Thomas DeMasse
Date: 12/10/2021
COMP IV: Project Portfolio
Fall 2021

Contents:

PS0 – Hello World with SFML

Introduction to assignment: This assignment was intended for us to get our environment set up using Ubuntu on a VirtualBox VM. Once set up we were required to install SFML and implement some very basic features in order to ensure everything was working properly as well as familiarize ourselves with our new environment.

DISCUSSION:

1. **Setting Up the Environment**: I was familiar with VirtualBox already as I already had an environment set up on my machine. However, I had this environment set up on a Mac computer instead of a Windows PC. I switched from Mac to PC for the sole purpose of this class. Setting up the environment on a PC was actually easier than on a Mac. I ran into memory issues on my Mac trying to get it set up but when I tried on Windows it worked fine. No issues. I stuck with my Windows OS the rest of the way. I had no issues installing any additional packages I wanted to use, including the Host Additions for Ubuntu. Everything went smoothly.

2. **Building the SFML Hello World Code:** Installing the SFML package was straightforward and easy. I got the Hello World code to display easily. When it came to drawing our own sprite there were four key components we had to implement:

    a. Drawing the sprite

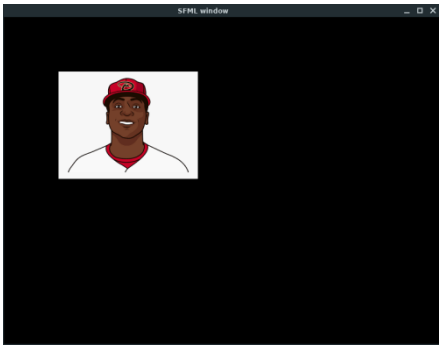
    b. Making the image sprite move

    c. Make the image sprite respond to key strokes

    d. Make it something else:

CONCLUSION: The only issue I ran into was on Part D. I attempted to turn the sprite green when clicking on it. However instead of turning green when clicked on, the sprite turned green if you clicked on anywhere in the window. With what I have learned from this class I am VERY confident I could make it turn green when clicked on now. It's pretty simple.

SCREENSHOT:


SFML window

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS0
 4:     Due Date: 09/13/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION:
 8:
 9:     1. drawing an image sprite
10:     2. make the image sprite move
11:     3. make the image sprite respond to keystrokes
12:     4. Turn the sprite a different color
13:
14:     BUGS: The sprite turns green when the left mouse button is clicked.
15:           It doesn't matter where it's clicked to turn the sprite green.
16:
17: */
18:
19: #include <SFML/Audio.hpp>
20: #include <SFML/Graphics.hpp>
21: #include <iostream>
22:
23: using namespace std;
24:
25: int main()
26: {
27:     // Create the main window
28:     sf::RenderWindow window(sf::VideoMode(800, 600), "SFML window");
29:
30:     // Load a sprite to display
31:     sf::Texture texture;
32:     if (!texture.loadFromFile("sprite.png"))
33:         return EXIT_FAILURE;
34:     sf::Sprite sprite(texture);
35:
36:     //create variables for position of sprite
37:     float xPos = 100;
38:     float yPos = 100;
39:
40:     //set position of sprite on screen
41:     sprite.setPosition(sf::Vector2f(xPos,yPos));
42:
43: // Start the game loop
44:     while (window.isOpen())
45:     {
46:         // Process events
47:         sf::Event event;
48:         while (window.pollEvent(event))
49:         {
50:             // Close window: exit
51:             if (event.type == sf::Event::Closed)
52:                 window.close();
53:
54:             //create event for when escape is pressed
55:             //closes SFML Window
56:             if(event.type == sf::Event::KeyPressed)
57:                 if(event.key.code == sf::Keyboard::Escape)
58:                 {
59:                     window.close();
60:
61:                 }
```

```
    //moves sprite to the right by pressing right key
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right))
    {
        xPos = 25;
        yPos = 0;
        sprite.move(xPos, yPos);

    }

    //Moves sprite to left by pressing the left key
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
    {
        xPos = -25;
        yPos = 0;
        sprite.move(xPos, yPos);

    }
    //Moves sprite down by pressing down key
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down))
    {
        xPos = 0;
        yPos = 25;
        sprite.move(xPos, yPos);

    }
    //Moves sprite up when pressing up key
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up))
    {
        xPos = 0;
        yPos = -25;
        sprite.move(xPos, yPos);

    }

//When mouse is left-clicked the sprite turns green
if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
{

    // left mouse button is pressed: turn sprite green
    sprite.setColor(sf::Color(0, 255, 0));

}



    }
        // Clear screen
        window.clear();
        // Draw the sprite
        window.draw(sprite);
        // Update the window
        window.display();
    }

    return EXIT_SUCCESS;
}
```

PS1(Part A) – Linear Feedback Shift Register

Introduction to assignment: For this assignment we were tasked with creating a Linear Feedback Shift Register. It was a program that produces pseudo-random bits. The goal was to be able to create the register with the idea we would be able to implement and encrypt a simple form of digital pictures for part b of the assignment.

DISCUSSION:

1. The first thing I did was try to understand exactly how a linear feedback shifts register works. Being a transfer student coming into the class we had not learned anything like this in previous classes I took at NSCC which made it a bit of a challenge for me.

2. I took the API we were given and a created the outline of the program. Then I created the necessary methods. First I created the constructor, then I implemented the step() method, and finally I created the generate() method. The step() method simulates one step of the LFSR. The generate() method simulates a desired number of steps for the LFSR. I also created a toString() method to return a string representation of the LFSR to insure it was working properly.

3. I installed the boost library to test my code and implemented a couple of basic unit tests. I thought at the time I had linked it properly but it turns out I had not. I have since figured out how to link the library properly and run tests on my code. Below is a screenshot of the code running. At the time I was unsure if my methods were working properly – I was in the process of debugging when I submitted Part A of the assignment.

4. I completed the makefile to run the program. At the time I was unsure if makefile was working properly. I wrote "I completed the makefile which I believe makes the test.cpp, fibLFSR.h, and FibLFSR.cpp files into one object file called ps1a.o The program runs without any tests failing and I'm not sure if a message should be displayed saying the tests have passed or if it's just good it compiles." I now know a message should have been displayed. It should display "no errors detected."

CONCLUSION: I had a really hard time figuring out how to implement all the tap positions necessary to mimic the Fibonacci LFSR in the assignment. I had a really hard time figuring out the step() function and the generate() function. I was able to make a simple makefile I was able to install the boost library but not implement it. I have a better understanding of how this all works after completing the course Computing IV.

SCREENSHOT:


PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
bash
osboxes ~ Documents PS1a ./FibLFSR
in step process
1
in step process
1
in step process
1
in step process
1
in step process
1
31
osboxes ~ Documents PS1a

PS1(Part B) – Linear Feedback Shift Register - PhotoMagic

Introduction to assignment: Write a C++ program to read three arguments from the command line: source image filename, output image filename, and FibLFSR seed. Then, use SFML to load the source image from disk and display it in its own window. Next, encode the image (or decode). Display the encoded/decoded image in its own window. Save the image to a disk.

DISCUSSION:

1. We were tasked to write a program called PhotoMagic.cpp that can encrypt and decrypt pictures. But before I started on the PhotoMagic program, I made sure Part A was running correctly. After making sure I understood how Part A worked, I moved on to Part B.

2. In part B of the assignment we had to implement a Transform() function which would transform a given image using the LFSR we created in Part A of the assignment. The following code is a snippet of the transform() function that was essential to the assignment.

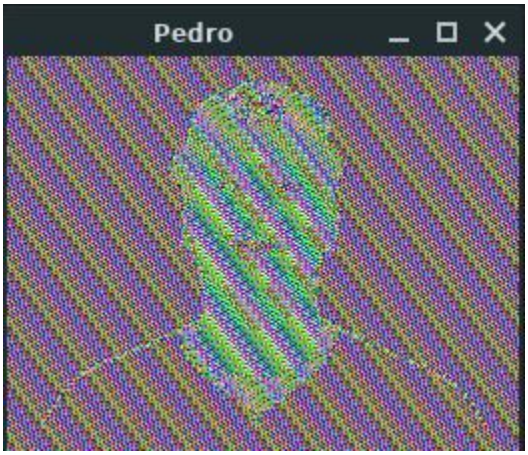
```
// create encrypted/decrypted image
  for (int x = 0; x<width; x++) {
          for (int y = 0; y< height; y++) {
                  p = image.getPixel(x, y);
                  int ebit = fibLFSR->generate(8);
                  p.r = p.r^ebit;
                  ebit = fibLFSR->generate(8);
                  p.g = p.g^ebit;
                  ebit = fibLFSR->generate(8);
                  p.b = p.b^ebit;
                  image.setPixel(x, y, p);
          }
```

3. The final thing we had to do was create a main function that would take the three command-line arguments: a picture filename, and the LFSR seed. At first I was getting a segmentation fault. After much trial and error, I realized I had an issue with where the main() was placed in the program. It was a silly error that took up way more time that it should have to figure out.

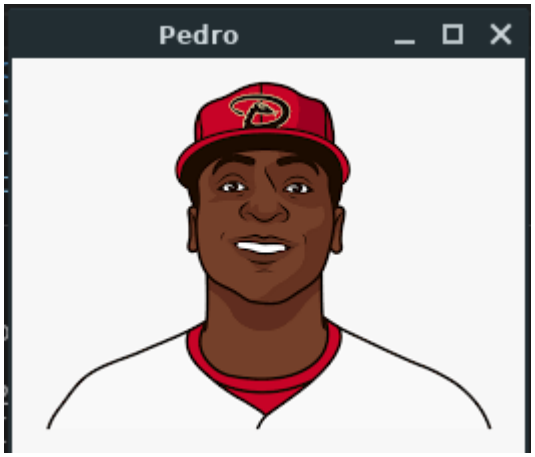
CONCLUSION: The program was a challenge for me at the time. Looking back it is a fairly simple program to implement. This was the first time during the course I was extremely proud of my accomplishments of figuring out this program. Below are screenshots of encrypted and decrypted images.

SCREENSHOTS:

ENCODED PHOTO:



DECODED PHOTO:



```
    1: CC = g++
    2: CFLAGS = -std=c++11 -c -g -Og -Wall -Werror -pedantic
    3: OBJ = PhotoMagic.o
    4: LFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lboost_unit_test_frame
work
    5: EXE = PhotoMagic
    6:
    7: all: $(OBJ)
    8:         $(CC) $(OBJ) -o $(EXE) $(LFLAGS)
    9:
   10: PhotoMagic.o: PhotoMagic.cpp FibLFSR.cpp test.cpp FibLFSR.h $(DEPS)
   11:         $(CC) $(CFLAGS) -o $@ $<
   12:
   13: clean:
   14:         \rm $(OBJ) $(EXE)
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS1a and PS1B
 4:     Due Date: 09/28/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To implement a Fibonacci LFSR
 8:                  To implement photomagic.cpp
 9:
10: */
11:
12:
13: #include <SFML/System.hpp>
14: #include <SFML/Window.hpp>
15: #include <SFML/Graphics.hpp>
16: #include "FibLFSR.h"
17: #include "FibLFSR.cpp"
18: #include <iostream>
19: using namespace std;
20:
21: void showImage(sf::Image& image) {
22:     sf::Vector2u size = image.getSize();
23:     sf::RenderWindow window(sf::VideoMode(size.x, size.y), "Pedro");
24:
25:     sf:: Texture texture;
26:     texture.loadFromImage(image);
27:     sf::Sprite sprite;
28:     sprite.setTexture(texture);
29:     while(window.isOpen())
30:     {
31:         sf::Event event;
32:         while(window.pollEvent(event))
33:         {
34:             if (event.type == sf::Event::Closed)
35:                     window.close();
36:         }
37:         window.clear(sf::Color::White);
38:         window.draw(sprite);
39:         window.display();
40:     }
41: }
42:
43: void transform(sf::Image& image, FibLFSR* fibLFSR) {
44:         sf::Color p; // p is a pixel
45:
46:         int height = image.getSize().x;
47:         int width = image.getSize().y;
48:
49:         // create encrypted/decrypted image
50:         for (int x = 0; x < height; x++) {
51:                 for (int y = 0; y< width; y++) {
52:                         p = image.getPixel(x, y);
53:                         int ebit = fibLFSR->generate(8);
54:                         p.r = p.r^ebit;
55:                         ebit = fibLFSR->generate(8);
56:                         p.g = p.g^ebit;
57:                         ebit = fibLFSR->generate(8);
58:                         p.b = p.b^ebit;
59:                         image.setPixel(x, y, p);
60:                 }
61:         }
```

```
62: }
63:
64: int main(int argc, char* argv[]) {
65:     std::string seed = argv[3];
66:     std::string inputFileName = argv[1];
67:     std::string outputFileName = argv[2];
68:     int tap = 0;
69:
70:     sf::Image image;
71:     if (!image.loadFromFile(inputFileName)) {
72:         return -1;
73:     }
74:     showImage(image);
75:
76:     FibLFSR lfsr(seed, tap);
77:     transform(image, &lfsr);
78:     showImage(image);
79:     if (!image.saveToFile(outputFileName)){
80:         return -1;
81:     }
82:     return 0;
83: }
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS1a and PS1B
 4:     Due Date: 09/27/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To implement a Fibonacci LFSR
 8:                  To implement photomagic.cpp
 9:
10: */
11:
12:
13:
14:
15: #ifndef FibLFSR_h
16: #define FibLFSR_h
17:
18: #include <string>
19: using namespace std;
20:
21: class FibLFSR {
22: private:
23:    int size;   // Length of the LFSR in bits
24:    int* reg;   // pointer to the registry
25:    int tap; // index positionof the tap bit
26:
27: public:
28:    // constructor to create the object
29:    FibLFSR(string seed, int t1);
30:
31:    ~FibLFSR();
32:
33:    //Function to convert toString();
34:    string to_string();
35:
36:    //simulates one step
37:    int step();
38:
39:    // generates the output of k bits
40:    int generate(int k);
41:
42:    int getTap();
43: };
44: #endif
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS1a and PS1B
 4:     Due Date: 09/28/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To implement a Fibonacci LFSR
 8:                  To implement photomagic.cpp
 9:
10: */
11:
12: #include <iostream>
13: #include <string>
14: #include <cmath>
15: #include "FibLFSR.h"
16:
17: using namespace std;
18:
19: // constructor to create LFSR with the given initial seed and tap
20: FibLFSR::FibLFSR(string seed, int t1)
21: {
22:     size = seed.length();
23:     tap = t1;
24:     reg = new int[size];
25:     for (int i = 0; i < size; i++)
26:     {
27:         if (seed[i] == '0')
28:         {
29:             reg[i] = 0;
30:         }
31:         else
32:         {
33:             reg[i] = 1;
34:         }
35:     }
36: }
37:
38: FibLFSR::~FibLFSR()
39: {
40:     delete[] reg;
41: }
42:
43: //Convert to string
44: string FibLFSR::to_string()
45: {
46:     string holder = "";
47:
48:     for (int i = 0; i < size;  i++)
49:     {
50:         if (reg[i] == 0)
51:         {
52:             holder = holder + '0';
53:         }
54:         else
55:         {
56:             holder = holder + '1';
57:         }
58:     }
59:
60:     return holder;
61: }
```

```
   62:
   63: // Executes one step of the LFSR
   64: int FibLFSR::step() {
   65:     int bit, temp, initTap;
   66:
   67:     temp = reg[0];
   68:     initTap = reg[(size-1)-tap];
   69:
   70:     for (int i=0; i<size; i++) {
   71:         if (i == size-1) {
   72:             reg[i] = temp^initTap;
   73:         }
   74:         else {
   75:             reg[i] = reg[i+1];
   76:         }
   77:     }
   78:
   79:     bit = reg[size-1];
   80:
   81:     return bit;
   82: }
   83:
   84: // Generates multiple steps if the LFSR
   85: int FibLFSR::generate(int k)
   86: {
   87:     int number = 0;
   88:     int *tempReg = new int[k];
   89:
   90:     for (int i = 0; i < k; i++)
   91:     {
   92:         tempReg[i] = step();
   93:     }
   94:
   95:     for (int i = 0; i < k; i++)
   96:     {
   97:         tempReg[(k - 1) - i] = tempReg[(k - 1) - i] * (int)pow(2, i);
   98:     }
   99:
  100:     for (int i = 0; i < k; i++)
  101:     {
  102:         number = number + tempReg[i];
  103:     }
  104:
  105:     delete[] tempReg;
  106:
  107:     return number;
  108: }
```

PS2 (Part A) – N-body Simulation

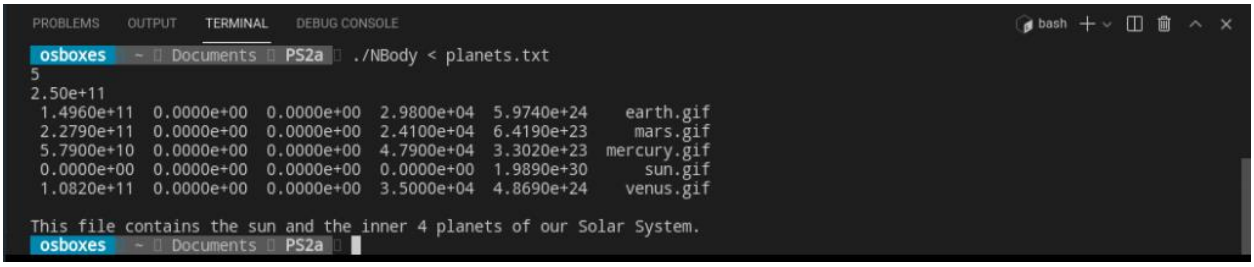
Introduction to assignment: For the first part of the N-Body simulation program our goal was to load and display a static universe using the planets.txt file. We had to create a CelestialBody class and a Universe class containing the celestial bodies. We also had to implement sf::Drawable with a private virtual void method draw(). I actually attempted this program twice. The first time I did not get it working correctly. The second time I scrapped my first attempt and successfully completed the assignment for PSX.

DISCUSSION:

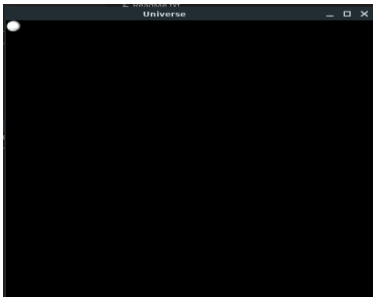
1. I had a really hard time getting the program to work. I was able to read all of the planets.txt information however I could not get the positions of the planets to display properly - they only displayed in the top-hand left corner of the screen. Even being able to read from the planets.txt file was challenging. I used a delimiter. I successfully created the Sf::Drawable method the private virtual void method named draw. This part took a lot of trial and error but eventually I got it. In the end I was frustrated but turned in what I had completed.

CONCLUSION: Because I had a lot of trouble I received a less than satisfactory grade for my efforts. It was an appropriate grade for the work I turned in, however I felt if I had more time I could successfully figure it out and I did for Part B.

SCREENSHOT ONE:



SCREENSHOT TWO:



PS2 (Part B) – N-body Simulation

Introduction to assignment: For the second part of the N-Body Simulation we had to add a physics simulation and animation to the program we created in Part A. The CelestialBody class I created needed to be extended with mutators in order to perform the physics simulation. We needed to implement a method named step() in the Universe class. The main routine needed to keep track of the time in order to properly terminate when the simulation completed.

DISCUSSION: I encountered a lot of problems with the way I attempted to implement the API in Part A of this assignment. I decided to completely scrap Part A and attempted to start the program over with a fresh mind on a new day. With a clear head (and a lot of coffee!) I completed the full simulation of the Universe.

1. My first goal was to get the planets to display.

2. After I figured that out, I extended the CelestialBody class with mutators and added the necessary logic and calculations inside the methods. This includes creation of the step() method.

3. Then I tested each method individually to ensure the calculations were being performed properly.

4. Finally, once I concluded all the calculations were correct, I drew the universe in the main function. I used a vector of CelestialBody objects to create each planet and draw them to the universe.

CONCLUSION: To be honest I found this whole program very challenging to complete in 1 week. I had trouble setting the positions of the planets. Just about everything to do with this assignment was a tremendous challenge for me. I was frustrated because I work a full-time job and to complete this assignment in one week used up all of my spare time. It's frustrating to know that the way I structured my program the first time ended up being the most challenging thing to overcome. I proud of this assignment and what I accomplished and I think being challenged in such a way only allowed me to grow as a programmer.

SCREENSHOT: This is a live shot of the planets rotating

```
 1: CC = g++
 2: CFLAGS = -std=c++11 -c -g -Og -Wall -pedantic
 3: OBJ = universe.o  main.o
 4: LIBS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
 5: EXE = NBody
 6:
 7: all: $(OBJ)
 8:         $(CC) $(OBJ) -o $(EXE) $(LIBS)
 9:
10: Universe.o: Universe.cpp universe.h $(DEPS)
11:         $(CC) $(CFLAGS) -o $@ $<
12:
13: main.o: main.cpp universe.h
14:
15: clean:
16:         \rm $(OBJ) $(EXE)
```

```
    1: #include <iostream>
    2: #include <vector>
    3: #include <SFML/Graphics.hpp>
    4: #include <SFML/Window.hpp>
    5: #include <SFML/Audio.hpp>
    6: #include <new>
    7: #include <cstdlib>
    8: #include "universe.h"
    9:
   10: int main(int argc, char *argv[]) {
   11:
   12:   const double gravity = 6.67 * 1e-11;
   13:   double radius = 0;
   14:   double force = 0;
   15:   double netXForce = 0;
   16:   double netYForce = 0;
   17:   double deltaX = 0;
   18:   double deltaY = 0;
   19:
   20:   int count = 0;
   21:
   22:   double time = 157788000.0;
   23:   double deltaT = 25000.0;
   24:
   25:   int numOfBodies;
   26:   std::cin >> numOfBodies;
   27:
   28:   double radiusOfuniverse;
   29:   std::cin >> radiusOfuniverse;
   30:
   31:   sf::Vector2f window_size(500, 500);
   32:
   33:   std::vector<CelestialBody*> bodies(numOfBodies);
   34:
   35:   for(int i = 0; i < numOfBodies; i++) {
   36:     bodies[i] = new CelestialBody(radiusOfuniverse, window_size);
   37:     std::cin >> *bodies[i];
   38:   }
   39:
   40:   for(int i = 0; i < numOfBodies; i++) {
   41:     bodies[i]->loadImage();
   42:     bodies[i]->setTexture();
   43:   }
   44:
   45:   sf::Texture starfield;
   46:   sf::Sprite background;
   47:
   48:   starfield.loadFromFile("starfield.jpg");
   49:   background.setTexture(starfield);
   50:
   51:   sf::RenderWindow window(sf::VideoMode(window_size.x, window_size.y), "Univ
erse");
   52:
   53:
   54:   int framesPerSec = 1 / (deltaT * 1e-6);
   55:   window.setFramerateLimit(framesPerSec);
   56:
   57:   int total = (time * 1e-6) * framesPerSec;
   58:
   59:   while(window.isOpen()) {
   60:     sf::Event event;
```

```
   61:     while(window.pollEvent(event)) {
   62:       if (event.type == sf::Event::Closed) window.close();
   63:     }
   64:     if(count < total) {
   65:       for(int i = 0; i < numOfBodies; i++) {
   66:         for(int j = 0; j < numOfBodies; j++) {
   67:           if(i != j) {
   68:             deltaX = Distance(bodies[i]->_xPos, bodies[j]->_xPos);
   69:             deltaY = Distance(bodies[i]->_yPos, bodies[j]->_yPos);
   70:             radius = calculateRadius(deltaX, deltaY);
   71:             force = calculateForce(bodies[i]->_mass, bodies[j]->_mass, gravi
ty, radius);
   72:             netXForce += calculateForceXY(deltaX, radius, force);
   73:             netYForce += calculateForceXY(deltaY, radius, force);
   74:           }
   75:         }
   76:         bodies[i]->_xForce = netXForce;
   77:         bodies[i]->_yForce = netYForce;
   78:         netXForce = 0;
   79:         netYForce = 0;
   80:       }
   81:
   82:       for(int i = 0; i < numOfBodies; i++) {
   83:         bodies[i]->step(deltaT);
   84:         bodies[i]->universeToWindow();
   85:       }
   86:
   87:       window.clear();
   88:       window.draw(background);
   89:       for(int i = 0; i < numOfBodies; i++) window.draw(*bodies[i]);
   90:       window.display();
   91:
   92:       count++;
   93:     }
   94:   }
   95:
   96:   return 0;
   97: }
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS6
    4:     Due Date: 11/29/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To create Nbody simulation
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14: #include <iostream>
   15: #include <string>
   16: #include <SFML/Graphics.hpp>
   17:
   18: class CelestialBody : public sf::Drawable {
   19:   public:
   20:     CelestialBody(double radius, const sf::Vector2f& size_of_window);
   21:     ~CelestialBody();
   22:     friend std::istream& operator>>(std::istream& in, CelestialBody& B);
   23:     void loadImage();
   24:     void setTexture();
   25:     void universeToWindow();
   26:     void step(double seconds);
   27:
   28:     double _xPos;
   29:     double _yPos;
   30:     double _xVel;
   31:     double _yVel;
   32:     double _xForce;
   33:     double _yForce;
   34:     double _mass;
   35:
   36:     const double radiusOfUniverse;
   37:     const sf::Vector2f sizeOfWindow;
   38:
   39:     std::string _image;
   40:     sf::Texture _texture;
   41:     sf::Sprite _sprite;
   42:
   43:   private:
   44:     void calculateVelocity(double xAccel, double yAccel, double deltaT);
   45:     void calculatePosition(double deltaT);
   46:     void virtual draw(sf::RenderTarget& target, sf::RenderStates states) con
st;
   47: };
   48:
   49: double Distance(double Pos1, double Pos2);
   50: double calculateRadius(double deltaX, double deltaY);
   51: double calculateForce(double mass1, double mass2, double Gravity, double rad
ius);
   52: double calculateForceXY(double delta, double radius, double force);
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS6
 4:     Due Date: 11/29/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To create Nbody simulation
 8:
 9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
10:
11:
12: */
13:
14: #include <cmath>
15: #include <iostream>
16: #include "universe.h"
17: #include <SFML/Graphics.hpp>
18:
19: CelestialBody::CelestialBody(double radius, const sf::Vector2f& windowSize)
20: : radiusOfUniverse(radius), sizeOfWindow(windowSize) {}
21:
22: CelestialBody::~CelestialBody() {}
23:
24: void CelestialBody::draw(sf::RenderTarget& target,
25: sf::RenderStates states) const {
26:     target.draw(_sprite);
27: }
28:
29: void CelestialBody::step(double deltaT) {
30:     double xAcceleration = _xForce / _mass;
31:     double yAcceleration = _yForce / _mass;
32:
33:   calculateVelocity(xAcceleration, yAcceleration, deltaT);
34:
35:   calculatePosition(deltaT);
36: }
37:
38: void CelestialBody::calculateVelocity(double xAcceleration,
39: double yAcceleration, double deltaT) {
40:     _xVel += (xAcceleration * deltaT);
41:     _yVel += (yAcceleration * deltaT);
42: }
43:
44: void CelestialBody::calculatePosition(double deltaT) {
45:     _xPos += (_xVel * deltaT);
46:     _yPos += (_yVel * deltaT);
47: }
48:
49: void CelestialBody::universeToWindow() {
50:     double xMod = ((_xPos/radiusOfUniverse) * (sizeOfWindow.x / 2))
51:     + sizeOfWindow.x / 2;
52:     double yMod = (-(_yPos/radiusOfUniverse) * (sizeOfWindow.y / 2))
53:     + sizeOfWindow.y / 2;
54:
55:   sf::FloatRect sprite_dimensions = _sprite.getGlobalBounds();
56:   xMod -= (sprite_dimensions.width / 2);
57:   yMod -= (sprite_dimensions.height / 2);
58:   _sprite.setPosition(xMod, yMod);
59: }
60:
61: void CelestialBody::loadImage() {
```

```
   62:     if (!(_texture.loadFromFile(_image))) std::cerr << "Error! Could not loa
d: "
   63:     << _image << " ." << std::endl;
   64: }
   65:
   66: void CelestialBody::setTexture() {
   67:     _sprite.setTexture(_texture);
   68: }
   69:
   70: std::istream& operator>>(std::istream& in, CelestialBody& B) {
   71:     in >> B._xPos >> B._yPos >> B._xVel >> B._yVel >> B._mass >> B._image;
   72:
   73:     return in;
   74: }
   75:
   76: double Distance(double Pos1, double Pos2) {
   77:     return Pos2 - Pos1;
   78: }
   79:
   80: double calculateRadius(double deltaX, double deltaY) {
   81:     return sqrt(pow(deltaX, 2) + pow(deltaY, 2));
   82: }
   83:
   84: double calculateForce(double mass1, double mass2, double Gravity,
   85:                       double radius) {
   86:                           return (Gravity * mass1 * mass2) / pow(radius, 2);
   87: }
   88:
   89: double calculateForceXY(double delta, double radius, double force) {
   90:     return force * (delta / radius);
   91: }
   92:
```

PS3 – Recursive Graphics

Introduction to assignment: The idea behind this assignment was to write a program that plots a Triangle Fractal. It is a variation of the Sierpinski Triangle. We were given specific API once again to implement. We had to write a TFractal.cpp program with a recursive function fTree() and a main() that calls the recursive function. The program required us to take two command-line arguments L (the length of the side of the base equilateral triangle) and N (the depth of the recursion).

DISCUSSION:

1. First my goal was to just draw one triangle to start and then implement the recursive part. Once I drew the first triangle then I added to the Triangle class constructor. I used an initialization list and performed some logic in the constructor for the creation of each triangle. I also created a destructor to delete triangles.

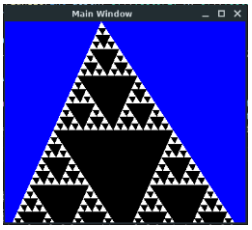
2. I incorporated the sf::Drawable to draw the triangle to the main window. We have used this before so I was familiar with how to implement. It was not as challenging for me as before. It uses sf::ConvexShape to draw the triangle. I also created fTree() and a mid() methods.

3. In my main function (Tfractal.cpp) I set the size of the window to adjust with the size of the triangle. It creates a Triangle object (Triangle(recursionDepth, A, B, C)); When it's working properly it takes in as parameters the size of the base of the triangle, and recursion depth; then it draws the SFML window containing the Sierpinksi Triangle to the screen.

4. This was the first time we were introduced to cpplint. It was pretty straightforward to get working. We were given all of the commands to get it working in the assignment. Once I had it set up I found it very tedious to keep going through my code a fixing mistakes.

CONCLUSION: I thought the assignment was less challenging than the more recent assignments. I was able to successfully draw the Sierpienski Triangle. All in all it was a tough assignment but not as tough as the previous couple. I got better at recursion. I like cpplint but there are better options to test your code I found out. sonarLint for example is a better option. There is a plugin for it to test your code as you go.

SCREENSHOT:



```
 1: CC = g++
 2: CFLAGS = -std=c++11 -c -g -Og -Wall -pedantic
 3: OBJ = Tfractal.o
 4: LIBS = -lsfml-graphics -lsfml-window -lsfml-system
 5: EXE = Triangle
 6:
 7: all: $(OBJ)
 8:         $(CC) $(OBJ) -o $(EXE) $(LIBS)
 9:
10: Tfractal.o: Tfractal.cpp Triangle.cpp Triangle.hpp $(DEPS)
11:         $(CC) $(CFLAGS) -o $@ $<
12:
13: clean:
14:         \rm $(OBJ) $(EXE)
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS3
    4:     Due Date: 10/18/21
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To draw the Sierpinski Triangle
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14:
   15: #include "Triangle.hpp"
   16: #include "Triangle.cpp"
   17: #include <iostream>
   18: #include <cstdlib>
   19: #include <math.h>
   20: #include <SFML/Graphics.hpp>
   21:
   22: int main(int argc, char* argv[]) {
   23:     int recursionDepth = std::atoi(argv[1]);
   24:     int size = std::atoi(argv[2]);
   25:     double windowHeight = (sqrt(3) / 2) * size;
   26:     sf::RenderWindow window(sf::VideoMode(size, windowHeight), "Main Window"
);
   27:     sf::Vector2f A(0, windowHeight);
   28:     sf::Vector2f B(size / 2, 0);
   29:     sf::Vector2f C(size, windowHeight);
   30:     while (window.isOpen()) {
   31:         sf::Event event;
   32:         Triangle triangle(recursionDepth, A, B, C);
   33:         while (window.pollEvent(event)) {
   34:             if (event.type == sf::Event::Closed)
   35:                 window.close();
   36:         }
   37:     window.clear(sf::Color::Blue);
   38:     window.draw(triangle);
   39:
   40:     window.display();
   41: }
   42: return 0;
   43: }
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS3
 4:     Due Date: 10/18/21
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To draw the Sierpinski Triangle
 8:
 9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
10:
11: */
12:
13:
14: #ifndef Triangle_hpp
15: #define Triangle_hpp
16: #include <iostream>
17: #include <SFML/Graphics.hpp>
18: class Triangle : public sf::Drawable {
19:  public:
20:     Triangle(int, sf::Vector2f, sf::Vector2f, sf::Vector2f);
21:
22:     ~Triangle();
23:
24:     sf::Vector2f mid(sf::Vector2f, sf::Vector2f);
25:     void fTree(int, sf::Vector2f, sf::Vector2f, sf::Vector2f);
26:
27:  private:
28:     void draw(sf::RenderTarget &target, sf::RenderStates states) const;
29:     int depth_of_recursion;
30:     sf::Vector2f triA;
31:     sf::Vector2f triB;
32:     sf::Vector2f triC;
33:
34:     sf::Vector2f subA;
35:     sf::Vector2f subB;
36:     sf::Vector2f subC;
37:     Triangle *top;
38:     Triangle *left;
39:     Triangle *right;
40: };
41: #endif  // _HOME_OSBOXES_DOCUMENTS_PS3_TRIANGLE_HPP_
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS3
    4:     Due Date: 10/18/21
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To draw the Sierpinski Triangle
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14:
   15: #include "Triangle.hpp"
   16: #include <math.h>
   17: #include <iostream>
   18: #include <SFML/Graphics.hpp>
   19:
   20:
   21: Triangle::Triangle(int depth, sf::Vector2f A, sf::Vector2f B, sf::Vector2f C
)
   22: : depth_of_recursion(depth), triA(A), triB(B), triC(C) {
   23:     int shallowDepth = depth_of_recursion - 1;
   24:     fTree(depth_of_recursion, triA, triB, triC);
   25:
   26: if (shallowDepth > 0) {
   27:     top = new Triangle(shallowDepth, mid(triA, triB), triB, mid(triB, triC))
;
   28:     left = new Triangle(shallowDepth, triA, mid(triA, triB), mid(triC, triA)
);
   29:     right = new Triangle(shallowDepth, mid(triC, A), mid(triB, triC), triC);
   30: } else {
   31:     top = NULL;
   32:     left = NULL;
   33:     right = NULL;
   34: }
   35: }
   36: Triangle::~Triangle() {
   37:     if (top != NULL) {
   38:         delete (top);
   39:         delete (left);
   40:         delete (right);
   41:     }
   42: }
   43:
   44: sf::Vector2f Triangle::mid(sf::Vector2f point1, sf::Vector2f point2) {
   45: sf::Vector2f midTriangle((point1.x + point2.x) / 2, (point1.y + point2.y) /
2);
   46:     return midTriangle;
   47: }
   48: void Triangle::fTree(int d, sf::Vector2f A, sf::Vector2f B, sf::Vector2f C)
{
   49:     subA = mid(A, B);
   50:     subB = mid(B, C);
   51:     subC = mid(C, A);
   52: }
   53: void Triangle::draw(sf::RenderTarget &target, sf::RenderStates states) const
 {
   54:     sf::ConvexShape baseTri;
   55:     baseTri.setFillColor(sf::Color::White);
```

```
    baseTri.setPointCount(3);
    baseTri.setPoint(0, triA);
    baseTri.setPoint(1, triB);
    baseTri.setPoint(2, triC);
    sf::ConvexShape subTri;
    subTri.setFillColor(sf::Color::Black);
    subTri.setPointCount(3);
    subTri.setPoint(0, subA);
    subTri.setPoint(1, subB);
    subTri.setPoint(2, subC);
    target.draw(baseTri, states);
    target.draw(subTri, states);

if (top != NULL) {
    top->draw(target, states);
    left->draw(target, states);
    right->draw(target, states);
}}
```

PS4a: Synthesizing a Plucked String Sound

Introduction to assignment: The goal of this assignment was to simulate the plucking of a guitar string using the Karplus-Strong algorithm. We needed to implement a class called CircularBuffer with a specific API.

DISCUSSION:

1. The first thing I did is create the CircularBuffer.h file with the necessary API. Then I began implementing it in the Circularbuffer.cpp class. I created the constructor with an exception that if the capacity was less than one an error message would display.

2. Next I worked on each method individually. There were six methods/functions total:

    a. int size()- which returns the number of items currently in the buffer.

    b. bool isEmpty() – determines if the buffer is empty

    c. bool isFull() – determines if the buffer is full

    d. void enqueue()int16_t x) – adds item x to the end

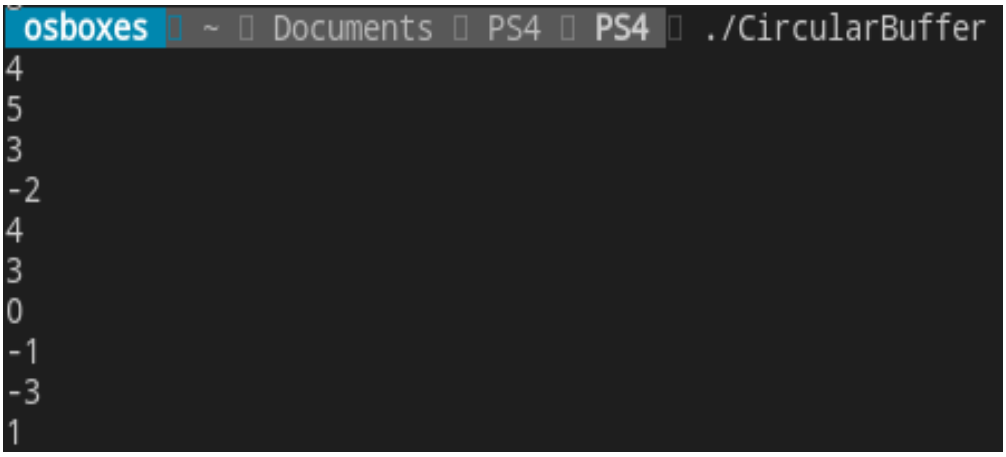
    e. int16_t dequeue() – deletes and returns item from the front

    f. int16_t peek() – return(but not delete) item from the front

3. We were also tasked with testing our code with the boost library. This is something I struggled with in the past and continued to struggle with in this assignment. Realistically when it came time to submit the assignment I ran out of time for boost. I focused my attention on getting the sound to play because the second half of the assignment would be dependent on the first. The program would not compile correctly when linking the test.cpp and the boost libraries in my makefile. I omitted it for the sole purpose of my program to compile.

CONCLUSION: In the end I got the circular buffer to work properly and I outputted the results of the buffer to the screen.

SCREENSHOT:



PS4b: StringSound implementation and SFML audio output

Introduction to assignment: The goal for Part B of the assignment was to implement the Karplus-Strong guitar string simulation, and generate a stream of string samples for audio playback under keyboard control.

DISCUSSION: We had to write a class called StringSound that had a specific API. It had to perform the Karplus-Strong string simulation from Part A. I did not get this whole program working properly. I ran into an error where it would segfault. I think it was a result of the number generator not working properly. I failed to get the sound of the keyboard to play properly because of this. I was unable to solve the problem in time so implementation is not 100% correct.

CONCLUSION: It seems as if I got most of the assignment done correctly aside from using the correct random number generator. Sometimes the program would run and I could play sound and sometimes it wouldn't run at all. It was totally random if it would work every time I ran the executable file.

SCREENSHOT:

There are no screenshots for this assignment because all the program did was output a blank SFML window and then you would use the keys to play sound.

```
 1: CC = g++
 2: CFLAGS = -std=c++11 -c -g -Og -Wall -pedantic
 3: OBJ = KSGuitarSim-Full-SegFault.o StringSound.o CircularBuffer.o
 4: EXE = KSGuitarSim
 5: LIBS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
 6:
 7: all: $(OBJ)
 8:         $(CC) $(OBJ) -o $(EXE) $(LIBS)
 9:
10: KSGuitarSim-Full-SegFault.o: KSGuitarSim-Full-SegFault.cpp
11:
12: CircularBuffer.o: CircularBuffer.cpp  CircularBuffer.hpp
13:         $(CC) $(CFLAGS) -o $@ $<
14:
15: StringSound.o: StringSound.cpp  StringSound.hpp
16:         $(CC) $(CFLAGS) -o $@ $<
17:
18: clean:
19:         \rm $(OBJ) $(EXE)
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS4
    4:     Due Date: 11/1/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To create a musical keyboard using the Karplus-Strong algor
ithm
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14:
   15: #include <math.h>
   16: #include <limits.h>
   17:
   18: #include <iostream>
   19: #include <string>
   20: #include <exception>
   21: #include <stdexcept>
   22: #include <vector>
   23:
   24: #include <SFML/Graphics.hpp>
   25: #include <SFML/System.hpp>
   26: #include <SFML/Audio.hpp>
   27: #include <SFML/Window.hpp>
   28:
   29: #include "CircularBuffer.hpp"
   30: #include "StringSound.hpp"
   31:
   32: #define CONCERT_A 220.0
   33: #define SAMPLES_PER_SEC 44100
   34:
   35: std::vector<sf::Int16> makeSamplesFromString(StringSound sound) {
   36:   std::vector<sf::Int16> samples;
   37:
   38:   sound.pluck();
   39:   int duration = 8;
   40:   int i;
   41:   for (i= 0; i < SAMPLES_PER_SEC * duration; i++) {
   42:     sound.tic();
   43:     samples.push_back(sound.sample());
   44:   }
   45:
   46:   return samples;
   47: }
   48:
   49: int main() {
   50:   sf::RenderWindow window(sf::VideoMode(300, 200), "KSGuitar Simulator");
   51:   sf::Event event;
   52:   double freq;
   53:   int size = 37;
   54:   std::vector<sf::SoundBuffer> buffers(size);
   55:   std::vector<sf::Sound> sounds(size);
   56:   std::vector<std::vector<sf::Int16> > samples(size);
   57:
   58:   for (int i = 0; i < size; i++) {
   59:     freq = 100 + (i * 20);
   60:     StringSound gString = StringSound(freq);
```

```
61:     samples[i] = makeSamplesFromString(gString);
62:     buffers[i].loadFromSamples(&samples[i][0], samples[i].size(), 2,
63:     SAMPLES_PER_SEC);
64:     sounds[i].setBuffer(buffers[i]);
65:   }
66:   int index;
67:   std::string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
68:   while (window.isOpen()) {
69:     while (window.pollEvent(event)) {
70:       switch (event.type) {
71:         case sf::Event::Closed:
72:           window.close();
73:           break;
74:
75:         case sf::Event::TextEntered:
76:           index = keyboard.find(event.text.unicode);
77:           if (index != std::string::npos)
78:             sounds[index].play();
79:           break;
80:
81:         default:
82:           break;
83:       }
84:       window.clear();
85:       window.display();
86:     }
87:   }
88:   return 0;
89: }
```

```
    1: /*
    2:   Copyright 2015 Fred Martin,
    3:   Y. Rykalova, 2020
    4: */
    5:
    6: #include <SFML/Graphics.hpp>
    7: #include <SFML/System.hpp>
    8: #include <SFML/Audio.hpp>
    9: #include <SFML/Window.hpp>
   10:
   11: #include <math.h>
   12: #include <limits.h>
   13:
   14: #include <iostream>
   15: #include <string>
   16: #include <exception>
   17: #include <stdexcept>
   18: #include <vector>
   19:
   20: #include "CircularBuffer.hpp"
   21: #include "StringSound.hpp"
   22:
   23: #define CONCERT_A 220.0
   24: #define SAMPLES_PER_SEC 44100
   25:
   26: std::vector<sf::Int16> makeSamples(StringSound gs) {
   27:   std::vector<sf::Int16> samples;
   28:
   29:   gs.pluck();
   30:   int duration = 8;  // seconds
   31:   int i;
   32:   for (i= 0; i < SAMPLES_PER_SEC * duration; i++) {
   33:     gs.tic();
   34:     samples.push_back(gs.sample());
   35:   }
   36:
   37:   return samples;
   38: }
   39:
   40: int main() {
   41:   sf::RenderWindow window(sf::VideoMode(300, 200), "SFML Plucked String Soun
d Lite");
   42:   sf::Event event;
   43:   double freq;
   44:   int size = 37;
   45:   std::vector<std::vector<sf::Int16> > samples(size);
   46:   std::vector<sf::Sound> sounds(size);
   47:   std::vector<sf::SoundBuffer> buffers(size);
   48:
   49:
   50:
   51:   for(int i = 0; i < size; i++) {
   52:     freq = 100 + (i * 20);
   53:     StringSound gString = StringSound(freq);
   54:     samples[i] = makeSamples(gString);
   55:     buffers[i].loadFromSamples(&samples[i][0], samples[i].size(), 2, SAMPLES
_PER_SEC);//NOLINT
   56:     sounds[i].setBuffer(buffers[i]);
   57:   }
   58:
   59:  int index;
```

```
60:  std::string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
61:   while (window.isOpen()) {
62:     while (window.pollEvent(event)) {
63:       switch (event.type) {
64:       case sf::Event::Closed:
65:         window.close();
66:         break;
67:
68:         case sf::Event::TextEntered:
69:           index = keyboard.find(event.text.unicode);
70:           if (index != std::string::npos)
71:             sounds[index].play();
72:           break;
73:
74:         default:
75:           break;
76:       }
77:       window.clear();
78:       window.display();
79:     }
80:   }
81:   return 0;
82: }
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS4
    4:     Due Date: 11/1/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To create a musical keyboard using the Karplus-Strong algor
ithm
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14: #ifndef _HOME_OSBOXES_DOCUMENTS_PS4_CIRCULARBUFFER_HPP_
   15: #define _HOME_OSBOXES_DOCUMENTS_PS4_CIRCULARBUFFER_HPP_
   16:
   17: #include <stdint.h>
   18:
   19: class CircularBuffer {
   20:  private:
   21:     int _capacity;
   22:     int16_t *buffer;
   23:     int16_t head_i;
   24:     int16_t tail_i;
   25:     int _size;
   26:
   27:  public:
   28:     explicit CircularBuffer(int capacity);
   29:     int size();
   30:     bool isEmpty();
   31:     bool isFull();
   32:     int16_t peek();
   33:     void enqueue(int16_t x);
   34:     int16_t dequeue();
   35: };
   36: #endif  // _HOME_OSBOXES_DOCUMENTS_PS4_CIRCULARBUFFER_HPP_
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS4
    4:     Due Date: 11/1/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To create a musical keyboard using the Karplus-Strong algor
ithm
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14: #include "CircularBuffer.hpp"
   15: #include <exception>
   16:
   17: CircularBuffer::CircularBuffer(int capacity) {
   18:     if (capacity < 1) {
   19:         throw "CircularBuffer constructor: capacity must be greater than zer
o.";
   20:     }
   21:     _size = 0;
   22:     head_i = 0;
   23:     _capacity = capacity;
   24:     buffer = new int16_t[capacity];
   25: }
   26:
   27: void CircularBuffer::enqueue(int16_t val) {
   28:     if (isFull()) {
   29:         throw "Can't enque to a full ring.";
   30:     }
   31:     buffer[(head_i + _size) % _capacity] = val;
   32:     _size++;
   33: }
   34:
   35: int16_t CircularBuffer::dequeue() {
   36:      if (isEmpty()) {
   37:          throw "Can't dequeue from an empty ring.";
   38:      }
   39:     _size--;
   40:     int val = buffer[head_i];
   41:     head_i = (head_i + 1) % _capacity;
   42:     return buffer[val];
   43: }
   44:
   45: int16_t CircularBuffer::peek() {
   46:     int16_t val = buffer[head_i];
   47:     return val;
   48: }
   49:
   50: int CircularBuffer::size() {
   51:     return _size;
   52: }
   53:
   54: bool CircularBuffer::isEmpty() {
   55:     if (_size == 0) {
   56:         return true;
   57:     } else {
   58:         return false;
   59:     }
```

```
60: }
61:
62: bool CircularBuffer::isFull() {
63:     if (_size == _capacity) {
64:         return true;
65:     } else {
66:         return false;
67:     }
68: }
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS4
    4:     Due Date: 11/1/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To create a musical keyboard using the Karplus-Strong algor
ithm
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14: #ifndef _HOME_OSBOXES_DOCUMENTS_PS4_STRINGSOUND_HPP_
   15: #define _HOME_OSBOXES_DOCUMENTS_PS4_STRINGSOUND_HPP_
   16:
   17: #include <vector>
   18: #include "CircularBuffer.hpp"
   19:
   20: class StringSound{
   21:  public:
   22:     explicit StringSound(double frequency);
   23:     explicit StringSound(std::vector<sf::Int16> init);
   24:     ~StringSound();
   25:     void pluck();
   26:     void tic();
   27:     sf::Int16 sample();
   28:     int time();
   29:  private:
   30:     CircularBuffer* _cb;
   31:     int tics;
   32: };
   33: #endif  // _HOME_OSBOXES_DOCUMENTS_PS4_STRINGSOUND_HPP_
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS4
    4:     Due Date: 11/1/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To create a musical keyboard using the Karplus-Strong algor
ithm
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14: #include <math.h>
   15: #include <vector>
   16: #include <stdexcept>
   17: #include <exception>
   18: #include <random>
   19: #include <iostream>
   20:
   21: #include <SFML/Graphics.hpp>
   22: #include <SFML/System.hpp>
   23: #include <SFML/Audio.hpp>
   24: #include <SFML/Window.hpp>
   25:
   26: #include "StringSound.hpp"
   27: #include "CircularBuffer.hpp"
   28:
   29: StringSound::StringSound(double frequency) {
   30:     tics = 0;
   31:     if (frequency <=0) {
   32:         throw std::invalid_argument("Error! Capacity!");
   33:     } else {
   34:         int size = ceil(44100 / frequency);
   35:         _cb = new CircularBuffer(size);
   36:         while (!_cb->isFull()) {
   37:          _cb->enqueue(0);
   38:         }
   39:     }
   40: }
   41:
   42: StringSound::StringSound(std::vector<sf::Int16> init) {
   43:      tics = 0;
   44:   if (init.size() <= 0) {
   45:     throw std::invalid_argument("Capacity must be > 0");
   46:   } else {
   47:     _cb = new CircularBuffer(init.size());
   48:     int i = 0;
   49:     while (!_cb->isFull()) {
   50:         _cb->enqueue(init[i]);
   51:         i++;
   52:     }
   53:   }
   54: }
   55:
   56: StringSound::~StringSound() {
   57:      delete _cb;
   58: }
   59:
   60:
```

```
61: void StringSound::pluck() {
62:     std::random_device rd;
63:     std::mt19937 mt(rd());
64:     std::uniform_real_distribution<double> dist(10000);
65:     if (_cb->isFull()) {
66:         for (int j = 0; j < _cb->size(); j++) {
67:             _cb->dequeue();
68:         }
69:     }
70:     while (!_cb->isFull()) {
71:         _cb->enqueue((int16_t)(dist(mt)));
72:     }
73: }
74:
75:
76: void StringSound::tic() {
77:   sf::Int16 t = 0.5 * 0.996 * (_cb->dequeue() + _cb->peek());
78:   _cb->enqueue(t);
79:   tics++;
80: }
81:
82: sf::Int16 StringSound::sample() {
83:     return _cb->peek();
84: }
85:
86: int StringSound::time() {
87:     return tics;
88: }
```

PS5: Edit Distance

Introduction to assignment: The goals of this assignment were to solve a fundamental problem in computational biology and to learn about dynamic programming. We were allowed to work with a partner but I did not. We had to install Valgrind which is a memory analysis tool.

DISCUSSION: In this program we were introduced to solving a matrix in programming. We were introduced to a functionality of dynamic programming called a map.

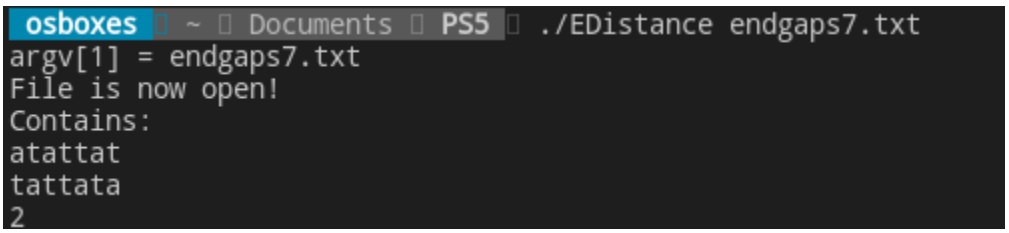
1. First I created the constructor. Next I created the penalty() method which compared two characters. If the characters were the same there was no penalty. If they weren't then there was a penalty of 1. Then I created the min() method which would return the minimum of three arguments.

2. Second I created the most important part of the assignment which was the OptDistance() method. This was the hardest part of the assignment because I really had to understand how to solve the matrix and implement the solution into code. There was a lot of logic involved.

3. The last thing I did was create the main. It reads in the file, compares the strings in the file, and then outputs the Edit Distance to the screen.

CONCLUSION: In the end I chose the dynamic programming approach. I chose it because it seemed to be the most efficient way to find the edit distance. Basically, for me, it was the simplest approach. The program runs with any of the sample *.txt files we were given. Below is a screenshot of the program calculating the Edit Distance of one of the files.

SCREENSHOT:



```
 1: CC = g++
 2: CFLAGS = -std=c++11 -c -g -Og -Wall -pedantic
 3: OBJ = EDistance.o
 4: EXE = EDistance
 5:
 6: all: $(OBJ)
 7:         $(CC) $(OBJ) -o $(EXE)
 8:
 9: EDistance.o: EDistance.cpp EDistance.hpp
10:
11: clean:
12:         \rm $(OBJ) $(EXE)
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS4
    4:     Due Date: 11/8/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To calculate the edit distance of two strings in a file.
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14:
   15:
   16: #include <fstream> // for file-access
   17: #include <string>
   18: #include <iostream>
   19: #include <vector>
   20: #include "EDistance.hpp"
   21:
   22: EDistance::EDistance(std::string s1, std::string s2){
   23:     _str1 = s1;
   24:     _str2 = s2;
   25: }
   26:
   27: int EDistance::penalty(char a, char b){
   28:
   29:     if(a == b){
   30:         return 0;
   31:     }
   32:     return 1;
   33: }
   34:
   35: int EDistance::min(int x, int y, int z) {
   36:     int _min = 0;
   37:     if(x < y) {
   38:         _min = x;
   39:     } else {
   40:         _min = y;
   41:     }
   42:     if(z < _min) {
   43:         _min = z;
   44:     }
   45:     return _min;
   46: }
   47:
   48: int EDistance::OptDistance() {
   49:     std::vector<std::vector<int>> matrix(_str1.size() + 1, std::vector<int>
(_str2.size() + 1));
   50:     for(int i = 0; i < matrix.size(); ++i){
   51:         matrix[i][0] = i;
   52:     }
   53:
   54:     for(int i = 0; i < matrix[0].size(); ++i){
   55:         matrix[0][i] = i;
   56:     }
   57:
   58:     for(int x = 1; x < matrix.size(); ++x){
   59:         for(int y = 1; y < matrix[0].size(); ++y){
   60:             if(_str1[y - 1] == _str2[x - 1]){
```

```
   61:                 matrix[x][y] = matrix[x - 1][y - 1];
   62:             } else {
   63:                 matrix[x][y] = min(matrix[x - 1][y - 1], matrix[x - 1][y], m
atrix[x][y - 1]) + 1;
   64:             }
   65:         }
   66:     }
   67:     return matrix[matrix.size() - 1][matrix[0].size() - 1];
   68: }
   69:
   70:
   71: int main(int argc, char* argv[])
   72: {
   73:
   74:     if (argc > 1) {
   75:         std::cout << "argv[1] = " << argv[1] << std::endl;
   76:     } else {
   77:         std::cout << "No file name entered. Exiting...";
   78:         return -1;
   79:     }
   80:     std::ifstream infile(argv[1]); //open the file
   81:
   82:     if (infile.is_open() && infile.good()) {
   83:         std::cout << "File is now open!\nContains:\n";
   84:         std::string line = "";
   85:         std::string string1 = "";
   86:         std::string string2 = "";
   87:         while (getline(infile, line)){
   88:             string1 = line;
   89:             while(getline(infile, line)) {
   90:                 string2 = line;
   91:             }
   92:             std::cout << string1 << std::endl;
   93:             std::cout << string2 << std::endl;
   94:         }
   95:         EDistance edistance(string1, string2);
   96:         edistance.OptDistance();
   97:         std::cout << edistance.OptDistance() << std::endl;
   98:
   99:     } else {
  100:         std::cout << "Failed to open file..";
  101:     }
  102:     return 0;
  103: }
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS4
 4:     Due Date: 11/8/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To calculate the edit distance of two strings in a file.
 8:
 9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
10:
11:
12: */
13:
14:
15: #ifndef _HOME_OSBOXES_DOCUMENTS_PS5_EDISTANCE_HPP_
16: #define _HOME_OSBOXES_DOCUMENTS_PS5_EDISTANCE_HPP_
17:
18: #include <iostream>
19: #include <fstream>
20: #include <string>
21: #include <cmath>
22: #include <vector>
23:
24: class EDistance{
25:
26: public:
27:     EDistance(std::string str1, std::string str2);
28:     static int penalty(char a, char b);
29:     static int min(int a, int b, int c);
30:     int OptDistance();
31:     std::string Alignment();
32:
33: private:
34:     std::string _str1;
35:     std::string _str2;
36:
37:
38: };
39: #endif
40:
41:
42:
43:
44:
```

PS6: Random Writer

Introduction to assignment: For this assignment we had to analyze input text for transitions between k-grams(a fixes number of characters) and the following letter. Then produce a probabilistic model of the text. Then use the model to generate nonsense text that's surprisingly reasonable.

DISCUSSION: Like all of the other programs there was a specific API we had to implement. For this program the API was:

1. RandWriter(string text, int k) constructor which would create the Markov model of order k from given text.
2. Int order_k()
3. Int freq(string k_gram)
4. Int freq(string k_gram, char c)
5. Char k_Rand(string k_gram)
6. String generate(string k_gram, int L)

I had a difficult time with this program. In fact I did not get all of it working properly. In the end when I ran the program all I could get it to output was "0." We also were supposed to use boost to test our code. Once again I failed to get this to work. Realistically, I believe if I devoted more time to working with the boost library I would have little issue with testing my code. Every time boost was required for an assignment it seemed more imperative to get the assignment done first, and boost was secondary. Below is a screenshot of the code I got working.

SCREENSHOT:

```
 1: CC = g++
 2: CFLAGS = -std=c++11 -c -g -Og -Wall -pedantic
 3: OBJ = RandWriter.o
 4: EXE = RandWriter
 5: LIBS = -lboost_unit_test_framework
 6:
 7: all: $(OBJ)
 8:         $(CC) $(OBJ) -o $(EXE) $(LIBS)
 9:
10: RandWriter.o: RandWriter.cpp test.cpp RandWriter.hpp
11:
12: clean:
13:         \rm $(OBJ) $(EXE)
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS6
 4:     Due Date: 11/15/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To produce class RandWriter using the Markov Chain
 8:
 9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
10:
11:
12: */
13:
14: #include "RandWriter.hpp"
15: #include <sstream>
16: #include <fstream>
17: #include <iostream>
18: #include <string>
19: #include <vector>
20: #include <cstdlib>
21: #include <cmath>
22: #include <map>
23: #include <iterator>
24:
25: RandWriter::RandWriter(std::string text, int k) {
26:     text = _text;
27:     k = _k;
28: }
29:
30: int RandWriter::order_k() {
31:     if(_k < 0)
32:         throw "Order must be a positive integer!";
33:         else
34:         return _k;
35: }
36:
37: int RandWriter::freq(std::string k_gram){
38:
39:     std::map<std::string, int> M;
40:
41:     std::string word = "";
42:     int frequency;
43:
44:     for (int i = 0; i < k_gram.size(); i++) {
45:         if (k_gram[i] == ' ') {
46:             if (M.find(word) == M.end()) {
47:                 frequency = 1;
48:                 M.insert(make_pair(word, frequency));
49:                 word = "";
50:             }
51:             else {
52:                 M[word] = frequency++;
53:                 word = "";
54:             }
55:         }
56:         else {
57:             word += k_gram[i];
58:         }
59:
60:     }
61:     return frequency;
```

```
   62: }
   63:
   64: int RandWriter::freq(std::string k_gram, char c){
   65:     int counter = 0;
   66:     if(RandWriter::order_k() == 0) {
   67:         for(int i = 0; i < _text.length(); i++) {
   68:             if(_text.c_str()[i] == c){
   69:                 counter++;
   70:             }
   71:         }
   72:     } else {
   73:         if(_text.find(k_gram)) {
   74:             int index = _text.find(k_gram);
   75:             int charPos = index + k_gram.length();
   76:             if(_text.c_str()[charPos] == c) {
   77:                 counter++;
   78:             }
   79:         }
   80:     }
   81:     return counter;
   82: }
   83:
   84: char RandWriter::k_Rand(std::string k_gram){
   85: // Outputing random text
   86:     char c;
   87:     int r;
   88:
   89:     srand (time(NULL));    // initialize the random number generator
   90:     r = rand() % 26;   // generate a random number
   91:     c = 'a' + r;            // Convert to a character from a-z
   92:
   93:     int count = RandWriter::freq(k_gram, c);
   94:     std::cout << c << "appears " << count << " times after " << k_gram << st
d::endl;
   95:     return c;
   96: }
   97:
   98: // ------------------------------------------------------------------
   99: // generate a string of length L characters
  100: // by simulating a trajectory through the corresponding
  101: // Markov chain. The first k characters of the newly
  102: // generated string should be the argument kgram.
  103: // Throw an exception if kgram is not of length k.
  104: // Assume that L is at least k.
  105: std::string RandWriter::generate(std::string k_gram, int L) {
  106:     std::vector<std::string> seedVec;
  107:     char ch;
  108:     std::string seed = "";
  109:     std::map<std::string,std::vector<char> > modelMap;
  110:     std::vector<char> charVector;
  111:     while(_text.find(ch) && seed.size()<RandWriter::order_k()) {
  112:         seed += ch;
  113:     }
  114:     seedVec.push_back(seed);
  115:     while(_text.find(ch)){
  116:         if(ch!='\n'){
  117:         modelMap[seed].push_back(ch);
  118:         seed = seed.substr(1) + ch;
  119:         seedVec.push_back(seed);
  120:         }
  121:     }
```

```
  122:     int count = 0;
  123:     std::string kGram = seedVec[rand()%seedVec.size()];
  124:     std::string randomText = kGram;
  125:
  126:     while(count<L){
  127:         char next;
  128:         std::vector<char> charVec = modelMap[kGram];
  129:         next = charVec[rand()%charVec.size()];
  130:         kGram = kGram.substr(1) + next;
  131:         randomText += next;
  132:         count++;
  133:     }
  134:   return randomText;
  135: }
  136:
  137: int main(int argc, char *argv[]) {
  138:     // Input from user about the markov model to be used
  139:     int orderK;
  140:     orderK = atoi(argv[1]);
  141:
  142:     int length;
  143:     length = atoi(argv[2]);
  144:
  145:     std::string fileName;
  146:     std::cin >> fileName;
  147:
  148:     // Reading the file from the computer
  149:     std::fstream input_file;
  150:     input_file.open(fileName.c_str());
  151:     std::string s = std::string((std::istreambuf_iterator<char>(input_file))
, std::istreambuf_iterator<char>());
  152:     RandWriter writer(s, orderK);
  153:     int count = writer.freq("ag");
  154:     std::cout << count << std::endl;
  155:     return 0;
  156: }
  157:
```

```
 1: /*
 2:     Thomas DeMasse
 3:     Computing IV - Assignment PS6
 4:     Due Date: 11/15/2021
 5:     Professor Dr. Yelena Rykalova
 6:
 7:     DESCRIPTION: To produce class RandWriter using the Markov Chain
 8:
 9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
10:
11:
12: */
13:
14: #ifndef _HOME_OSBOXES_DOCUMENTS_PS6_RANDWRITER_HPP_
15: #define _HOME_OSBOXES_DOCUMENTS_PS6_RANDWRITER_HPP_
16:
17: #include <string>
18: #include <iostream>
19:
20: class RandWriter {
21:
22:     public:
23:         RandWriter(std::string text, int k);
24:         int order_k();
25:         int freq(std::string k_gram);
26:         int freq(std::string k_gram, char c);
27:         char k_Rand(std::string k_gram);
28:         std::string generate(std::string k_gram, int L);
29:
30:
31:
32:     private:
33:         std::string _text;
34:         int _k;
35: };
36: #endif
```

```
 1:
 2:
 3: #include "RandWriter.hpp"
 4: #include <boost/test/included/unit_test.hpp>
 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(testOrder) {
11:
12:   RandWriter l(" ", 0);
13:   BOOST_CHECK(l.order_k());
14: }
15:
16: BOOST_AUTO_TEST_CASE(testFreq1){
17:   RandWriter l1("", 0)
18:   BOOST_CHECK(l1.freq());
19:
20: }
21:
22: BOOST_AUTO_TEST_CASE(testGenerate){
23:   RandWriter l2("", 0)
24:   BOOST_CHECK(l2.generate());
25:
26: }
27:
28: BOOST_AUTO_TEST_CASE(testRand){
29:   RandWriter l3("", 0)
30:   BOOST_CHECK(l3.k_Rand())
31:
32: }
```

PS7: Kronos – Intro to Regular Expression

Introduction to assignment: The goal of this assignment was simple. We were tasked with analyzing the Kronos time clock log by using regular expressions to parse the file. And then we were to verify device boot up timing.

DISCUSSION: The first thing I did for this assignment was to install the Boost regex library and all other necessary components to complete the assignment. Once I installed the necessary components I began working on the actual assignment. The program was fairly straightforward. It started by:

1. Creating the regex to insure the server started.
2. Creating the regex code to search for "oejs.AbstractConnector:Started SelectChannelConnector" – if this is found than the boot was successful.
3. Last create the report file of all the findings.

CONCLUSION: I really didn't find this program too challenging at all. After navigating my way through this course, I feel like I am miles ahead from where I began. At the beginning of the course I was extremely overwhelmed. By the end I felt like I could conquer any programming challenge thrown my way. That is good feeling. Below is a screenshot of my final program in Computing IV.

SCREENSHOT:

```
 1 BEGIN BOOT
 2 435369(device1_intouch.log): 2014-03-25 19:11:59 START
 3 435759(device1_intouch.log): 2014-03-25 19:15:02 FINISHED
 4    SUCCESS: 183000ms
 5
 6 BEGIN BOOT
 7 436500(device1_intouch.log): 2014-03-25 19:29:59 START
 8 436859(device1_intouch.log): 2014-03-25 19:32:44 FINISHED
 9    SUCCESS: 165000ms
10
11 BEGIN BOOT
12 440719(device1_intouch.log): 2014-03-25 22:01:46 START
13 440791(device1_intouch.log): 2014-03-25 22:04:27 FINISHED
14    SUCCESS: 161000ms
15
16 BEGIN BOOT
17 440866(device1_intouch.log): 2014-03-26 12:47:42 START
18 441216(device1_intouch.log): 2014-03-26 12:50:29 FINISHED
19    SUCCESS: 167000ms
20
21 BEGIN BOOT
22 442094(device1_intouch.log): 2014-03-26 20:41:34 START
23 442432(device1_intouch.log): 2014-03-26 20:44:13 FINISHED
24    SUCCESS: 159000ms
25
26 BEGIN BOOT
27 443073(device1_intouch.log): 2014-03-27 14:09:01 START
28 443411(device1_intouch.log): 2014-03-27 14:11:42 FINISHED
29    SUCCESS: 161000ms
30
```

```
    1: CC = g++
    2: OFLAGS = -g -Wall -ansi -pedantic -Werror -ansi -std=c++0x
    3: CFLAGS = -Wall -ansi -pedantic -Werror -ansi -std=c++0x -lboost_unit_test_fr
amework
    4: BFLAGS = -lboost_regex -lboost_date_time -lboost_unit_test_framework
    5:
    6: all: main
    7:
    8: main: main.o
    9:         $(CC) main.o $(CFLAGS) $(BFLAGS) -o main
   10:
   11: main.o: main.cpp
   12:         $(CC) -c main.cpp $(OFLAGS) -o main.o
   13:
   14: clean:
   15:         \rm -f *.o  *~ ps7a *.rpt
```

```
    1: /*
    2:     Thomas DeMasse
    3:     Computing IV - Assignment PS6
    4:     Due Date: 11/15/2021
    5:     Professor Dr. Yelena Rykalova
    6:
    7:     DESCRIPTION: To produce class RandWriter using the Markov Chain
    8:
    9:     Copyright [2021] <Copyright Thomas DeMasse>"  [legal/copyright]
   10:
   11:
   12: */
   13:
   14: #include <iostream>
   15: #include <exception>
   16: #include <fstream>
   17: #include <string>
   18: #include <boost/regex.hpp>
   19: #include <boost/date_time/gregorian/gregorian.hpp>
   20: #include <boost/date_time/posix_time/posix_time.hpp>
   21:
   22:
   23: using boost::gregorian::date;
   24: using boost::gregorian::years;
   25: using boost::gregorian::months;
   26: using boost::gregorian::days;
   27: using boost::gregorian::from_simple_string;
   28: using boost::gregorian::date_duration;
   29: using boost::gregorian::date_period;
   30: using boost::posix_time::ptime;
   31: using boost::posix_time::hours;
   32: using boost::posix_time::minutes;
   33: using boost::posix_time::seconds;
   34: using boost::posix_time::time_duration;
   35:
   36: int main(int argc, char* argv[]) {
   37:   std::ifstream logFile(argv[1]);
   38:   std::string fileName = argv[1];
   39:   std::ofstream reportFile(fileName + ".rpt", std::ofstream::out);
   40:   std::string line;
   41:   date _date;
   42:   ptime timeStart;
   43:   if (argc != 2) {
   44:     throw std::runtime_error("Error!");
   45:   } else {
   46:     bool boot = false;
   47:     int lineNum = 1;
   48:     boost::regex begin(
   49:       "([0-9]+)-([0-9]+)-([0-9]+) "
   50:       "([0-9]+):([0-9]+):([0-9]+): "
   51:       "\\(log.c.166\\) server started.*");
   52:     boost::regex end(
   53:       "([0-9]+)-([0-9]+)-([0-9]+) "
   54:       "([0-9]+):([0-9]+):([0-9]+).([0-9]+):INFO:"
   55:       "oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*"
);
   56:     if (logFile.is_open()) {
   57:       while (getline(logFile, line)) {
   58:         boost::smatch match;
   59:         if (regex_match(line, match, begin)) {
   60:           date _tempDate(stoi(match[1]), stoi(match[2]), stoi(match[3]));
```

```
   61:             _date = (_tempDate);
   62:           ptime _tempStart(_date,
   63:           time_duration(stoi(match[4]), stoi(match[5]), stoi(match[6])));
   64:           timeStart = _tempStart;
   65:           if (boot) {
   66:             reportFile << "EPIC FAILURE\n" << std::endl;
   67:             boot = false;
   68:           }
   69:           reportFile << "BEGIN BOOT\n"
   70:                   << lineNum << "(" << argv[1] << "): "
   71:                   << match[1] << "-" << match[2] << "-" << match[3]
   72:                   << " "
   73:                   << match[4] << ":" << match[5] << ":" << match[6]
   74:                   << " START" << std::endl;
   75:           boot = true;
   76:         } else if (regex_match(line, match, end)) {
   77:           date _tempDateFinish(stoi(match[1]), stoi(match[2]), stoi(match[3]
));
   78:           ptime timeEnd(_tempDateFinish,
   79:           time_duration(stoi(match[4]), stoi(match[5]), stoi(match[6])));
   80:           reportFile << lineNum << "(" << argv[1] << "): "
   81:                   << match[1] << "-" << match[2] << "-" << match[3]
   82:                   << " "
   83:                   << match[4] << ":" << match[5] << ":" << match[6]
   84:                   << " FINISHED" << std::endl;
   85:           time_duration _td = timeEnd - timeStart;
   86:           int _time = _td.total_milliseconds();
   87:           reportFile << "    "
   88:                   << "SUCCESS: " << _time << "ms\n" << std::endl;
   89:           boot = false;
   90:         }
   91:         lineNum++;
   92:       }
   93:       reportFile.close();
   94:       logFile.close();
   95:     } else {
   96:       throw std::runtime_error("Failed to open file");
   97:     }
   98:   }
   99:   return 0;
  100: }
```