COMP IV Sec 203: Project Portfolio

THARUNI DONAPATI

Fall 2022

Contents

Time to Complete Portfolio: 11 hours

1 PS0: Hello World with SFML

1.1 Discussion

Figure 1: This is my sprite image

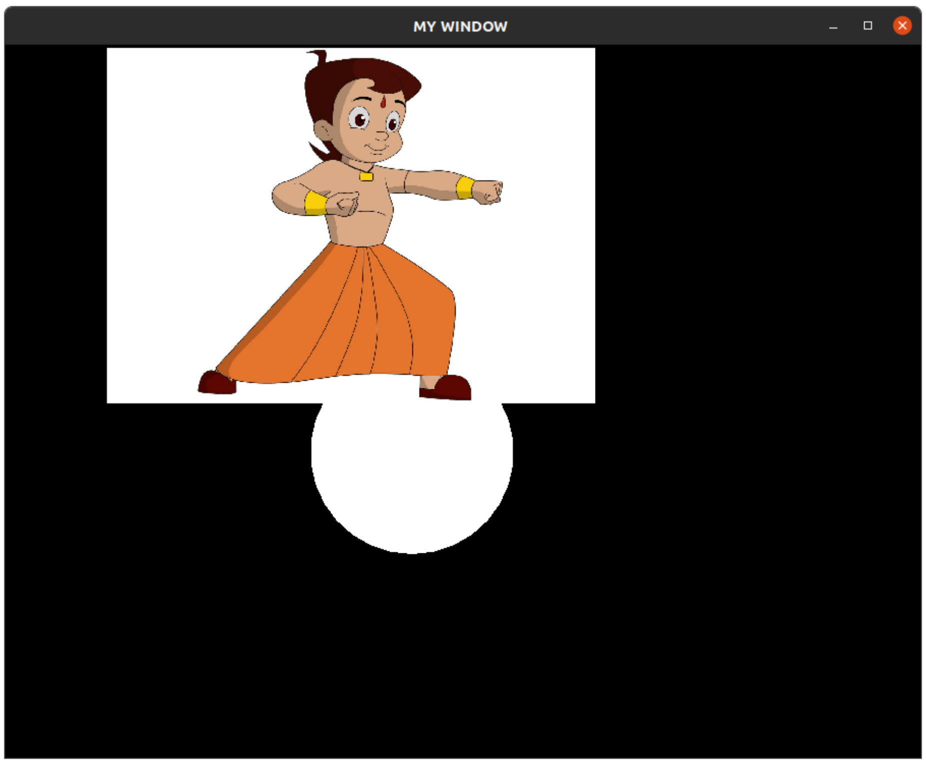

Figure 2: This is my output image

1.2 Assignment Description

This assignment served as both a general introduction to the course and a practice exercise for some of the foundational SFML features. Input from the keyboard, the importation of picture files, as well as drawing and modifying objects in the display area are a few examples of such features.

In the project I created for this assignment, I randomly loaded an image into the computer and gave it the name "sprite.jpg," allowing it to react to keystrokes. Pressing the up, down, left, and right arrows and the A and B keys will move and rotate the image, respectively.

1.3 Key Algorithms, Data Structures, OO Design, Etc.

Utilizing SFML's window object and render loop was a crucial component of this assignment that we had to work with and put to use. As long as the window is active, this loop will keep running and check every frame to see what events the built-in event handler has picked up. By reading the kind of the Event object that this event handler produces, we can decide if and how to change the sprite that is being drawn to the output screen. This operation is repeated when the revised sprite has been redrawn.

1.4 What I've Learned

The SFML library was totally unknown to me prior to enrolling in this course, therefore this task performed a great job of teaching me the fundamentals of the library. The new information I had acquired and the experience I had with the window's render loop would be beneficial to me in multiple upcoming tasks that required the use of the library itself, whether it be for projects requiring various render loops or just as a simple structure.

1.5 Codebase

Main.cpp

```
// Tharuni Donapati
#include <SFML/Graphics.hpp>
#include<iostream>
int main()
{
    sf::RenderWindow window(sf::VideoMode(900,700), "MY WINDOW ");
    sf::CircleShape shape(100.f);
    shape.setPosition( 300,300 );
    shape.setFillColor(sf::Color::White);
    sf::Texture texture;
    if(!texture.loadFromFile("sprite.jpg"))
    {
    return EXIT_FAILURE;
     }
    sf::Sprite sprite(texture);
    sprite.setScale(0.3, 0.3);
    sprite.setPosition(100.0f,0);
while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
          if(sf::Event::KeyPressed)
        {
        std::cout<<"key pressed"<<std::endl;
        if(event.key.code ==  sf::Keyboard::A)
        {
        sprite.rotate(30);
        }
       else if(event.key.code == sf::Keyboard::B)
       {
        sprite.rotate(-30);
        }
        if( sf::Event::KeyReleased)
        {
        std::cout<<"key released"<<std::endl;
        }
        }
           if (event.type == sf::Event::Closed)
                window.close();
                }
        if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::Left))
        {
        sprite.move(-1,0);
        }
        if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::Right))
        {
        sprite.move(1,0);
        }
        if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::Up))
        {
        sprite.move(0,-1);
```

```cpp
        }
        if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::Down))
        {
        sprite.move(0,1);
        }
         window.clear();
        window.draw(shape);
        window.draw(sprite);
        window.display();
    }
    return 0;
}
```

2 PS1a: linear feedback shift register.

2.1 Assignment Description

The ps1a assignment is an implementation of the Fibonacci LFSR (Linear Feedback shift Register), which is utilized for the ps1b, PhotoMagic. The two primary methods in this project are step() and generate(). The step() function is used for left shifting the one bit of the provided seed, along the lsb is the result of the tap places. These tap places employ XOR procedures to get the lsb result. The create() function generates states based on the k inputs.

2.2 Key Algorithms, Data Structures, OO Design, Etc.

In this project, to represent register bits I used vectors because I feel it very easy to use other than that I also used this over a dynamically allocated array and I used string for the seed as mentioned in the ps1a pdf. In short I followed the instructions given:

```
   class FibLFSR {
public:
FibLFSR(string seed); // constructor to create LFSR with
                      // the given initial seed
int step();           // simulate one step and return the
                      // new bit as 0 or 1
int generate(int k);  // simulate k steps and return
                      // k-bit integer
private: ...
}
```

2.3 What I've Learned

From this assignment I've learned about how LFSR (Linear Feedback shift Register) works and it's mechanism.I also learned that how this LFSR is going to be used for the Image encoding and decoding of the Image in PS1b assignment code. Where we utilize this part of assignment.

2.4 Codebase

Makefile

```
CC = g++
CFLAGS = -Wall -Werror -std=c++11 -pedantic
BOOST_ARG = -lboost_unit_test_framework
OBJ = FibLFSR.o test.o
EXE = ps1a

all: ps1a

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

test.o: test.cpp
    $(CC) $(CFLAGS) -c test.cpp -o test.o $(BOOST_ARG)

ps1a: $(OBJ)
    $(CC) $(CFLAGS) -o $(EXE) $(OBJ) $(BOOST_ARG)

clean:
    rm -f $(EXE) $(OBJ) *.o
```

FibLFSR.cpp

```cpp
#include "FibLFSR.h"

const int As0 = 48;
const int As1 = 49;

const int Total_bits = 16;
const int LHS_bit = 0;

const int tap1= 2;
const int tap2= 3;
const int tap3 = 5;

FibLFSR::FibLFSR(std::string seed)
{
    tap_bits[0] = tap1;
    tap_bits[1] = tap2;
    tap_bits[2] = tap3;
    tap_bits[3] = LHS_bit;
    finalvalue= 0;
    vec_Len = seed.length();
    int i = 0;
    while (i < vec_Len)
    {
        if (seed[i] == As0 || seed[i] == As1)
        {
        bitVector.push_back(seed[i] - As0) ;
        }
        i++;
    }
}
int FibLFSR::step(void)
{
    int xorvalue = 0;
    xorvalue = (bitVector[tap1] ^ bitVector[tap2]) ^ (bitVector[tap3] ^
   bitVector[LHS_bit]);
    bitVector.erase(bitVector.begin());
    bitVector.push_back(xorvalue);
    finalvalue= xorvalue;
    return xorvalue;
}
int FibLFSR::generate(int k)
{
    int resultInt= 0;
        int i = 0;
    while(i < k)
    {
        resultInt = (resultInt << 1) + this->step();
        i++;
    }
    finalvalue= resultInt;
    return resultInt;
}
std::ostream& operator<< (std::ostream& outputStream, const FibLFSR& fibLFSR
   ){
    int i = 0;
    while(i < fibLFSR.vec_Len)
    {
        outputStream << fibLFSR.bitVector[i];
        i++;
    }
    outputStream << "\t" << fibLFSR.finalvalue;
    return outputStream;}
```

FibLFSR.h

```

#define FIBLFSR_H

#include <iostream>

#include <string>

#include <vector>


class FibLFSR {
public:
    FibLFSR(std::string seed);
    int step(void);
    int generate(int k);

    friend std::ostream& operator<< (std::ostream& outputStream, const
   FibLFSR& fibLFSR);

private:
    std::vector<int> bitVector;
    int vec_Len;
    int tap_bits[4];
    int finalvalue;
};


std::ostream& operator<< (std::ostream& outputStream, const FibLFSR& fibLFSR
   );
```

test.cpp

```
#include <iostream>

#include <string>

#include "FibLFSR.h"

#define BOOST_TEST_DYN_LINK

#define BOOST_TEST_MODULE Main

#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
    FibLFSR l("1011011000110110");
    BOOST_REQUIRE(l.step() == 0);
    BOOST_REQUIRE(l.step() == 0);
    BOOST_REQUIRE(l.step() == 0);
    BOOST_REQUIRE(l.step() == 1);
    BOOST_REQUIRE(l.step() == 1);
    BOOST_REQUIRE(l.step() == 0);
    BOOST_REQUIRE(l.step() == 0);
    BOOST_REQUIRE(l.step() == 1);
    FibLFSR l2("1011011000110110");
    BOOST_REQUIRE(l2.generate(9) == 51);}
```

```
BOOST_AUTO_TEST_CASE(fifteenBits)
{
    FibLFSR l("1001001001");
    std::cout<<l;
}


BOOST_AUTO_TEST_CASE(sixteenBitsWithNonbinaryDigits)
{
        FibLFSR l("1234567890987654");
       std::cout<<l;
}
```

3 PS1b: PhotoMagic

3.1 Assignment Description

The ps1b utilizes the ps1a i.e LFSR: LINEAR FEEDBACK SHIFT REGISTER as a supporting for the encoding and decoding of the image given as input.This project takes the input image i.e, encode.png and Encrypts using the LFSR Randomizer and encrypts each and every pixel of the image and saves to the decode.png file. To encrypt and decode the image for this project there are Two key functions used:

1. the transform function, which applies the LSFR to image transformation. Every bit in the image has its 8bit LSFR and RGB values xored.

2. The main function, which accepts three commands—two pictures and a 16-bit LSFR seed value—takes three arguments.

The image was decoded, presented after encryption, and saved as a decoded image.

Figure 3: This is the image I used

3.2 Key Algorithms, Data Structures, OO Design, Etc.

In the LFSR, I chose The Vector since I found it to be more flexible and much easier to use. Additionally, to attain the ideal result, the example code was modified to become the final code. I utilized two windows and two sprites to demonstrate the difference between the typical cat image and the encoded and decoded images by using Image and its functions, a letter given to Texture, then to Sprite, to draw on the window. Encoding and decoding are done using the following code:

```
void transform(sf::Image& image, FibLFSR* seedbit);
void transform(sf::Image& image, FibLFSR* seedbit)
{
unsigned int x;
unsigned int y;
sf::Vector2u size = image.getSize();
sf::Color change;
for(x = 0; x < size.x; x++)
{
for(y = 0; y < size.y; y++)
{
```

Figure 4: This is my decoded image

```
change = image.getPixel(x,y);
change.r = change.r ^ seedbit->generate(8);
change.b = change.b ^ seedbit->generate(8);
change.g = change.g ^ seedbit->generate(8);
image.setPixel(x,y,change);
}
}
}
```

3.3 What I've learned

I acquired the knowledge of using two windows for various outputs but I'm supposed to display both the encoded and decoded image on single window which I didn't do. I was able to understand how the LFSR Randomizer and the image's pixels may be used to encode and decode data. I also learned about how the pixels were calculated mathematically.

3.4 Codebase

Makefile

```
CC = g++
Debug = -g
CFLAGS = -c -Wall -Werror -std=c++11
OBJECTS = FibLFSR.o PhotoMagic.o
LFLAGS = -lsfml-graphics -lsfml-window -lsfml-audio -lsfml-system

all: PhotoMagic

PhotoMagic: $(OBJECTS)
	g++ -Wall $(OBJECTS) -o PhotoMagic $(LFLAGS)

PhotoMagic.o: PhotoMagic.cpp
	$(CC) $(CFLAGS) -c PhotoMagic.cpp -o PhotoMagic.o $(LFLAGS)

FibLFSR.o: FibLFSR.h FibLFSR.cpp
```

```
	$(CC) $(CFLAGS) -c FibLFSR.cpp -o FibLFSR.o $(LFLAGS)
clean:
	rm $(OBJECTS) PhotoMagic
```

PhotoMagic.cpp

```cpp
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <algorithm>

#include "FibLFSR.h"

#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>

using namespace std;

void transform(sf::Image& image, FibLFSR* seedbit);
void transform(sf::Image& image, FibLFSR* seedbit)
{
unsigned int x;
unsigned int y;

sf::Vector2u size = image.getSize();

sf::Color change;

for(x = 0; x < size.x; x++)
{
for(y = 0; y < size.y; y++)
{
change = image.getPixel(x,y);

change.r = change.r ^ seedbit->generate(8);
change.b = change.b ^ seedbit->generate(8);
change.g = change.g ^ seedbit->generate(8);

image.setPixel(x,y,change);
}
}
}
int main()
{
string input;

string output;

string LSFR;
cout <<" % PhotoMagic :-";

std::cin>>input;
std::cin.ignore(1, ' ');

std::cin>>output;
std::cin.ignore(1, ' ');

std::cin>>LSFR;
```

```cpp

sf::String input_image = input;
sf::String output_image = output;

FibLFSR seedbit(LSFR);

sf::Image input_file;

if(!input_file.loadFromFile(input_image))
{
return -1;
}
sf::Image output_file;

if(!output_file.loadFromFile(input_image))
{
return -1;
}
transform(output_file, &seedbit);

sf::Vector2u size = input_file.getSize();
sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "Image");

sf::Texture texture;
texture.loadFromImage(input_file);

sf::Sprite sprite;
sprite.setTexture(texture);

sf::Vector2u size1 = output_file.getSize();
sf::RenderWindow window2(sf::VideoMode(size1.x, size1.y), "Decoded Image");

sf::Texture texture1;
texture1.loadFromImage(output_file);

sf::Sprite sprite1;
sprite1.setTexture(texture1);

while (window1.isOpen() && window2.isOpen())
{
sf::Event event;
while (window1.pollEvent(event))
 {
 if (event.type == sf::Event::Closed)
 window1.close();
        }
while (window2.pollEvent(event))
{
if (event.type == sf::Event::Closed)
window2.close();
}
window1.clear();
window1.draw(sprite);
window1.display();

window2.clear();
window2.draw(sprite1);
window2.display();
}
```

```

if(!output_file.saveToFile(output_image))
{
return -1;
}
return 0;
}
```

FibLFSR.cpp

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

using namespace std;

class FibLFSR
{
public:

FibLFSR(string seedbit);

~FibLFSR()
{
    delete B;
}
int step();
int generate(int k);

friend std::ostream& operator<< (std::ostream &out, const FibLFSR &FibLFSR);

private:

int *B;
};
```

FibLFSR.h

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "FibLFSR.h"

using namespace std;

FibLFSR::FibLFSR(string seedbit)
{
if(seedbit.length() == 16)
{
B = new int[16];
if (B != NULL)
{
 int i =0;
while((int)i < (int)seedbit.length())
{
B[i] = seedbit.at(i) - '0';
i++;
}
}
}
else
{
B = NULL;
}
}
int FibLFSR::step()
{
int i;
if (B != NULL)
{
int copyB[16];
int newB = ((B[0]^B[2])^B[3])^B[5];
copyB[15] = newB;

for(i = 0; i < 15; i++)
{
copyB[i] = B[i+1];
}
for(i = 0; i < 16; i++)
{
B[i] = copyB[i];
}
return B[14];
}
else return -1;
}

int FibLFSR::generate(int k)
{
if (B != NULL && k > 0)
{
int stepB;
int sum = 0;
while(k > 0)
{
stepB = step();
sum += (stepB * pow(2, k-1));
k--;
}
return sum;
}
else return -1;
}
std::ostream& operator<< (std::ostream &out, const FibLFSR &FibLFSR)
{
int i=0;
while(i < 16)
{
out << FibLFSR.B[i];
i++;
}
return out;
}
```

4 PS2: Triangle Fractal

4.1 Assignment Description

In this ps2 assignment we should write a program that plots a Triangle Fractal. Writing a program called snowFlake.cpp containing the recursive method snowFlake() and a main() program that invokes the recursive function is our assignment. Instead of having snowFlake() itself recursive, you might construct a recursive helper function that snowFlake() calls.

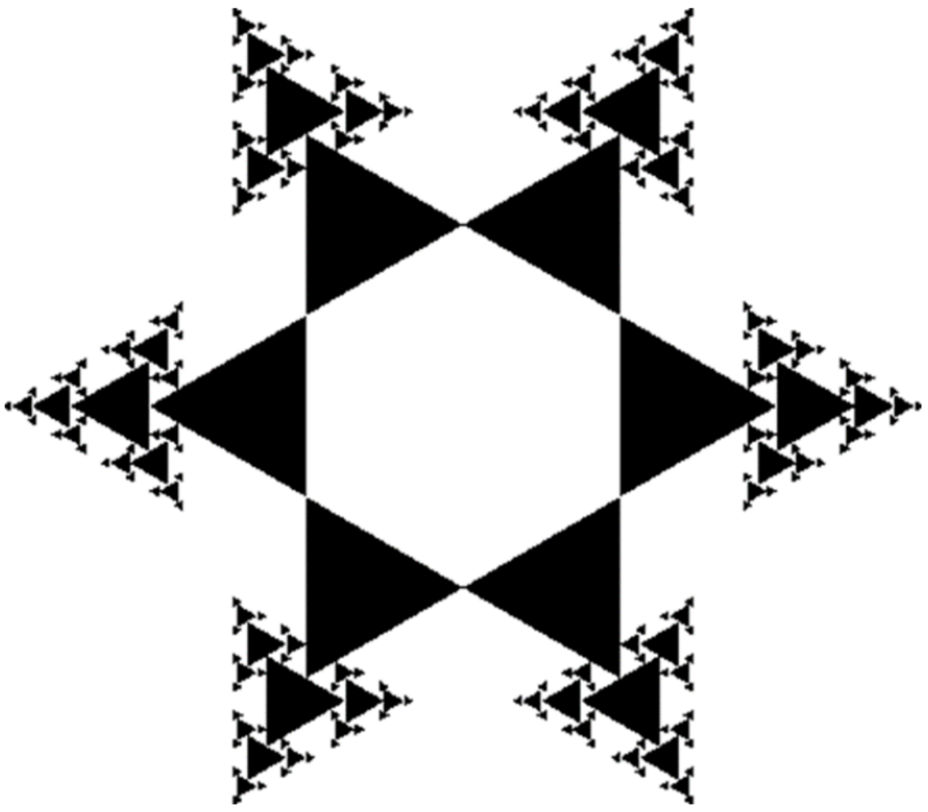

Figure 5: This is snowflake

4.2 Problems I faced

I could not complete this assigment because I was not confident that I could complete this task and I felt a difficult in ploting the logic to create this snow flake.

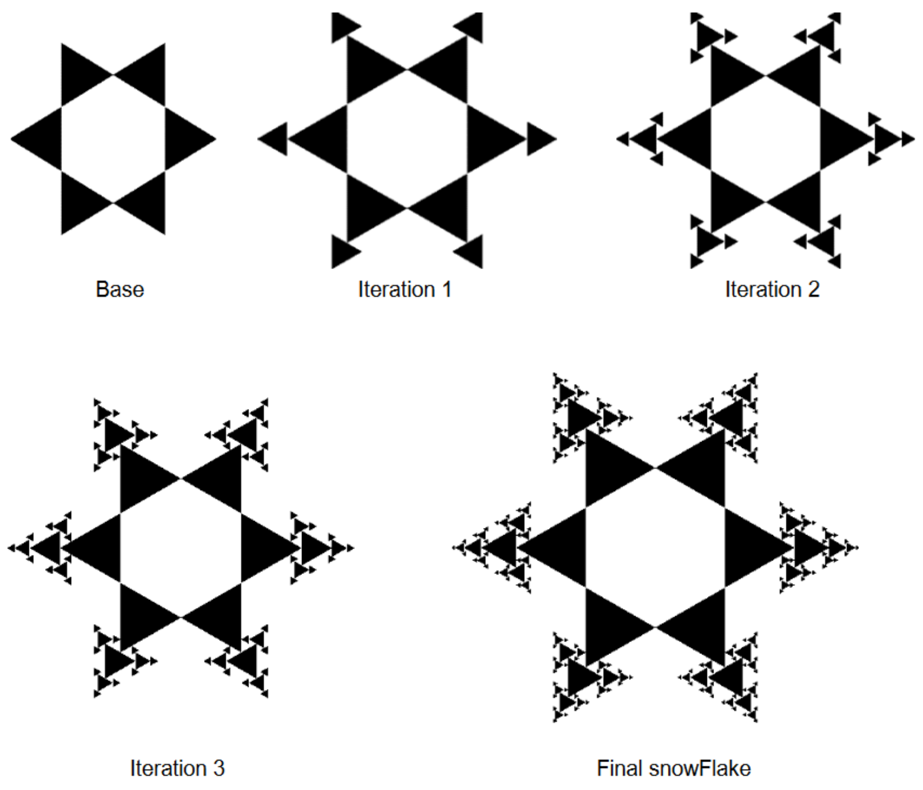

Figure 6: These are the iterations

5 PS3a: Circular Array

5.1 Assignment Description

The ps3a assignment generates a circular buffer that will be utilized in the ps3b assignment. In this assignment, we essentially design a circular queue with connected head and tail nodes that uses the FIFO (First In First Out) queuing method.

5.2 Key Algorithms, Data Structures, OO Design, Etc.

In this project, I created a program that uses the Karplus-Strong algorithm to imitate plucking a guitar string. In the implementation, I also used exceptions and unit testing. used cpp linting as well. The majority of the code uses if-else statements to check for exceptions before executing a member of the dequeue function. I utilized Lambda Expression in my CircularBuffer.cpp file in the e value constructor to make my ringbuffer use less time and the functions I built should take constant time.

5.3 What I learned

Since I'm already aware of Circular Buffer concept theoretically, by this assignment I got a chance to implement practically using the given guidelines in the project description.

5.4 Codebase

Makefile

```
CC = g++
CFLAGS = -Wall -Werror -std=c++11 -pedantic -O1 -g
BOOST_ARG = -lboost_unit_test_framework
OBJ = CircularBuffer.o test.o
EXE = ps3

all: $(EXE)

$(EXE): $(OBJ)
    $(CC) $(CFLAGS) -o $(EXE) $(OBJ) $(BOOST_ARG)

ps4a.o: CircularBuffer.cpp CircularBuffer.h
    $(CC) $(CFLAGS) -c CircularBuffer.cpp -o CircularBuffer.o

test.o: test.cpp
    $(CC) $(CFLAGS) -c test.cpp -o test.o $(BOOST_ARG)

clean:
    rm $(EXE) *.o

```

CircularBuffer.cpp

```
// Copyright 2022 Tharuni

#include "CircularBuffer.h"
#include <stdexcept>

const int e_value = 1337;

CircularBuffer::CircularBuffer(int cap) {
        auto checkBadInput = [=]() {
        if (cap > 0) {
            max_container_size = cap;
        } else {
            throw std::invalid_argument
            ("capacity must be greater than zero.");
        }
    };

    checkBadInput();
}

int16_t CircularBuffer::dequeue(void) {
    int16_t frontVal = 0;
    if (!isEmpty()) {
        frontVal = container.front();
        container.pop_front();
    }  else {
        throw std::runtime_error("can't dequeue");
        frontVal = e_value;
    }

    return frontVal;
}

void CircularBuffer::enqueue(int16_t x) {
    if (!isFull()) {
        container.push_back(x);
    } else {
        throw std::runtime_error("can't enqueue");
    }
}

int16_t CircularBuffer::peek(void) {
    if (!isEmpty()) {
        return container.front();
    } else {
        throw std::runtime_error("can't peek");
        return e_value;
    }
}

int CircularBuffer::size(void) {
    return container.size();
}

bool CircularBuffer::isFull(void) {
    return (container.size() == max_container_size);
}

bool CircularBuffer::isEmpty(void) {
    return container.empty();
}
```

CircularBuffer.h

```
// Copyright 2022 Tharuni

#ifndef  CIRCULARBUFFER_H_
#define  CIRCULARBUFFER_H_
#include <stdint.h>
#include <iostream>
#include <deque>

class CircularBuffer {
 public:
     explicit CircularBuffer(int cap);
     int16_t dequeue(void);
     void enqueue(int16_t x);
     int16_t peek(void);
     int size(void);
     bool isFull(void);
     bool isEmpty(void);
 private:
     std::deque<int16_t> container;
     int max_container_size;
};
#endif  // CIRCULARBUFFER_H_
```

test.cpp

```
// Copyright 2022 Tharuni

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

BOOST_AUTO_TEST_CASE(testFunctionsWork) {
    CircularBuffer testObj(3);
    testObj.enqueue(1);
    testObj.enqueue(2);
    testObj.enqueue(3);
    BOOST_REQUIRE(testObj.size() == 3);
    BOOST_REQUIRE(testObj.isEmpty() == false);
    BOOST_REQUIRE(testObj.isFull() == true);
    BOOST_REQUIRE(testObj.dequeue() == 1);
    BOOST_REQUIRE(testObj.peek() == 2);
}
BOOST_AUTO_TEST_CASE(testExceptionsThrow) {
    BOOST_CHECK_THROW(CircularBuffer testObj(0), std::invalid_argument);
    BOOST_CHECK_THROW(CircularBuffer testObj(-1), std::invalid_argument);
    CircularBuffer testObj(1);
    BOOST_CHECK_THROW(testObj.dequeue(), std::runtime_error);
    BOOST_CHECK_THROW(testObj.peek(), std::runtime_error);
    testObj.enqueue(1);
    BOOST_CHECK_THROW(testObj.enqueue(2), std::runtime_error);
}
BOOST_AUTO_TEST_CASE(testExceptionsDontThrow) {
    BOOST_CHECK_NO_THROW(CircularBuffer testObj(1));
    CircularBuffer testObj(1);
    BOOST_CHECK_NO_THROW(testObj.enqueue(1));
    BOOST_CHECK_NO_THROW(testObj.peek());
    BOOST_CHECK_NO_THROW(testObj.dequeue());
}
```

6 PS3b: String Sound

6.1 Assignment Description

This project uses the SFML package "Audio" to generate sound, and we use our keyboards to create notes. In this project, the ps3a CircularBuffer is used. The primary goal of this project is to build a simulation of a guitar being played, however since I wanted to get additional credit, I produced a piano simulation note. The Karplus-Strong algorithm is the main component of this research.

6.2 Key Algorithms, Data Structures, OO Design, Etc.

The Vector is used to retain the sound tempo, create a sound buffer, and sample audio. Key inputs are sent to a basic switch function, which then calculates the frequency and returns an audio. In developing this assignment, new functions were used extensively.

6.3 What I've learned

I learnt how the frequency can be use to create different instruments. I understood how to take the sample inputs from keyboard and produce sound.

6.4 Codebase

Makefile

```
CC = g++
CFLAGS = -Wall -Werror -std=c++11 -pedantic -O2 -g
#BOOST_ARG = -lboost_unit_test_framework
SFML = -lsfml-system -lsfml-audio -lsfml-graphics -lsfml-window
OBJ = CircularBuffer.o StringSound.o KSGuitarSim.o
EXE = KSGuitarSim

all: $(EXE)

$(EXE): $(OBJ)
    $(CC) $(CFLAGS) -o $(EXE) $(OBJ) $(SFML)

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

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

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

clean:
    rm $(EXE) *.o

```

KSGuitarSim.cpp

```
/* Copyright 2015 Fred Martin, Y.
  Rykalova, 2020
  Tharuni Donapati, 2022
*/
#include "CircularBuffer.h"
#include "StringSound.h"
#include <limits.h>
#include <string>
#include <exception>
#include <SFML/Graphics.hpp>
```

```cpp
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#define CONCERT_A 220.0
#define SAMPLES_PER_SEC 44100
std::vector<sf::Int16> makeSamples(StringSound& gs) {
  std::vector<sf::Int16> samples;
  gs.pluck();
  int duration = 8;
  int i;
  for (i = 0; i < SAMPLES_PER_SEC * duration; i++) {
    gs.tic();
    samples.push_back(gs.sample());
  }
  return samples;
}
int main() {
  sf::RenderWindow window(sf::VideoMode(300, 300),
  "MY WINDOW");
  sf::Event event;
  std::vector<sf::Int16> samples;
  double freq = CONCERT_A;
  StringSound gs1(freq);
  sf::Sound sound1;
  sf::SoundBuffer buf1;
  samples = makeSamples(gs1);
  if (!buf1.loadFromSamples(&samples[0], samples.size(), 2, SAMPLES_PER_SEC)
   )
      throw std::runtime_error("sf::SoundBuffer: failed to load from samples
   .");
  sound1.setBuffer(buf1);
  while (window.isOpen()) {
    while (window.pollEvent(event)) {
      switch (event.type) {
      case sf::Event::Closed:
        window.close();
        break;
      case sf::Event::KeyPressed:
        switch (event.key.code) {
        case sf::Keyboard::Q:
        case sf::Keyboard::Num2:
        case sf::Keyboard::W:
        case sf::Keyboard::E:
        case sf::Keyboard::Num4:
        case sf::Keyboard::R:
        case sf::Keyboard::Num5:
        case sf::Keyboard::T:
        case sf::Keyboard::Y:
        case sf::Keyboard::Num7:
        case sf::Keyboard::U:
        case sf::Keyboard::Num8:
        case sf::Keyboard::I:
        case sf::Keyboard::Num9:
        case sf::Keyboard::O:
        case sf::Keyboard::P:
        case sf::Keyboard::Hyphen:
        case sf::Keyboard::LBracket:
        case sf::Keyboard::Equal:
        case sf::Keyboard::Z:
        case sf::Keyboard::X:
```

```
        case sf::Keyboard::D:
        case sf::Keyboard::C:
        case sf::Keyboard::F:
        case sf::Keyboard::V:
        case sf::Keyboard::G:
        case sf::Keyboard::B:
        case sf::Keyboard::N:
        case sf::Keyboard::J:
        case sf::Keyboard::M:
        case sf::Keyboard::K:
        case sf::Keyboard::Comma:
        case sf::Keyboard::Period:
        case sf::Keyboard::Semicolon:
        case sf::Keyboard::Slash:
        case sf::Keyboard::Quote:
        case sf::Keyboard::Space:
          sound1.play();
          break;
        default:
          break;
        }
      default:
        break;
      }
      window.clear();
      window.display();
    }
  }
  return 0;
}
```

CircularBuffer.cpp

```
// Copyright 2022 Tharuni Donapati

#ifndef  CIRCULARBUFFER_H_
#define  CIRCULARBUFFER_H_

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

class CircularBuffer {
 public:
     explicit CircularBuffer(int cap);
     int size(void);
     bool isEmpty(void);
     bool isFull(void);
     void enqueue(int16_t x);
     int16_t dequeue(void);
     int16_t peek(void);
     inline std::deque<int16_t> getContainer(void) const {
         return container;
     }
     inline unsigned int getMaxContainerSize(void) const {
         return max_container_size;
     }

 private:
     std::deque<int16_t> container;
     unsigned int max_container_size;
};

#endif  //  CIRCULARBUFFER_H_
```

CircularBuffer.h

```
// Copyright 2022 Tharuni Donapati

#include "CircularBuffer.h"
#include <stdexcept>

const int e_value = 1337;

CircularBuffer::CircularBuffer(int cap) {
        auto checkBadInput = [=]() {
        if (cap > 0) {
            max_container_size = cap;
        } else {
            throw std::invalid_argument
            ("cap must be greater than zero");
        }
    };

    checkBadInput();
}

int16_t CircularBuffer::dequeue(void) {
    int16_t frontVal = 0;
    if (!isEmpty()) {
        frontVal = container.front();
        container.pop_front();
    } else {
        throw std::runtime_error("cannot dequeue");
        frontVal = e_value;
    }

    return frontVal;
}

void CircularBuffer::enqueue(int16_t x) {
    if (!isFull()) {
        container.push_back(x);
    } else {
        throw std::runtime_error("cannot Enqueue");
    }
}

int16_t CircularBuffer::peek(void) {
    if (!isEmpty()) {
        return container.front();
    } else {
        throw std::runtime_error("cannot peek");
        return e_value;
    }
}
int CircularBuffer::size(void) {
    return container.size();
}

bool CircularBuffer::isEmpty(void) {
    return container.empty();
}

bool CircularBuffer::isFull(void) {
    return (container.size() == max_container_size);
}
```

StringSound.h

```
// Copyright 2022 Tharuni Donapati

#ifndef  STRINGSOUND_H_
#define  STRINGSOUND_H_

#include "CircularBuffer.h"
#include <math.h>
#include <vector>
#include <SFML/Audio.hpp>

class StringSound {
 public:
    explicit StringSound(double frequency);
    explicit StringSound(std::vector<sf::Int16> init);
    ~StringSound(void);
    void pluck(void);
    void tic(void);
    sf::Int16 sample(void);
    int time(void);
 private:
    CircularBuffer* cb;
    int tym;
};

#endif  // STRINGSOUND_H_
```

StringSound.cpp

```
// Copyright 2022 Tharuni Donapati

#include "CircularBuffer.h"
#include "StringSound.h"
#include <random>
#define SAMPLING_RATE 44100
#define ENERGY_DECAY_FACTOR 0.996
#define HALF 0.5
StringSound::StringSound(double freq) {
    if (freq <= 0) {
        throw std::runtime_error
        ("Can't have 0 or negative frequency");
    } else {
        cb = new CircularBuffer(std::ceil(SAMPLING_RATE / freq));
        tym = 0;
    }
}
StringSound::StringSound(std::vector<sf::Int16> init) {
    cb = new CircularBuffer(init.size());
    unsigned int i = 0;
    while (i < init.size()) {
        (*cb).enqueue(init[i]);
        ++i;
    }
    tym = 0;
}
```

```cpp
StringSound::~StringSound(void) {
    delete cb;
    tym = 0;
}
void StringSound::pluck(void) {
    if ((*cb).isFull()) {
        (*cb).getContainer().clear();
    }
    auto generateRandomNums = [=] () {
      std::random_device rd;
      std::mt19937 gen(rd());
      std::uniform_int_distribution<int16_t> dist(-32768, 32767);
      while (!(*cb).isFull()) {
        (*cb).enqueue(dist(gen));
      }
    };
    generateRandomNums();
}

void StringSound::tic(void) {
    pluck();
    sf::Int16 oldFrontVal = (*cb).dequeue();
    sf::Int16 newVal = ENERGY_DECAY_FACTOR *
      (HALF * (oldFrontVal + (*cb).peek()));
    (*cb).enqueue(newVal);
    tym = tym + 1;
}
sf::Int16 StringSound::sample(void) {
    if ((*cb).isEmpty()) {
        throw std::runtime_error
        ("Cannot peek");
        return 0;
    }
    return (*cb).peek();
}
int StringSound::time(void) {
    return tym;
}
```

7 PS4a: Sokoban - The Beginning

7.1 Assignment Description

In this assignment we code to create a sokoban game. This is done in two parts. In this ps4a, we only code to load and display the game structure and the components used. All the given components are used in creating the game structure.

Figure 7: Box image

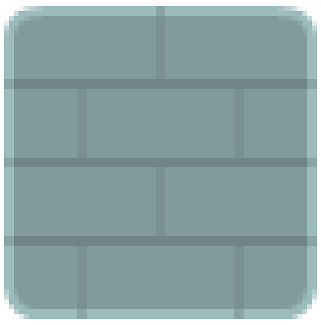
Figure 8: Brick image

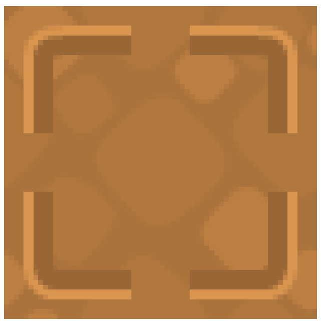
Figure 9: Destinantion of box

7.2 Key Algorithms, Data Structures, OO Design, Etc.

I could not complete the assignment completely. But I learned how to create a game in a different way of impementation. I could not use the required format for this assignment to complete the code. So I used my own format using many functions and loops.

7.3 Codebase

Makefile

```
CC = g++
CFLAGS = -Wall -Werror -std=c++11 -pedantic
BOOST_ARG = -lboost_unit_test_framework
OBJ = main.o Sokoban.o
EXE = Sokoban

all: $(EXE)

```

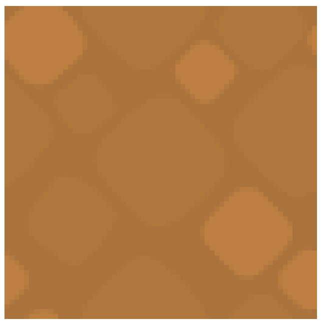
Figure 10: floor image

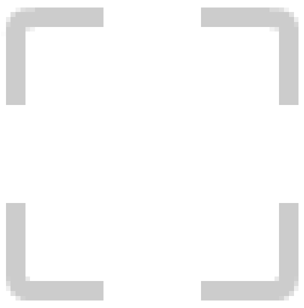
Figure 11: Environment

```
Sokoban: $(OBJ)
    $(CC) $(CFLAGS) -o $(EXE) $(OBJ) $(BOOST_ARG)

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

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

clean:
    rm $(EXE) *.o

```

Sokoban.cpp

```
// Copyright 2022 Tharuni
#include "Sokoban.h"
Sokoban::Sokoban() {
c_level = 0;
}
void Sokoban::input_level(string s) {
ifstream source(s.c_str());
char c;
if (source.fail())
cout << "error" << endl;
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
source.get(c);
map[i][j] = int(c) - 48;  //  NOLINT
}
source.ignore();
}
}
void Sokoban::load_sprites() {
switch (c_level) {
case 1:
player_t.loadFromFile("player.png", IntRect(0, 0, 32, 40));
player_s.setTexture(player_t);
brick_t.loadFromFile("brick.png");
```

Figure 12: Player image

```
brick_s.setTexture(brick_t);
box_t.loadFromFile("box.png");
box_s.setTexture(box_t);
destination_t.loadFromFile("destination.png");
destination_s.setTexture(destination_t);
floor_t.loadFromFile("floor.png");
floor_s.setTexture(floor_t);
break;
}
}
void Sokoban::draw_map() {
no_destinations = 0;
window.clear(Color::Black);
for (int i = 0; i < size; i++)
for (int j = 0; j < size; j++)
if (map[i][j] == player) {
xplayer = i;
yplayer = j;
floor_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(floor_s);
map[i][j] = floor;
} else if (map[i][j] == brick) {
brick_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(brick_s);
} else if (map[i][j] == box) {
box_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(box_s);
} else if (map[i][j] == destination) {
no_destinations++;
destination_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(destination_s);
} else if (map[i][j] == floor) {
floor_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(floor_s);
}
player_s.setPosition(yplayer*tile_size + 60, xplayer*tile_size + 60);
window.draw(player_s);
window.display();
}
void Sokoban::run() {
window.create(VideoMode(600, 600), "Sokoban Game");
no_movements = 50;
change_level = false;
while (window.isOpen()) {
Event evnt;
while(window.pollEvent(evnt))
if ((evnt.type == evnt.Closed) || (evnt.key.code == Keyboard::Escape))
window.close();
```

```
if (!change_level) {
input_level(level[c_level]);
load_sprites();
draw_map();
change_level = true;
}
if (n != 100) {
draw_map();
}
}
}
```

Sokoban.h

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <string>
#include <fstream>
using namespace sf;
using namespace std;

class Sokoban
{
private:
    string level[5] = { "level 1.txt"};
    enum directions { left, right, up, down, previous, next };
    enum objects { player, brick, box, destination, floor, space };
    int xplayer, yplayer, no_destinations, no_movements, n, c_level = 0;
    const int size = 12, tile_size = 40;
    int map[12][12];
    Sprite player_s, brick_s, box_s, destination_s, floor_s;
    Texture player_t, brick_t, box_t, destination_t, floor_t;
    bool change_level;
    RenderWindow window;
public:
    Sokoban();
    void input_level(string);
    void load_sprites();
    void draw_map();
    void run();
};
```

main.cpp

```
// Copyright 2022 Tharuni
#include "Sokoban.h"
int main() {
Sokoban game;
game.run();
}
```

8 PS4b: Sokoban - Play a game

8.1 Assignment Description

This ps4b assignment is continuation of the ps4a assignment. In ps4a, we just loaded and displayed the game and its structure. But in ps4b we'll actually implement the gameplay elements using moveplayer() function.

8.2 Key Algorithms, Data Structures, OO Design, Etc.

Since it is the continuation of the previous assignment, I could not complete this assignment as well. But as I already mentioned in the previous discription I used my my own methodology to implement this assignment.

8.3 Codebase

Makefile

```
CC = g++
CFLAGS = -std=c++11 -c -g -Og -Wall -Werror -pedantic
OBJ = Sokoban.o main.o
DEPS = Sokoban.cpp Sokoban.h main.cpp
LIBS = -lsfml-graphics -lsfml-window -lsfml-system
EXE = Sokoban

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

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

lint:
    cpplint --filter=-runtime/references, --filter=-build/c++11 *.cpp

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

Sokoban.cpp

```
// Copyright 2022 Tharuni
#include "Sokoban.h"

Sokoban::Sokoban() {
c_level = 0;
}
void Sokoban::input_level(string s) {
ifstream source(s.c_str());
char c;
if (source.fail())
cout << "error" << endl;
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
source.get(c);
map[i][j] = int(c) - 48; //NOLINT
}
source.ignore();
}
}
void Sokoban::load_sprites() {
switch (c_level) {
case 0:
player_t.loadFromFile("player.png", IntRect(0, 0, 32, 40));
player_s.setTexture(player_t);
brick_t.loadFromFile("brick.png");
brick_s.setTexture(brick_t);
box_t.loadFromFile("box.png");
box_s.setTexture(box_t);
destination_t.loadFromFile("destination.png");
destination_s.setTexture(destination_t);
floor_t.loadFromFile("floor.png");
floor_s.setTexture(floor_t);
fnt.loadFromFile("font.otf");
txt.setFont(fnt);
txt2.setFont(fnt);
break;
}
}
void Sokoban::draw_map() {
no_destinations = 0;
window.clear(Color::Black);
for (int i = 0; i < size; i++)
for (int j = 0; j < size; j++)
if (map[i][j] == player) {
xplayer = i;
yplayer = j;
floor_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(floor_s);
map[i][j] = floor;
} else //NOLINT
if (map[i][j] == brick) {
brick_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(brick_s);
} else //NOLINT
if (map[i][j] == box) {
box_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(box_s);
} else //NOLINT
if (map[i][j] == destination) {
no_destinations++;
destination_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(destination_s);
} else //NOLINT
if (map[i][j] == floor) {
floor_s.setPosition(j*tile_size + 60, i*tile_size + 60);
window.draw(floor_s);
}
player_s.setPosition(yplayer*tile_size + 60, xplayer*tile_size + 60);
window.draw(player_s);
txt2.setFillColor(Color::White);
txt2.setCharacterSize(15);
txt2.setPosition(0, 500);
window.draw(txt2);
txt.setString(to_string(no_movements));
txt.setFillColor(Color::White);
txt.setCharacterSize(120);
window.draw(txt);
window.display();
}
int Sokoban::check_movement() {
while (c.getElapsedTime().asMilliseconds() < 500);// NOLINT
if (Keyboard::isKeyPressed(Keyboard::Left))
```

```
return left;
else
if (Keyboard::isKeyPressed(Keyboard::Right))
return right;
else
if (Keyboard::isKeyPressed(Keyboard::Up))
return up;
else
if (Keyboard::isKeyPressed(Keyboard::Down))
return down;
else
if (Keyboard::isKeyPressed(Keyboard::P))
return previous;
else
if (Keyboard::isKeyPressed(Keyboard::N))
return next;
else
if (Keyboard::isKeyPressed(Keyboard::Space)) {
c.restart();
window.close();
run();
} else //NOLINT
return 100;
return 0;
}
void Sokoban::update(int direction) {
int newx = 0, newy = 0;
switch (direction) {
case left:
newx = 0;
newy = -1;
break;
case right:
newx = 0;
newy = 1;
break;
case up:
newx = -1;
newy = 0;
break;
case down:
newx = 1;
newy = 0;
break;
case previous:
if (c_level >= 1) {
no_movements = 50;
c_level--;
change_level = false;
c.restart();
}
break;
case next:
if (c_level < 4) {
no_movements = 50;
c_level++;
change_level = false;
c.restart();
}
```

```
break;
}
if ((newx != 0) || (newy != 0)) {
if ((map[xplayer + newx][yplayer + newy] != brick) &&
(map[xplayer + newx][yplayer + newy] != box)) {
c.restart();
no_movements--;
xplayer = xplayer + newx;
yplayer = yplayer + newy;
player_s.setPosition(xplayer*tile_size + 60, yplayer*tile_size + 60);
window.draw(player_s);
}
else //NOLINT
if ((map[xplayer + newx][yplayer + newy] == box)
&& (map[xplayer + newx * 2][yplayer + newy * 2] != brick)
&& (map[xplayer + newx * 2][yplayer + newy * 2] != box)
&& (map[xplayer + newx]
[yplayer + newy] != player)) {
c.restart();
no_movements--;
xplayer = xplayer + newx;
yplayer = yplayer + newy;
map[xplayer][yplayer] = floor;
player_s.setPosition(xplayer*tile_size + 60, yplayer*tile_size + 60);
window.draw(player_s);
map[xplayer + newx][yplayer + newy] = box;
}
}
}
void Sokoban::check_conditions() {
if (no_destinations == 0) {
window.clear(Color::Black);
txt.setString("WINNER");
txt.setCharacterSize(120);
txt.setPosition(0, 0);
window.draw(txt);
window.display();
}
if (no_movements == -1) {
window.clear(Color::Black);
txt.setString("LOSER");
txt.setCharacterSize(120);
txt.setPosition(0, 0);
window.draw(txt);
window.display();
}
}
void Sokoban::run() {
window.create(VideoMode(900, 900), "Sokoban Game");
no_movements = 50;
change_level = false;
while (window.isOpen()) {
Event evnt;
while(window.pollEvent(evnt))
if ((evnt.type == evnt.Closed) || (evnt.key.code == Keyboard::Escape))
window.close();
if (!change_level) {
input_level(level[c_level]);
load_sprites();
draw_map();
change_level = true;
}
n = check_movement();
if (n != 100) {
update(n);
draw_map();
check_conditions();
}
}
}
```

Sokoban.h

```
// Copyright 2022 Tharuni
#ifndef _THARUNI_SOKOBAN_
#include <SFML/Graphics.hpp> // NOLINT
#include <iostream>  // NOLINT
#include <string>    // NOLINT
#include <fstream>   // NOLINT
using namespace sf;
using namespace std;

class Sokoban {
private:
string level[5] = { "level 1.txt", "level 2.txt"};
enum directions { left, right, up, down, previous, next, restart };
enum objects { player, brick, box, destination, floor, space };
int xplayer, yplayer, no_destinations, no_movements, n, c_level = 0;
Clock c;
const int size = 12, tile_size = 40;
int map[12][12];
Sprite player_s, brick_s, box_s, destination_s, floor_s;
Texture player_t, brick_t, box_t, destination_t, floor_t;
Text txt, txt2;
Font fnt;
bool change_level;
RenderWindow window;
        public:
Sokoban();
void input_level(string);
void load_sprites();
void draw_map();
int check_movement();
void update(int);
void check_conditions();
void run();
};
#endif
```

9 PS5: DNA Sequence Alignment

9.1 Assignment Description

In this task, we had to analyze the program's space and time properties while it was running and compare two ASCII strings to determine how far apart they were edited. Using a dynamic programming methodology, we were required to create a program that would compute the best sequence alignment between two DNA sequences. After that, we could assess the space complexity of our program using a programming tool called Valgrind.

9.2 Key Algorithms, Data Structures, OO Design, Etc.

A matrice of integers is represented by a vector of vectors of type int in the ED (Edit Distance) class. Using the min() and penalty() functions, the function optDistance() would then fill the matrix from bottom to top with the bare minimum of three distances before returning the best distance between the two sequences. Following a reverse-order traversal of the filled-out matrix, the alignment() function would construct the output alignment string using the best route.

9.3 What I've learned

Along with the many uses of Edit Distance, I also learned about the advantages of dynamic programming. This is because the recursive solution for this assignment would have required more than 2 N recursive calls to compare two strings of length N, which would have resulted in a much higher space complexity. I also discovered how to utilize the Valgrind massif tool to efficiently monitor program memory utilization. This utility provides information on how much memory the heap consumed while running. I was able to estimate the run time and memory use of a bigger sample of strings of length N by using the doubling method.

9.4 Codebase

Makefile

```
CC= g++
CFLAGS= -g -Wall -Werror -std=c++0x -pedantic
SFLAGS= -lsfml-system

all:    EDistance

EDistance:  main.o EDistance.o
    $(CC) main.o EDistance.o -o EDistance $(SFLAGS)

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

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

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

EDistance.cpp

```
//  Copyright Tharuni  Donapati
#include "EDistance.h"
EDistance::EDistance() {
}
EDistance::EDistance(std::string string_one, std::string string_two) {
  _string_one = string_one;
  _string_two = string_two;
```

```cpp
}
EDistance::~EDistance() {
}
int EDistance::penalty(char a, char b) {
  if (a == b) {
    return 0;
  } else if (a != b) {
    return 1;
  }
  return -1;
}
int EDistance::min(int a, int b, int c) {
  if (a < b && a < c) {
    return a;
  } else if (b < a && b < c) {
    return b;
  } else if (c < a && c < b) {
    return c;
  }
  return a;
}
int EDistance::OptDistance() {
  int i, j;
  int N = _string_one.length();
  int M = _string_two.length();
  for (i = 0; i <= M; i++) {
    std::vector<int> tmp;
    _matrix.push_back(tmp);
    for (j = 0; j <= N; j++) {
      _matrix.at(i).push_back(0);
    }
  }
  for (i = 0; i <= M; i++) {
    _matrix[i][N] = 2 * (M - i);
  }
  for (j = 0; j <= N; j++) {
    _matrix[M][j] = 2 * (N - j);
  }
  for (i = M - 1; i >= 0; i--) {
    for (j = N - 1; j >= 0; j--) {
      int opt1 = _matrix[i+1][j+1] + penalty(_string_one[j], _string_two[i])
   ;
      int opt2 = _matrix[i+1][j] + 2;
      int opt3 = _matrix[i][j+1] + 2;
      _matrix[i][j] = min(opt1, opt2, opt3); //NOLINT
    }
  }

  return _matrix[0][0];
}
void EDistance::PrintMatrix() {
  std::cout << "\n\nPrinting out the Matrix for debug purposes: \n\n";
  std::vector< std::vector<int> >::iterator a;
  std::vector<int>::iterator b;

  for (a = _matrix.begin(); a != _matrix.end(); a++) {
    for (b = (*a).begin(); b != (*a).end(); b++) {
      std::cout << std::right << std::setw(3) << *b << " ";
    }
    std::cout << "\n";
  }
}

std::string EDistance::Alignment() {
  std::ostringstream return_string;
  int M = _string_two.length();
  int N = _string_one.length();
  int i = 0, j = 0;
  int pen, opt1, opt2, opt3;
  std::string ret_str;
  while (i < M || j < N) {
    try {
      pen =  penalty(_string_one[j], _string_two[i]);
      opt1 = _matrix.at(i+1).at(j+1) + pen;
    }
    catch(const std::out_of_range& error) {
      opt1 = -1;
    }
    try {
      opt2 = _matrix.at(i+1).at(j) + 2;
    }catch(const std::out_of_range& error) {
      opt2 = -1;
    }
    try {
      opt3 = _matrix.at(i).at(j+1) + 2;
    }catch(const std::out_of_range& error) {
      opt3 = -1;
    }
    if (_matrix[i][j] == opt1) {
      return_string << _string_one[j] << " "
      <<  _string_two[i] << " "  << pen << "\n";
      i++;
      j++;
    } else if (_matrix[i][j] == opt2) {
      return_string << "- " << _string_two[i] << " 2\n";
      i++;
    } else if (_matrix[i][j] == opt3) {
      return_string << _string_one[j] << " -" << " 2\n";
      j++;
    }
  }
  ret_str = return_string.str();
  return ret_str;
}
```

EDistance.h

```cpp
//  Copyright Tharuni Donapati
#ifndef EDISTANCE_H_
#define EDISTANCE_H_

#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdexcept>      // std::out_of_range
#include <vector>
#include <SFML/System.hpp>

class EDistance {
 public:
    EDistance();
    EDistance(std::string string_one, std::string string_two);
    ~EDistance();
    int penalty(char a, char b);
    int min(int a, int b, int c); //NOLINT
    int OptDistance();
    std::string Alignment();

    void PrintMatrix();
 private:
    std::string _string_one, _string_two;
    std::vector< std::vector<int> > _matrix;
};

#endif     //    EDISTANCE_H_
```

main.cpp

```
// Copyright Tharuni Donapati
#include "EDistance.h"

int main(int argc, const char* argv[]) {
  sf::Clock clock;
  sf::Time t;
  std::string string1, string2;
  std::cin >> string1 >> string2;
  EDistance ed_test(string1, string2);
  int distance = ed_test.OptDistance();
  std::string alignment = ed_test.Alignment();
  std::cout << "Edit distance = " << distance << "\n";
  std::cout << alignment;
  ed_test.PrintMatrix();
  std::cout << "\nEdit distance = " << distance;
  t = clock.getElapsedTime();
  std::cout << "\nExecution time is " << t.asSeconds() << " seconds \n";
  return 0;
}
```

10 PS6: Random Writer

10.1 Assignment Description

The Markov Model makes predictions about the following characters in a string of kgrams, or k-length words. The RandWriter class uses the input string and order k to map each kgram in the string to the character that comes after it and its frequency. This class may now create strings with any desired length using the previously seen kgrams. Based on the probability of the k-length strings that were kept in the hashmap, the string's characters are determined.

10.2 Key Algorithms, Data Structures, OO Design, Etc.

The order of the kgrams is also input into the constructor of the RandWriter along with a string. The hashmap is constructed and initialized. The hashmap that was created is used by the kRand function to produce a string of length n. The function merely loops over the character map (the nested, inner map) of a kgram and appends each character to the end of a string. Then it returns a character at random from this string. The freq function scans or traverses the string and returns the frequency of a kgram. The generate function simply calls the rand function to create a new string each time the specified output length is not met. The second overloaded function yields the frequency of a character, which is straightforward and simply looks it up in the hashmap.

10.3 What I've learned

I have learnt about the Markov model and how it works. Also learnt about the Hash Map.

10.4 Codebase

Makefile

```
CC = g++
CFLAGS = -g -Wall -Werror -std=c++0x -pedantic
SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
Boost = -lboost_unit_test_framework
all:    TextWriter test

TextWriter: TextWriter.o RandWriter.o
    $(CC) TextWriter.o RandWriter.o -o TextWriter
test:   test.o RandWriter.o
    $(CC) test.o RandWriter.o -o test $(Boost)
TextWriter.o:TextWriter.cpp RandWriter.h
    $(CC) -c TextWriter.cpp RandWriter.h $(CFLAGS)
RandWriter.o:RandWriter.cpp RandWriter.h
    $(CC) -c RandWriter.cpp  RandWriter.h $(CFLAGS)
test.o:test.cpp
    $(CC) -c test.cpp $(Boost)
clean:
    rm *.o
    rm *.gch
    rm TextWriter
    rm test
```

TextWriter.cpp

```
// Copyright Tharuni Donapati
#include <string>
#include "RandWriter.h"
int main(int argc, const char* argv[]) {
if (argc != 3) {
    std::cout << "Usage: ./TextGenerator (int K) (int T)\n";
    return 0;
```

```
  }
std::string str_k(argv[1]);
std::string str_t(argv[2]);//NOLINT
int k = std::stoi(str_k);
int t = std::stoi(str_t);
std::string input = "";//NOLINT
std::string current_txt = "";//NOLINT
while (std::cin >> current_txt) {
input += " " + current_txt;
current_txt = "";
}
std::cout << "ORIGINAL INPUT TEXT BELOW THIS LINE.\n\n";
for (int a = 0; a < t; a++) {
std::cout << input[a];
if (input[a] == '.' || input[a] == '!') {
std::cout << "\n";
}
}
std::string output_string = "";//NOLINT
RandWriter amazing(input, k);
output_string += "" + amazing.gen(input.substr(0, k), t);
std::cout << "\n\nFINAL OUTPUT TEXT BELOW THIS LINE.\n\n";
for (int a = 0; a < t; a++) {
std::cout << output_string[a];
if (output_string[a] == '.' || output_string[a] == '!') {
std::cout << "\n";
    }
  }
std::cout << "\n";
return 0;
}
```

RandWriter.h

```
// Copyright Tharuni Donapati
#ifndef RANDWRITER_H_
#define RANDWRITER_H_
#include <algorithm>
#include <iostream>
#include <map>
#include <string>
#include <stdexcept>


class RandWriter {
 public:
  RandWriter(std::string text, int k);
  int order();
  int freq(std::string kgram);
  int freq(std::string kgram, char c);
  char randk(std::string kgram);
  std::string gen(std::string kgram, int T);
  friend std::ostream& operator<< (std::ostream &out, RandWriter&mm);
 private:
  int _order;
  std::map <std::string, int> _kgrams;
  std::string _alphabet;
};
#endif   //  RANDWRITER_H_
```

RandWriter.cpp

```cpp
// Copyright Tharuni Donapati
#include "RandWriter.h"
#include <algorithm>
#include <map>
#include <string>
#include <stdexcept>
#include <vector>
#include <utility>
RandWriter::RandWriter(std::string text, int k) {
  _order = k;
  srand((int)time(NULL)); //NOLINT
  std::string text_circular = text;
  for (int a = 0; a < _order; a++) {
    text_circular.push_back(text[a]);
  }
  int text_len = text.length();
  char tmp;
  bool inAlpha = false;
  for (int i = 0; i < text_len; i++) {
    tmp = text.at(i);
    inAlpha = false;
    for (unsigned int y = 0; y < _alphabet.length(); y++) {
      if (_alphabet.at(y) == tmp) {
        inAlpha = true;
      }
    }
    if (!inAlpha) {
      _alphabet.push_back(tmp);
    }
  }
  std::sort(_alphabet.begin(), _alphabet.end());

  std::string tmp_str;
  int x, y;
  for (x = _order; x <= _order + 1; x++) {
    for (y = 0; y < text_len; y++) {
      tmp_str.clear();
      tmp_str = text_circular.substr(y, x);
      _kgrams.insert(std::pair<std::string, int>(tmp_str, 0));
    }
  }
  std::map<std::string, int>::iterator it;
  int count_tmp = 0;
  for (x = _order; x <= _order + 1; x++) {
    for (y = 0; y < text_len; y++) {
      tmp_str.clear();
      tmp_str = text_circular.substr(y, x);
      it = _kgrams.find(tmp_str);
      count_tmp = it->second;
      count_tmp++;
      _kgrams[tmp_str] = count_tmp;
    }
  }
}
int RandWriter::order() {
  return _order;
}
int RandWriter::freq(std::string kgram) {
  if (kgram.length() != (unsigned)_order) {
    throw
      std::runtime_error("Error - kgram not of length k.");
  }
  std::map<std::string, int>::iterator it;
  it = _kgrams.find(kgram);
  if (it == _kgrams.end()) {
    return 0;
  }
  return it->second;
}
int RandWriter::freq(std::string kgram, char c) {
  if (kgram.length() != (unsigned)_order) {
    throw
      std::runtime_error("Error - kgram not of length k.");
  }
  std::map<std::string, int>::iterator it;
  kgram.push_back(c);
  it = _kgrams.find(kgram);
  if (it == _kgrams.end()) {
    return 0;
  }
  return it->second;
}
char RandWriter::randk(std::string kgram) {
  if (kgram.length() != (unsigned)_order) {
    throw std::runtime_error("Error - kgram not of length k (randk)");
  }
  std::map<std::string, int>::iterator it;
  it = _kgrams.find(kgram);
  if (it == _kgrams.end()) {
    throw std::runtime_error("Error - Could not find the given kgram! (randk
   )");
  }
  int kgram_freq = freq(kgram);
  int random_value = rand() % kgram_freq; //NOLINT
  double test_freq = 0;
  double random_num = static_cast<double>(random_value) / kgram_freq;
  double last_values = 0;
  for (unsigned int a = 0; a < _alphabet.length(); a++) {
    test_freq =  static_cast<double>(freq(kgram, _alphabet[a])) / kgram_freq
   ;
    if (random_num < test_freq + last_values && test_freq != 0) {
      return _alphabet[a];
      }
last_values += test_freq;
}
return '-';
}
std::string RandWriter::gen(std::string kgram, int T) {
if (kgram.length() != (unsigned)_order) {
throw std::runtime_error("Error - kgram not of length k. (gen)");
}
std::string final_string = "";//NOLINT
char return_char;
final_string += "" + kgram;
for (unsigned int a = 0; a < (T - (unsigned)_order); a++) {
return_char = randk(final_string.substr(a, _order));
final_string.push_back(return_char);
}
```

```
return final_string;
}
std::ostream& operator<< (std::ostream &out, RandWriter&mm) {
out << "\n_Order: " << mm._order << "\n";
out << "Alphabet: "<< mm._alphabet << "\n";
out << "Kgrams map: \n\n";
std::map<std::string, int>::iterator it;
for (it = mm._kgrams.begin(); it != mm._kgrams.end(); it++) {
out << it->first << "\t" << it->second << "\n";
}
return out;
}
```

test.cpp

```
//  Copyright Tharuni Donapati
#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include "RandWriter.h"
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
using namespace std; //NOLINT
BOOST_AUTO_TEST_CASE(order0) {
BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 0));
  RandWriter mm("gagggagaggcgagaaa", 0);
  BOOST_REQUIRE(mm.order() == 0);
  BOOST_REQUIRE(mm.freq("") == 17);
  BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
  BOOST_REQUIRE(mm.freq("", 'g') == 9);
  BOOST_REQUIRE(mm.freq("", 'a') == 7);
  BOOST_REQUIRE(mm.freq("", 'c') == 1);
  BOOST_REQUIRE(mm.freq("", 'x') == 0);
}

BOOST_AUTO_TEST_CASE(order1) {
  BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 1));
  RandWriter mm("gagggagaggcgagaaa", 1);
  BOOST_REQUIRE(mm.order() == 1);
  BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
  BOOST_REQUIRE_THROW(mm.freq("xx"), std::runtime_error);
  BOOST_REQUIRE(mm.freq("a") == 7);
  BOOST_REQUIRE(mm.freq("g") == 9);
  BOOST_REQUIRE(mm.freq("c") == 1);
  BOOST_REQUIRE(mm.freq("a", 'a') == 2);
  BOOST_REQUIRE(mm.freq("a", 'c') == 0);
  BOOST_REQUIRE(mm.freq("a", 'g') == 5);
  BOOST_REQUIRE(mm.freq("c", 'a') == 0);
  BOOST_REQUIRE(mm.freq("c", 'c') == 0);
  BOOST_REQUIRE(mm.freq("c", 'g') == 1);
  BOOST_REQUIRE(mm.freq("g", 'a') == 5);
  BOOST_REQUIRE(mm.freq("g", 'c') == 1);
  BOOST_REQUIRE(mm.freq("g", 'g') == 3);
  BOOST_REQUIRE_NO_THROW(mm.randk("a"));
  BOOST_REQUIRE_NO_THROW(mm.randk("c"));
  BOOST_REQUIRE_NO_THROW(mm.randk("g"));
  BOOST_REQUIRE_THROW(mm.randk("x"), std::runtime_error);
  BOOST_REQUIRE_THROW(mm.randk("xx"), std::runtime_error);
}
```

```
BOOST_AUTO_TEST_CASE(order2) {
  BOOST_REQUIRE_NO_THROW(RandWriter("gagggagaggcgagaaa", 2));
  RandWriter mm("gagggagaggcgagaaa", 2);
  BOOST_REQUIRE(mm.order() == 2);
  BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
  BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
  BOOST_REQUIRE_NO_THROW(mm.freq("xx"));
  BOOST_REQUIRE_THROW(mm.freq("", 'g'), std::runtime_error);
  BOOST_REQUIRE_THROW(mm.freq("x", 'g'), std::runtime_error);
  BOOST_REQUIRE_THROW(mm.freq("xxx", 'g'), std::runtime_error);
  BOOST_REQUIRE(mm.freq("aa") == 2);
  BOOST_REQUIRE(mm.freq("aa", 'a') == 1);
  BOOST_REQUIRE(mm.freq("aa", 'c') == 0);
  BOOST_REQUIRE(mm.freq("aa", 'g') == 1);
  BOOST_REQUIRE(mm.freq("ag") == 5);
  BOOST_REQUIRE(mm.freq("ag", 'a') == 3);
  BOOST_REQUIRE(mm.freq("ag", 'c') == 0);
  BOOST_REQUIRE(mm.freq("ag", 'g') == 2);
  BOOST_REQUIRE(mm.freq("cg") == 1);
  BOOST_REQUIRE(mm.freq("cg", 'a') == 1);
  BOOST_REQUIRE(mm.freq("cg", 'c') == 0);
  BOOST_REQUIRE(mm.freq("cg", 'g') == 0);
  BOOST_REQUIRE(mm.freq("ga") == 5);
  BOOST_REQUIRE(mm.freq("ga", 'a') == 1);
  BOOST_REQUIRE(mm.freq("ga", 'c') == 0);
  BOOST_REQUIRE(mm.freq("ga", 'g') == 4);
  BOOST_REQUIRE(mm.freq("gc") == 1);
  BOOST_REQUIRE(mm.freq("gc", 'a') == 0);
  BOOST_REQUIRE(mm.freq("gc", 'c') == 0);
  BOOST_REQUIRE(mm.freq("gc", 'g') == 1);
  BOOST_REQUIRE(mm.freq("gg") == 3);
  BOOST_REQUIRE(mm.freq("gg", 'a') == 1);
  BOOST_REQUIRE(mm.freq("gg", 'c') == 1);
  BOOST_REQUIRE(mm.freq("gg", 'g') == 1);
}
```

11 PS7: Kronos

11.1 Assignment Description

We parse the Kronos Intouch time clock log with regular expressions to examine it. We also check the device boot up timing. In this section, we take the specified device[1-6]*underscore*intouch.log files and generate a file that contains the time, date, status, and line number of the boot to the.rpt file.

11.2 Key Algorithms, Data Structures, OO Design, Etc.

I began by writing two regular expressions and testing them with strings in main rather than the log file. Then I tested with.rpt to see whether I could output the device[1-6] intouch.log file. Following that, I double-checked that I could successfully read each line from the log file and utilized both of my regular expressions. So I set enhanced dates and times to calculate the elapsed time for startups that are profitable Finally, I produced the rpt outputs file.

11.3 What I've learned

I learnt implementing of the regex library <regex> More efficiently.

I've written the following regexes to parse the file:
boost::regex start("[(]log[.]c[.]166[)] server started"); and
boost::regex end("oejs[.]AbstractConnector:Started SelectChannelConnector");and
boost::regex end("oejs[.]AbstractConnector:Started SelectChannelConnector");
start("[(]log[.]c[.]166[)] server started"); for boost::regex
–When the server is launched, it looks for the precise phrase (log.c.166).
–This string is unique to boot startups only.
end("oejs[.]AbstractConnector:Started SelectChannelConnector"); for boost::regex
–checks the accuracy of a sentence. AbstractConnector:Began SelectChannelConnector.
–That string is specific to boot completions only.

11.4 Codebase

Makefile

```
CC = g++
CFLAGS = -g -Wall -Werror -std=c++0x -pedantic
SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
Boost = -lboost_regex -lboost_date_time
all:    ps7

ps7:    ps7.o
    $(CC) ps7.o -o ps7 $(Boost)

ps7.o: ps7.cpp
    $(CC) -c ps7.cpp $(CFLAGS)
clean:
    rm *.o
    rm ps7
```

ps7.cpp

```
// Copyright Tharuni Donapati
#include <iostream>
#include <cstdint>
#include <vector>
#include <memory>
#include <exception>
#include <stdexcept>
#include <fstream>
#include <string>
#include <boost/regex.hpp>
```

```cpp
#include "boost/date_time/gregorian/gregorian.hpp"
#include "boost/date_time/posix_time/posix_time.hpp"
using boost::gregorian::date;
using boost::gregorian::from_simple_string;
using boost::gregorian::date_period;
using boost::gregorian::date_duration;
using boost::posix_time::ptime;
using boost::posix_time::time_duration;
int main(int argc, char* argv[]) {
  std::ifstream kronus(argv[1]);
  std::string input_file = argv[1];
  int line = 0;
  int started = 0;
  date d1;
  date d2;
  ptime t1;
  ptime t2;
  std::ofstream output_file(input_file + ".rpt", std::ofstream::out);
  std::string current_line;
  boost::regex start("[(]log[.]c[.]166[)] server started");
  boost::regex end("oejs[.]AbstractConnector:Started SelectChannelConnector"
    ); // NOLINT
  if (kronus.is_open() && output_file.is_open()) {
    output_file << "Device Boot Report\n"
    << "InTouch log file: " << input_file << "\n\n";
    while (getline(kronus, current_line)) {
      ++line;
      if (boost::regex_search(current_line, start)) {  // startup
        if (started == 1) {  // startup found again before a completion
          output_file << "Startup failed \n\n";
          started = 0;
        }
          output_file << current_line
          << "at line: " << line << "\n";
          started = 1;
          std::string temp_date = current_line.substr(0, 10);  // to make
    date
          std::string temp_time = current_line.substr(11, 8);  // to make
    time
          d1 = from_simple_string(temp_date);
          ptime temp(d1, time_duration(stoi(temp_time.substr(0, 2)), stoi(
    temp_time.substr(3, 2)), stoi(temp_time.substr(6, 2)), 0)); // NOLINT
          t1 = temp;
        } else if (boost::regex_search(current_line, end)) {  // completion
        if (started == 1) {
          std::string temp_date = current_line.substr(0, 10);  // to make
    date
          std::string temp_time = current_line.substr(11, 8);  // to make
    time
          d2 = from_simple_string(temp_date);
          ptime temp(d2, time_duration(stoi(temp_time.substr(0, 2)), stoi(
    temp_time.substr(3, 2)), stoi(temp_time.substr(6, 2)), 0)); // NOLINT
          t2 = temp;
          std::string date_time = current_line.substr(0, 19);
          time_duration td = t2-t1;
          output_file << date_time << ": " << "Startup Success \n"
          << "Elasped Time: " << td.total_milliseconds() << " ms\n\n";
          started = 0;
        }
      }
    }
  }
  if (started == 1) {  // if last startup failed
    output_file << "Startup failed \n\n";
    started = 0;
  }
  kronus.close();
  output_file.close();
  return 0;
}
```