRunways::RunwayNumber run
way);
36:
37:     /**
38:     *   Called by an Airplane when it is finished landing
39:     */
40:     void releaseRunway(int airplaneNum, AirportRunways::RunwayNumber run
way);
41:
42:
43: private:
44:
45:     // Constants and Random number generator for use in Thread sleep cal
ls
46:     static const int MAX_TAXI_TIME = 10; // Maximum time the airplane wi
ll occupy the requested runway after landing, in milliseconds
47:     static const int MAX_WAIT_TIME = 100; // Maximum time between landin
gs, in milliseconds
48:
49:     /**
50:     *   Declarations of mutexes and condition variables
51:     */
52:     mutex runwaysMutex; // Used to enforce mutual exclusion for acquirin
g & releasing runways
53:     /**
54:     *   ***** Add declarations of your own Locks and Condition Variables

```

```
here *****
55:          */
56:          condition_variable cv;
57:
58:
59: }; // end class AirportServer
60:
61: #endif
```

```
1: #include <iostream>
2: #include <thread>
3:
4: #include "AirportServer.h"
5:
6: int num_4L = 0;
7: int num_4R = 0;
8: int num_15R = 0;
9: int num_15L = 0;
10: int num_9 = 0;
11: int num_14 = 0;
12:
13: /**
14:  *   Called by an Airplane when it wishes to land on a runway
15:  */
16: void AirportServer::reserveRunway(int airplaneNum, AirportRunways::RunwayNum
ber runway)
17: {
18:     // Acquire runway(s)
19:     { // Begin critical region
20:
21:         unique_lock<mutex> runwaysLock(runwaysMutex);
22:
23:         {
24:             lock_guard<mutex> lk(AirportRunways::checkMutex);
25:
26:             cout << "Airplane #" << airplaneNum << " is acquirin
g any needed runway(s) for landing on Runway "
27:                 << AirportRunways::runwayName(runway) << en
dl;
28:         }
29:
30:         /**
31:          *   ***** Add your synchronization here! *****
32:          */
33:
34:         switch (runway)
35:         {
36:             case AirportRunways::RUNWAY_4L:
37:                 cv.wait(runwaysLock, [] {return num_4L == 0;})
);
38:                 num_4L++;
39:                 num_15R++;
40:                 num_15L++;
41:                 break;
42:
43:             case AirportRunways::RUNWAY_4R:
44:                 cv.wait(runwaysLock, [] {return num_4R == 0;})
);
45:                 num_4R++;
46:                 num_15R++;
47:                 num_15L++;
48:                 num_9++;
49:                 break;
50:
51:             case AirportRunways::RUNWAY_9:
52:                 cv.wait(runwaysLock, [] {return num_9 == 0;})
;
53:                 num_9++;
54:                 num_15R++;
55:                 num_4R++;
```

```

56:                                     break;
57:
58:                                     case AirportRunways::RUNWAY_14:
59:                                         cv.wait(runwaysLock, [] {return num_14 == 0; }
);
60:                                     num_14++;
61:                                     break;
62:
63:                                     case AirportRunways::AirportRunways::RUNWAY_15L:
64:                                         cv.wait(runwaysLock, [] {return num_15L == 0;
});
65:                                     num_4L++;
66:                                     num_4R++;
67:                                     num_15L++;
68:                                     break;
69:
70:                                     case AirportRunways::RUNWAY_15R:
71:                                         cv.wait(runwaysLock, [] {return num_15R == 0;
});
72:                                     num_15R++;
73:                                     num_4L++;
74:                                     num_4R++;
75:                                     num_9++;
76:                                     break;
77:
78:                                     default:
79:                                         cout << "Unknown runway ";
80:                                         break;
81:                                     }
82:                                     // Check status of the airport for any rule violations
83:                                     AirportRunways::checkAirportStatus(runway);
84:
85:                                     //runwaysLock.unlock();
86:                                     cout << "\nBBBBBBBBBB\n";
87:
88:                                     } // End critical region
89:
90:                                     // obtain a seed from the system clock:
91:                                     unsigned seed = std::chrono::system_clock::now().time_since_epoch().
count();
92:                                     std::default_random_engine generator(seed);
93:
94:                                     // Taxi for a random number of milliseconds
95:                                     std::uniform_int_distribution<int> taxiTimeDistribution(1, MAX_TAXI_
TIME);
96:                                     int taxiTime = taxiTimeDistribution(generator);
97:
98:                                     {
99:                                         lock_guard<mutex> lk(AirportRunways::checkMutex);
100:
101:                                         cout << "Airplane #" << airplaneNum << " is taxiing on Runwa
y " << AirportRunways::runwayName(runway)
102:                                             << " for " << taxiTime << " milliseconds\n";
103:                                     }
104:
105:                                     std::this_thread::sleep_for(std::chrono::milliseconds(taxiTime));
106:
107:                                     } // end AirportServer::reserveRunway()
108:
109:
110:                                     /**

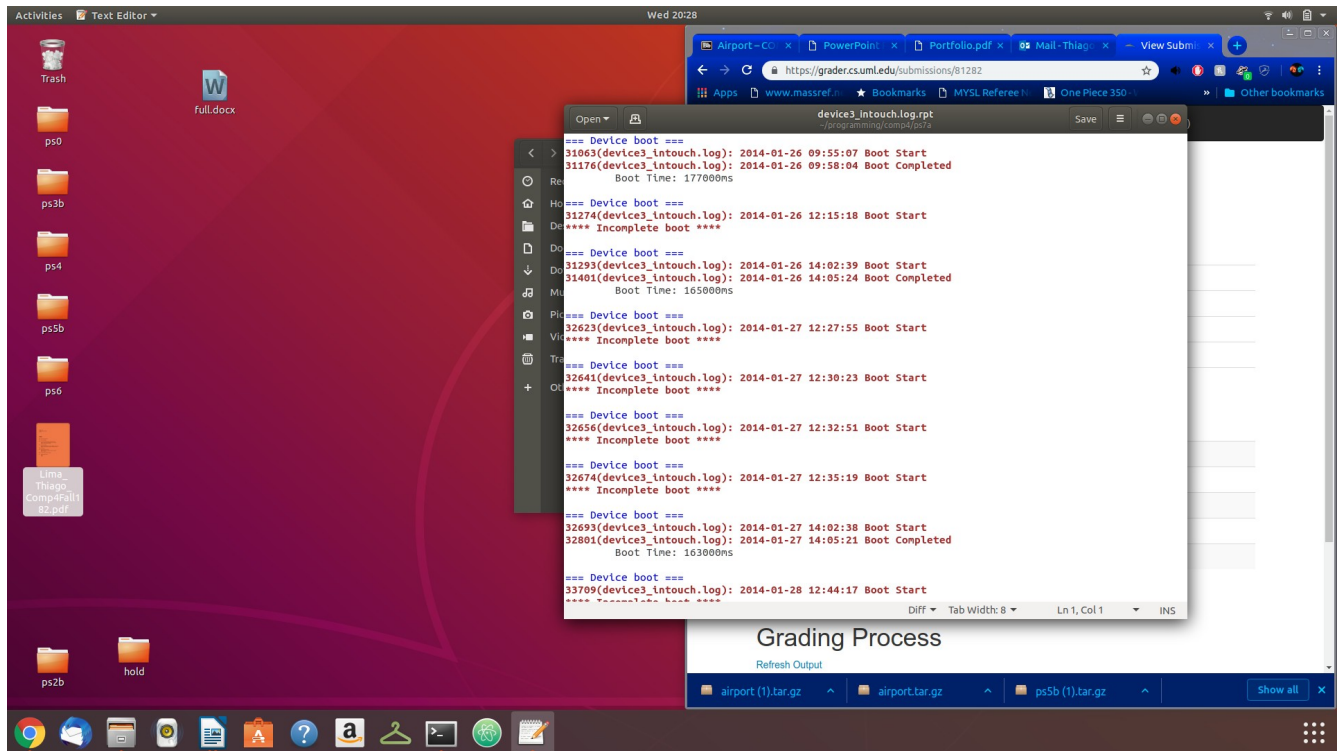
```

```
111:  *   Called by an Airplane when it is finished landing
112:  */
113: void AirportServer::releaseRunway(int airplaneNum, AirportRunways::RunwayNum
ber runway)
114: {
115:     cout << "\nCCCCCCCCC\n";
116:     // Release the landing runway and any other needed runways
117:     { // Begin critical region
118:
119:         unique_lock<mutex> runwaysLock(runwaysMutex);
120:
121:         {
122:             lock_guard<mutex> lk(AirportRunways::checkMutex);
123:
124:             cout << "Airplane #" << airplaneNum << " is releasin
g any needed runway(s) after landing on Runway "
125:                 << AirportRunways::runwayName(runway) << en
dl;
126:         }
127:
128:         /**
129:         *   ***** Add your synchronization here! *****
130:         */
131:         switch (runway)
132:         {
133:             case AirportRunways::RUNWAY_4L:
134:                 runwaysLock.unlock();
135:                 num_4L--;
136:                 num_15R--;
137:                 num_15L--;
138:                 cv.notify_all();
139:                 break;
140:
141:             case AirportRunways::RUNWAY_4R:
142:                 runwaysLock.unlock();
143:                 num_4R--;
144:                 num_15R--;
145:                 num_15L--;
146:                 num_9--;
147:                 cv.notify_all();
148:                 break;
149:
150:             case AirportRunways::RUNWAY_9:
151:                 runwaysLock.unlock();
152:                 num_9--;
153:                 num_15R--;
154:                 num_4R--;
155:                 cv.notify_all();
156:                 break;
157:
158:             case AirportRunways::RUNWAY_14:
159:                 runwaysLock.unlock();
160:                 num_14--;
161:                 cv.notify_all();
162:                 break;
163:
164:             case AirportRunways::AirportRunways::RUNWAY_15L:
165:                 runwaysLock.unlock();
166:                 num_4L--;
167:                 num_4R--;
168:                 num_15L--;
```

```
169:                                     cv.notify_all();
170:                                     break;
171:
172:                                     case AirportRunways::RUNWAY_15R:
173:                                         runwaysLock.unlock();
174:                                         num_15R--;
175:                                         num_4L--;
176:                                         num_4R--;
177:                                         num_9--;
178:                                         cv.notify_all();
179:                                         break;
180:
181:                                     default:
182:                                         cout << "Unknown runway ";
183:                                         break;
184:                                     }
185:
186:                                     // Update the status of the airport to indicate that the lan
ding is complete
187:                                     AirportRunways::finishedWithRunway(runway);
188:
189:                                     runwaysLock.unlock();
190:
191:                                     } // End critical region
192:
193:                                     // obtain a seed from the system clock:
194:                                     unsigned seed = std::chrono::system_clock::now().time_since_epoch().
count();
195:                                     std::default_random_engine generator(seed);
196:
197:                                     // Wait for a random number of milliseconds before requesting the ne
xt landing for this Airplane
198:                                     std::uniform_int_distribution<int> waitTimeDistribution(1, MAX_WAIT_
TIME);
199:                                     int waitTime = waitTimeDistribution(generator);
200:
201:                                     {
202:                                         lock_guard<mutex> lk(AirportRunways::checkMutex);
203:
204:                                         cout << "Airplane #" << airplaneNum << " is waiting for " <<
waitTime << " milliseconds before landing again\n";
205:                                     }
206:
207:                                     std::this_thread::sleep_for(std::chrono::milliseconds(waitTime));
208:
209:                                     } // end AirportServer::releaseRunway()
```

Kronos Time Clock: Introduction to Regular Expression Parsing:

In this program I had to create a program to take in a few kronos time clock log and create an out put file of the logs with boot information like when the boot started and the report that goes with it, if the boot was successful or if the boot failed. If the boot was successful when is ended how long it was from start to finish. The way I completed the assignment was to use the regex library creating a number of strings that I could match with strings in the log files I was given, the string would match the string start and check to find a end string before the next start string. This assignment was pretty easy and I learned to work with regex and the regex library.



The screenshot displays a Linux desktop environment. A text editor window titled "device3_intouch.log.rpt" is open, showing a list of boot logs. The logs are organized into sections, each starting with "=== Device boot ===". Each section contains two lines of log data: the first line shows the device ID, log file path, date, time, and "Boot Start"; the second line shows the device ID, log file path, date, time, and "Boot Completed". Below each pair of lines, the "Boot Time" is listed in milliseconds. The logs are as follows:

| Device ID | Log File | Date | Time | Status | Boot Time (ms) |
|-----------|---------------------|------------|----------|----------------|----------------|
| 31063 | device3_intouch.log | 2014-01-26 | 09:55:07 | Boot Start | |
| 31176 | device3_intouch.log | 2014-01-26 | 09:58:04 | Boot Completed | 177800ms |
| 31274 | device3_intouch.log | 2014-01-26 | 12:15:18 | Boot Start | |
| 31293 | device3_intouch.log | 2014-01-26 | 14:02:39 | Boot Start | |
| 31401 | device3_intouch.log | 2014-01-26 | 14:05:24 | Boot Completed | 165800ms |
| 32623 | device3_intouch.log | 2014-01-27 | 12:27:55 | Boot Start | |
| 32641 | device3_intouch.log | 2014-01-27 | 12:30:23 | Boot Start | |
| 32656 | device3_intouch.log | 2014-01-27 | 12:32:51 | Boot Start | |
| 32674 | device3_intouch.log | 2014-01-27 | 12:35:19 | Boot Start | |
| 32693 | device3_intouch.log | 2014-01-27 | 14:02:38 | Boot Start | |
| 32801 | device3_intouch.log | 2014-01-27 | 14:05:21 | Boot Completed | 163800ms |
| 33709 | device3_intouch.log | 2014-01-28 | 12:44:17 | Boot Start | |

The text editor window also shows a sidebar with a file list and a bottom status bar indicating "Diff", "Tab Width: 8", "Ln 1, Col 1", and "INS". A web browser window is open in the background, showing a "Grading Process" page with a "Refresh Output" button and a list of files: "airport (1).tar.gz", "airport.tar.gz", and "ps5b (1).tar.gz". The desktop background is a red and purple abstract design. The taskbar at the bottom shows various application icons.


```
1:
2:
3: #include <iostream>
4: #include <string>
5: #include <boost/regex.hpp>
6: #include <fstream>
7: #include <sstream>
8:
9:
10: #include "boost/date_time/gregorian/gregorian.hpp"
11: #include "boost/date_time/posix_time/posix_time.hpp"
12:
13:
14: using namespace std;
15: using namespace boost;
16:
17: using boost::gregorian::date;
18: using boost::gregorian::from_simple_string;
19: using boost::gregorian::date_period;
20: using boost::gregorian::date_duration;
21: using boost::posix_time::ptime;
22: using boost::posix_time::time_duration;
23:
24: int main(int argc, char* argv[]){
25:     int line_num = 1;
26:     string line;
27:     string name(argv[1]);
28:     string Filename;
29:     string date_time = "([0-9]{4})-([0-9]{2})-([0-9]{2}) ([0-9]{2}):([0-9]{2}
}):([0-9]{2})";
30:     string boot_ups = "(.*log.c.166.*)";
31:     string completion = ".([0-9]{3}):INFO:oejs.AbstractConnector:Started Sel
ectChannelConnector@0.0.0.0:9080";
32:     bool success = true;
33:     int boot_success = 0;
34:     ptime time_start, time_end;
35:
36:     if (argc != 2) {
37:         cout << "invalid num of arg";
38:         return 0;
39:     }
40:
41:
42:     name = string(argv[1]);
43:     Filename = name + ".rpt";
44:
45:     regex start(date_time + boot_ups);
46:     regex end(date_time + completion);
47:
48:     smatch sm;
49:
50:     ifstream in(name.c_str());
51:     ofstream out;
52:     out.open(Filename.c_str());
53:
54:     while (getline(in, line)) {
55:
56:         string s("2000-01-01");
57:         date dl(from_simple_string(s));
58:
59:         if (regex_match(line, sm, start)) {
```

```
60:
61:         if (success == false) {
62:             out << "**** Incomplete boot **** " << endl << endl;
63:         }
64:
65:         out << "=== Device boot ===" << endl;
66:         out << line_num << "(" << name << "): " << sm[1] << "-" << sm[2] <
< "-" << sm[3] << " " << sm[4] << ":" << sm[5] << ":" << sm[6] << " Boot Start" <<
endl;
67:
68:         int time1, time2, time3;
69:
70:         stringstream timeA(sm[4]);
71:         timeA >> time1;
72:         stringstream timeB(sm[5]);
73:         timeB >> time2;
74:         stringstream timeC(sm[6]);
75:         timeC >> time3;
76:
77:         time_start = ptime(d1, time_duration(time1,time2,time3));
78:
79:         success = false;
80:     }
81:
82:     if (regex_match(line, sm, end)) {
83:
84:         out << line_num << "(" << name << "): " << sm[1] << "-" << sm[2] <<
"-" << sm[3] << " " << sm[4] << ":" << sm[5] << ":" << sm[6] << " Boot Completed" <
< endl;
85:
86:         int time1, time2, time3;
87:
88:         stringstream timeA(sm[4]);
89:         timeA >> time1;
90:         stringstream timeB(sm[5]);
91:         timeB >> time2;
92:         stringstream timeC(sm[6]);
93:         timeC >> time3;
94:         time_end = ptime(d1, time_duration(time1,time2,time3));
95:
96:         time_duration boot_time = time_end - time_start;
97:
98:         out << "\tBoot Time: " << boot_time.total_milliseconds() << "ms " <<
endl << endl;
99:
100:        success = true;
101:        boot_success++;
102:    }
103:
104:    line_num++;
105:
106:    }
107:    out.close();
108:    return 0;
109: }
```