Kristina Russell
COMP IV Sec 201: Project Portfolio
Spring 2023

CONTENTS:

Time to complete: 7 hours

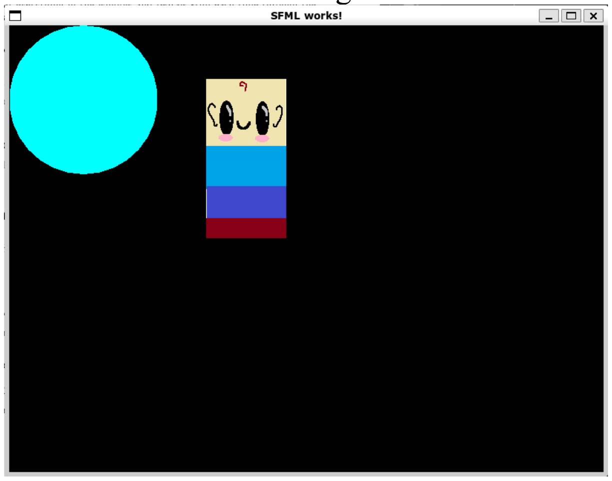
1. PS0 - Hello World with SFML

2. Summary:
   This assignment was an introduction to setting up and utilizing a graphics library, in this case the Simple and Fast Multimedia Library (SFML). First, students must make their own Ubuntu Linux setup function on their computer in order to be able to work with SFML. Personally, I chose the Windows Subsystem for Linux (WSL) with an Ubuntu shell as the environment I preferred to work in. After these basics had been set up, students followed the 2.4 tutorial on the SFML website in order to create a green circle in a pop up window. Students were also asked to create a sprite and add movement (via keystrokes), as well as another feature of their choice. I created a custom sprite, using WASD as the movement keys, and made the green circle cycle through colors of the rainbow with each keystroke pressed. After the assignment had been completed, students were also required to gzip tarball the directory that contained their code.

3. Algorithms/Data Structures/OO Designs Utilized:
   N/A

4. Knowledge Gained from the Assignment:
   For the first time, I was faced with being given documentation for a library I was totally unfamiliar with. This allowed me to develop an understanding of how to comb through documentation in order to make use of all the resources associated with a library. I also was interacting with and deciphering another programmer's code and documentation for the first time. Apart from using a library that was foreign to me, I also learned how to work with WSL in a way that made compiling my code simple and hassle free.

5. Evidence of Code Running:



6. Any Incomplete Portions of the Assignment:
   N/A

7. Source Code:

Makefile

```makefile
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system

.PHONY: all clean lint

all: sfml-app

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

sfml-app: main.o
	$(CC) $(CFLAGS) -o $@ $^ $(LIB)

clean:
	rm *.o sfml-app

lint:
	cpplint *.cpp *.hpp
```

Main.cpp

```cpp
#include <SFML/Graphics.hpp>

int main()
{
    sf::RenderWindow window(sf::VideoMode(800, 600), "SFML works!");
    sf::CircleShape shape(100.f);
    shape.setFillColor(sf::Color::Green);

    sf::Texture steveTex;
          if(!steveTex.loadFromFile("sprite.png"))
              exit(1);

          sf::Sprite steve;
          steve.setTexture(steveTex);

          sf::Vector2f pos;

          while (window.isOpen())
          {
              sf::Event event;
              while (window.pollEvent(event))
              {
                  if (event.type == sf::Event::Closed)
                      window.close();
                  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) {
                      pos = steve.getPosition(); pos.x--;
                      steve.setPosition(pos);
                      shape.setFillColor(sf::Color::Red);
                  }

                  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) {
                      pos = steve.getPosition(); pos.x++;
                      steve.setPosition(pos);
                      shape.setFillColor(sf::Color::Blue);
                  }

                  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) {
                      pos = steve.getPosition(); pos.y--;
                      steve.setPosition(pos);
                      shape.setFillColor(sf::Color::Cyan);
                  }

                  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) {
                      pos = steve.getPosition(); pos.y++;
                      steve.setPosition(pos);
                      shape.setFillColor(sf::Color::Green);
                  }

              }

              window.clear();
              window.draw(shape);
```

```
                window.draw(steve);
                window.display();
            }

            return 0;
        }
```

1. PS1 - Linear Feedback Shift Register and Image Encoding

2. Summary:

    The goal of this project was to create an image scrambler/encoder via the use of a linear feedback shift register (LFSR), more specifically a Fibonacci LFSR. The LFSR was used to generate an encryption/decryption key from a string that the user inputted. This key was 8 bits long, originating from that 16 bit long user inputted seed. The rgb values were then adjusted for each pixel using the key generated. This resulted in an image that was indistinguishable from the original. If the same key was used on the image again, the image would decrypt and the picture would look identical to the original.

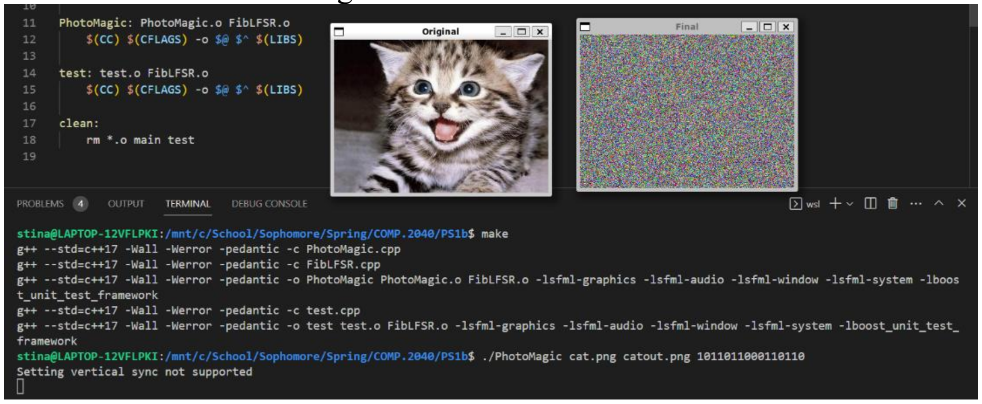
3. Algorithms/Data Structures/OO Designs Utilized:

    I had to use vectors in order to store and iterate through the seed/resulting key.

4. Knowledge Gained from the Assignment:

    I gained further knowledge on SFML and how to deal with displaying multiple outputs/windows. I was able to see a direct result of how my code was interacting with the original .png image, and how the encrypted image would change as a result. This was my first time working to encrypt and decrypt an image, so using operators such as XORs to achieve that was an intriguing experience.

5. Evidence of Code Running:



```
PhotoMagic: PhotoMagic.o FibLFSR.o
	$(CC) $(CFLAGS) -o $@ $^ $(LIBS)

test: test.o FibLFSR.o
	$(CC) $(CFLAGS) -o $@ $^ $(LIBS)

clean:
	rm *.o main test
```

```
stina@LAPTOP-12VFLPKI:/mnt/c/School/Sophomore/Spring/COMP.2040/PS1b$ make
g++ --std=c++17 -Wall -Werror -pedantic -c PhotoMagic.cpp
g++ --std=c++17 -Wall -Werror -pedantic -c FibLFSR.cpp
g++ --std=c++17 -Wall -Werror -pedantic -o PhotoMagic PhotoMagic.o FibLFSR.o -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboos
t_unit_test_framework
g++ --std=c++17 -Wall -Werror -pedantic -c test.cpp
g++ --std=c++17 -Wall -Werror -pedantic -o test test.o FibLFSR.o -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboost_unit_test_
framework
stina@LAPTOP-12VFLPKI:/mnt/c/School/Sophomore/Spring/COMP.2040/PS1b$ ./PhotoMagic cat.png catout.png 1011011000110110
Setting vertical sync not supported
```

6. Any Incomplete Portions of the Assignment:

The assignment required I created my own tests using the boost library, which I did not end up doing due to a time constraint.

7. Source Code:
   Makefile

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic
LIBS = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboost_unit_test_framework
DEPS = FibLFSR.hpp

all: PhotoMagic test

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

PhotoMagic: PhotoMagic.o FibLFSR.o
	$(CC) $(CFLAGS) -o $@ $^ $(LIBS)

test: test.o FibLFSR.o
	$(CC) $(CFLAGS) -o $@ $^ $(LIBS)

clean:
	rm *.o PhotoMagic test
```

Photomagic.cpp

```
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "FibLFSR.hpp"

// Transforms image using FibLFSR
void transform(sf::Image& pic, FibLFSR* key){
    //pixel is a pixelimage.getPixel(x, y);
    sf::Color pixel;

          //index through the image to get the rgb values and
          sf::Vector2u sizeP = pic.getSize();
          for (int x = 0; x < (signed)sizeP.x; x++) {
              for (int y = 0; y < (signed)sizeP.y; y++) {
                  //choose the pixel
                  pixel = pic.getPixel(x, y);

                  //xoring r g and b with 8bit generated ints
                  pixel.r = pixel.r^key->generate(8);
                  pixel.g = pixel.g^key->generate(8);
                  pixel.b = pixel.b^key->generate(8);
                  pic.setPixel(x, y, pixel);
              }
          }
      }
      // Display an encrypted copy of the picture, using the LFSR to do the encryption

      int main (int argc, char* argv[]){
          //argv[1] input-file.png
          //argv[2] output-file.png
          //argv[3] initial LFSR seed

          //seed the object with the given string
          FibLFSR key(argv[3]);

          //check that image can be loaded
          sf::Image image;
          if (!image.loadFromFile(argv[1]))
              return -1;

          //generate original texture and sprite before encryption/decryption
          sf::Texture text1;
          text1.loadFromImage(image);
          sf::Sprite sp1;
          sp1.setTexture(text1);

          //encrypt/decrypt
          transform(image, &key);

          sf::Vector2u sizeP = image.getSize();
          sf::RenderWindow window1(sf::VideoMode(sizeP.x, sizeP.y), "Original");
          sf::RenderWindow window2(sf::VideoMode(sizeP.x, sizeP.y), "Final");

```

```
        sf::Texture text2;
        text2.loadFromImage(image);
        sf::Sprite sp2;
        sp2.setTexture(text2);


        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(sp1);
            window1.display();
            window2.clear();
            window2.draw(sp2);
            window2.display();
        }

        // fredm: saving a PNG segfaults for me, though it does properly
        //   write the file
        if (!image.saveToFile(argv[2]))
            return -1;

        return 0;

    }


    /*
    step -> gets xorResult (single int returned)
    generate -> xors entire seed string (decimal version of xored binary
result)

    each pxl in row
        get r g b
        red xor w 8 bit generated int
        green xor w 8 bit generated int
        blue xor w w 8 bit generated int
        output new color

    */
```

FibLFSR.hpp

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

class FibLFSR{
public:
    //constructor to create LFSR w given initial seed
    FibLFSR(std::string seed);

          //do 1 step and ret new bit as 0 or 1
          int step();
          //do k steps and ret a k-bit int
          int generate(int k);
          friend std::ostream& operator<<(std::ostream&, const FibLFSR& lfsr);
      private:
          //any fields you need
          std::vector<int> s;

      };
```

FibLFSR.cpp

```
   #include "FibLFSR.hpp"

//data type that sim op of 16 bt fib lfsr

//constructor to create LFSR w given initial seed
FibLFSR::FibLFSR(std::string seed){
  //make a substring of the seed (a single character) so we can use
  //stoi to then store it on the int vector
  for(size_t i=0; i < seed.size(); i++){
       s.push_back(std::stoi(seed.substr(i, 1)));
     }
   }

   //Sim 1 step and ret new bit as 0 or 1
   int FibLFSR::step(){
     int xorResult;
     //grab 0 (15) and 2 (13) then xor result to x
     //grab x and 3 (12) then xor result to x
     //grab x and 5 (10) then xor result to x (our new bit finally)
     xorResult = s[0] ^ s[2];
     xorResult = xorResult ^ s[3];
     xorResult = xorResult ^ s[5];

     //use a temp vector to get rid of first guy in slot 0 (15) (he dies) by iterating through s vector, slap that bad boi (xorResult) on the end
     std::vector<int> temp;
     std::vector<int>::iterator i = s.begin(); i++;
     for(;i != s.end(); i++){
       temp.push_back(*i);
     }
     temp.push_back(xorResult);
     s = temp;
     return xorResult;
   }

   //sim k steps and ret a k-bit int
   int FibLFSR::generate(int k){
     int kthBit = 0, bToD = 1;
     std::vector<int> temp;

     //call step k times to create the binary number
     for(int i = 0; i < k; i++) {
       temp.push_back(this->step());
     }

     //convert binary number to base 10 number by reverse iterating through temp
     std::vector<int>::reverse_iterator j = temp.rbegin();
     for(size_t i = 0; i < temp.size(); i++) {
       if(*j == 1) {
         kthBit += bToD;
       }
       bToD *= 2;
       j++;
     }
```

```

        return kthBit;
    }

    std::ostream& operator<<(std::ostream& out, const FibLFSR& lfsr){
        //iterate thru
        for(size_t i = 0; i < lfsr.s.size(); i++){
            out << lfsr.s[i];
        }
        return out;
    }
```

Test.cpp

```
// Copyright 2022
// By Dr. Rykalova
// Editted by Dr. Daly
// test.cpp for PS1a
// updated 5/12/2022

#include <iostream>
#include <string>

#include "FibLFSR.hpp"

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

BOOST_AUTO_TEST_CASE(testStepInstr1) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
}

BOOST_AUTO_TEST_CASE(testStepInstr2) {
  FibLFSR l2("1011011000110110");
  BOOST_REQUIRE_EQUAL(l2.generate(9), 51);
}
```

1. PS2 - Sokoban

2. Summary:

   This assignment was focused on the box mover puzzle game, also known as Sokoban. I was tasked with creating the UI and mechanics in separate portions of the assignment. To start off with the UI, I needed to make a way to display the game board and sprites from the level files I was supplied with. The level files were text files filled with an ascii version of the map in order to have a place to load the level design from. Once the level could be read in, it could then be stored in a vector of vectors for parsing and then SFML could be used to load texture .pngs to then be drawn on the screen in relation to the ascii characters on the ascii map. Like PS0, movement was then implemented, but also collision this time between the player, boxes, walls, and the "win" tiles.

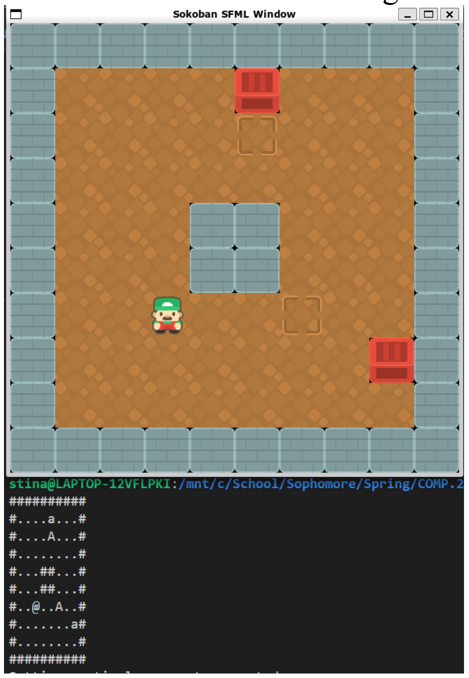
3. Algorithms/Data Structures/OO Designs Utilized:

   I used a vector of vector of chars in order to store a level map and another vector of vector of chars to store where the collide-ables sat. In using these I have essentially created matrices.

4. Knowledge Gained from the Assignment:

   Being able to read in from a file and break up the information I was being given into multiple matrices in order to layer the .pngs I was displaying was a new concept. Most of this project had to do with a lot of simple concepts weaved together. For example, for drawing a player who was pushing a box, I needed to work within the ascii map to find and then store the player location and how many boxes would be on the map. Once that was done, I had to look for the player ascii interacting with an ascii box and check that the box was not obstructed by a wall or another box. Then I could move on to trying to draw the sprites of each object out. I would move the box in the direction of the keyboard input, move the player, and overwrite the image where the player had previously sat.

5. Evidence of Code Running:



```
stina@LAPTOP-12VFLPKI:/mnt/c/School/Sophomore/Spring/COMP.2
##########
#....a...#
#....A...#
#........#
#...##...#
#...##...#
#..@..A..#
#.......a#
#........#
##########
```

6. Any Incomplete/Non-Working Portions of the Assignment:
I didn't include a lambda or a function call to the <algorithm> library. I couldn't seem to find a scenario where a lambda function would be useful in my code. My movement also wasn't as smooth as I would have liked it to be as the player moved quickly across the tiles of the board.

7. Source Code:

Makefile

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic
LIBS = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboost_unit_test_framework
DEPS = Sokoban.hpp

all: Sokoban

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

lint:
    cpplint *.cpp *.hpp

Sokoban: main.o Sokoban.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIBS)

clean:
    rm *.o Sokoban
```

Main.cpp

```
// Copyright [2023] <Kristina Russell>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "Sokoban.hpp"

int main(int argc, char* argv[]) {
    // argv[0] sokoban
    // argv[1] level?.txt
          Sokoban sok(argv[1]);
          std::cout << sok;
          sf::RenderWindow window(sf::VideoMode(
              sok.getHeight()*64, sok.getWidth()*64), "Sokoban SFML Window");

          // Start the game loop
          while (window.isOpen()) {
              // Process events
              sf::Event event;
              while (window.pollEvent(event)) {
                  // Close window: exit
                  if (event.type == sf::Event::Closed)
                      window.close();
              }
              // keyboard inputs
              if (sf::Keyboard::isKeyPressed(sf::Keyboard::W) ||
               sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) {
                  // up (0, -1)
                  sok.movePlayer(UP);
              } else if (sf::Keyboard::isKeyPressed(sf::Keyboard::A) ||
               sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) {
                  // left (-1, 0)
                  sok.movePlayer(LEFT);
              } else if (sf::Keyboard::isKeyPressed(sf::Keyboard::S) ||
               sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) {
                  // down (0, 1)
                  sok.movePlayer(DOWN);
              } else if (sf::Keyboard::isKeyPressed(sf::Keyboard::D) ||
               sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) {
                  // right (1, 0)
                  sok.movePlayer(RIGHT);
              } else if (sf::Keyboard::isKeyPressed(sf::Keyboard::R)) {
                  // reset!
                  window.clear();
                  sok.reset(argv[1]);
              }
              // Clear screen
              window.clear();
              // Draw the sprite
              window.draw(sok);
              // Update the window
              window.display();

              if (sok.isWon()) {
                  std::cout << "YOU WIN!!!" << std::endl;
                  return 0;
```

```
        }
    }
    return 0;
}
```

Sokoban.hpp

```
// Copyright [2023] <Kristina Russell>
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

      enum dir {UP, DOWN, LEFT, RIGHT};

      class Sokoban : public sf::Drawable {
       private:
          // display the sprite
          virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
const;

          sf::Sprite pSprite;
          sf::Texture pTexture;
          int height, width, pLocH, pLocW, numBoxes;

          // one matrix to store the lvl, the other to store collide/movea-
bles
          std::vector<std::vector<char>> level;
          std::vector<std::vector<char>> collidables;

       public:
          explicit Sokoban(std::string source);

          // changes the players location via being fed an enum
          void movePlayer(dir input);
          void reset(std::string source);

          bool const isWon();

          int getWidth(void);
          int getHeight(void);

          // overloading the insertion operator
          friend std::istream& operator>>(std::istream& in, Sokoban& soko);

          // overloading the extraction operator
          friend std::ostream& operator<<(std::ostream& out, Sokoban& soko);
      };
```

Sokoban.cpp

```cpp
// Copyright [2023] <Kristina Russell>
#include "Sokoban.hpp"

Sokoban::Sokoban(std::string source) {
    std::fstream file;
    file.open(source);
    if (!file) {
        throw std::runtime_error("Error level file could not be opened!");
    }
    file >> *this;
    file.close();
}

void Sokoban::reset(std::string source) {
    level.clear();
    collidables.clear();
    std::fstream file;
    file.open(source);
    if (!file) {
        throw std::runtime_error("Error level file could not be opened!");
    }
    file >> *this;
    file.close();
}

void Sokoban::draw(sf::RenderTarget& target, sf::RenderStates states) const {
    // loading textures once to be shared between sprites
    sf::Texture wall;
    if (!wall.loadFromFile("block_06.png")) {
        throw std::runtime_error("Error loading wall texture in draw()");
    }
    sf::Texture floor;
    if (!floor.loadFromFile("ground_01.png")) {
        throw std::runtime_error("Error loading floor texture in draw()");
    }
    sf::Texture area;
    if (!area.loadFromFile("ground_04.png")) {
        throw std::runtime_error("Error loading area texture in draw()");
    }
    sf::Texture crate;
    if (!crate.loadFromFile("crate_03.png")) {
        throw std::runtime_error("Error loading crate texture in draw()");
    }
    sf::Texture playerForward;
    if (!playerForward.loadFromFile("player_05.png")) {
        throw std::runtime_error("Error loading crate texture in draw()");
    }
    sf::Sprite sprite;
    int i = 0;
    while (level[i] != level.back()) {
        for (int j = 0; j < width; j++) {
            switch (level[i][j]) {
            case '#':
                sprite.setTexture(wall);
                sprite.setPosition(j*64, i*64);
                target.draw(sprite);
                break;
            case '.':
                sprite.setTexture(floor);
                sprite.setPosition(j*64, i*64);
                target.draw(sprite);
                break;
            case 'a':
                sprite.setTexture(area);
                sprite.setPosition(j*64, i*64);
                target.draw(sprite);
                break;
            default:
                break;
            }
            switch (collidables[i][j]) {
                case 'A':
                    sprite.setTexture(crate);
                    sprite.setPosition(j*64, i*64);
                    target.draw(sprite);
                    break;
                case '@':
                    sprite.setTexture(playerForward);
                    sprite.setPosition(j*64, i*64);
                    target.draw(sprite);
                    break;
            }
        }
        i++;
    }
}

void Sokoban::movePlayer(dir input) {
    int x, y;
    if (input == UP) {
        x = 0, y = -1;
    } else if (input == DOWN) {
        x = 0, y = 1;
    } else if (input == LEFT) {
        x = -1, y = 0;
    } else if (input == RIGHT) {
        x = 1, y = 0;
    }
    // checking infront of player for wall
    if (level[pLocH+y][pLocW+x] == '#') {
        return;
    } else if (collidables[pLocH+y][pLocW+x] == 'A') {
        // checking infront of player for crate
        // check for wall or crate in front of crate
        if (level[pLocH+(y*2)][pLocW+(x*2)] == '#' ||
         collidables[pLocH+(y*2)][pLocW+(x*2)] == 'A') {
            return;
```

```cpp
            }
            collidables[pLocH+(y*2)][pLocW+(x*2)] = 'A';
        }
        collidables[pLocH][pLocW] = '-';
        pLocH += y;
        pLocW += x;
        collidables[pLocH][pLocW] = '@';
    }

    bool const Sokoban::isWon() {
        int i = 0, numCorrect = 0;
        while (i != height) {
            for (int j = 0; j < width; j++) {
                if (level[i][j] == 'a' && collidables[i][j] == 'A') {
                    numCorrect++;
                }
            }
            i++;
        }
        if (numCorrect == numBoxes) {
            return 1;
        }
        return 0;
    }

    std::istream& operator>>(std::istream& in, Sokoban& soko) {
        int i = 0;
        std::string line;
        std::vector<char> temp1;
        std::vector<char> temp2;

        // get first line from txt file for width & height
        in >> line;
        soko.width = std::stoi(line);
        in >> line;
        soko.height = std::stoi(line);

        // read in the lvl txt file line by line, breaking it down to chars
        // i is the column, j is the row
        while (!in.eof()) {
            in >> line;
            for (int j = 0; j < soko.width; j++) {
                // store where the players inital location is & overwrite
                // the player char for storage in lvl
                if (line[j] == '@') {
                    soko.pLocH = i;
                    soko.pLocW = j;
                    temp2.push_back(line[j]);
                    line[j] = '.';
                } else if (line[j] == 'A') {
                    soko.numBoxes++;
                    temp2.push_back(line[j]);
                    line[j] = '.';
                }
                temp1.push_back(line[j]);
                temp2.push_back('-');
            }
            temp2.push_back('\n');
            soko.level.push_back(temp1);
            soko.collidables.push_back(temp2);
            temp1.clear();
            temp2.clear();
            line.clear();
            i++;
        }

        return in;
    }

    std::ostream& operator<<(std::ostream& out, Sokoban& soko) {
        int i = 0;
        while (i != soko.height) {
            for (int j = 0; j < soko.width; j++) {
                out << soko.level[i][j];
            }
            out << std::endl;
            i++;
        }
        out << std::endl << std::endl << std::endl;
        i = 0;
        while (i != soko.height) {
            for (int j = 0; j < soko.width; j++) {
                out << soko.collidables[i][j];
            }
            out << std::endl;
            i++;
        }
        return out;
    }

    int Sokoban::getWidth(void) {return width;}

    int Sokoban::getHeight(void) {return height;}

    /*  check for a wall being in the way
        if (level[pLocH+y][pLocW+x] == '#') {
            return;
        }
        // default floor
        else if (level[pLocH+y][pLocW+x] == '.') {
            level[pLocH][pLocW] = '.';
        }
        // crate
        else if (level[pLocH+y][pLocW+x] == 'A') {
            // check for wall in front of crate
            if (level[pLocH+(y*2)][pLocW+(x*2)] == '#') {
                return;
            }
            level[pLocH+(y*2)][pLocW+(x*2)] = 'A';
        }
        // win area
        else if (level[pLocH+y][pLocW+x] == 'a') {
            level[pLocH][pLocW] = 'a';
        }
    */
```

1. PS3 - Pythagorean Tree

2. Summary:

   For this assignment I was tasked with created a fractal, in this instance a Pythagorean tree. A fractal is an image that continues to repeat as you get smaller and smaller. Due to this fractal being a Pythagorean one, it was a tree that was started with a square, and with each iteration, two smaller child squares would spawn on top of each new parent square. Those children squares would then become parent squares, and the pattern would continue. The particular orientation of the child squares came from utilizing the Pythagorean theorem. The Pythagorean theorem is a way of calculating the sides of a right triangle by using the formula $a^2+b^2=c^2$. This allowed us to used the length of the parent square (c) to calculate the lengths of two identical child squares (a and b). Utilizing the equation and the relationships that accompany it, we could then draw the tree recursively.

   I did not end up accomplishing anything notable with this project. I never ended up submitting the code that I worked on, but I did find the project to be challenging in regards of thinking up some sort of solution. It seems to be a good project to teach how recursion can be utilized.

3. Algorithms/Data Structures/OO Designs Utilized:

   I don't believe that there would be any particular that I would've used from the algorithms library or any data structures.

4. Knowledge Gained from the Assignment:

   Recursion and fractals go hand in hand.

5. Evidence of Code Running:

   N/A

6. Any Incomplete Portions of the Assignment:

   Technically I did not complete any portion of this assignment. I had a trouble with the recursion and I couldn't figure out how to use SFML to draw the triangles on top of the parent square, with two smaller squares branching off the triangle at the right locations. I think it was a similar issue to one I had in the next project, PS4, where I was calculating the area to draw the shapes in the window incorrectly.

7. Source Code:

   N/A didn't submit

1. PS4 - Checkers

2. Summary:

   Here we were tasked with creating checkers using SFML. There were two portions, one where the UI would be designed, and the other where the mechanics would be implemented. For the UI portion, we simply needed to display an 8x8 tiled board with a black and red checkered pattern and the chips set up. The player should have been able to select a piece by clicking on it. To know it was selected, the tile underneath the piece would highlight a yellow color, but there was an issue when I programmed it where the pieces would not unselect. In order to create this checker board with all of its associated pieces, I created a Checkers class as instructed and had a vector of vectors of chars store the location of the board pieces starting location so that they could be drawn in correctly, and another vector of vector of chars to store the actual color tiling orientation of the board. It was similar to the approach I took for PS2 (Sokoban) in the way that the drawing of the chips and board had to be done in a certain order, and stored in separate locations so the chips could move around without losing the tiling underneath each piece.

3. Algorithms/Data Structures/OO Designs Utilized:

   I made use of multiple vector of vectors of chars in order to create matrices to store board data, highlight board data, the location of black chips, and the location of red chips.

4. Knowledge Gained from the Assignment:

   Although this assignment was not necessarily quite unlike Sokoban (PS2), I still believe some of the differences helped me become even more familiar with SFML. Requiring the pieces to be clickable left me having to troubleshoot and problem solve by reading documentation and rubber ducking my code much more than normal. It was quite a challenge to work with the upsizing from the ascii map I decided to use all while trying to make the area detection for clicking to select work.

5. Evidence of Code Running:

   N/A; As of last I worked on it, I hit a segmentation fault that I was not able to fix.

6. Any Incomplete Portions of the Assignment:

   I wasn't able to get the pieces to select correctly in the submission I provided. Later on, I was able to make the pieces select, but unselecting the piece when clicking on another chip did not work. Due to the inability to highlight the correct piece, I never progressed any further than the displaying the board portion. While trying to fix the highlight display, I progressed into having a segmentation fault, and thus have not been able to go any further since then.

7. Source Code:

Makefile

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic
LIBS = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboost_unit_test_framework
DEPS = Checkers.hpp

all: Checkers

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

      lint:
          cpplint *.cpp *.hpp

      Checkers: main.o checkers.o
          $(CC) $(CFLAGS) -o $@ $^ $(LIBS)

      clean:
          rm *.o Checkers
```

Main.cpp

```cpp
// Copyright [2023] <Kristina Russell>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "Checkers.hpp"

int main(int argc, char* argv[]) {
    Checkers curr;
    sf::RenderWindow window(sf::VideoMode(8*64, 8*64), "Checkers SFML Window");

          // run the program as long as the window is open
          while (window.isOpen()) {
              // check all the window's events that were triggered since the last iteration of the loop
              sf::Event event;
              while (window.pollEvent(event)) {
                  if (sf::Mouse::isButtonPressed(sf::Mouse::Left)) {
                      sf::Vector2i pos = sf::Mouse::getPosition(window);
                      // std::cout << pos.x << " " << pos.y << std::endl;
                      // std::cout << "click" << std::endl;
                      curr.clicked(pos);
                  }
                  // "close requested" event: we close the window
                  if (event.type == sf::Event::Closed)
                      window.close();
              }
              window.clear();
              // Draw the sprite
              window.draw(curr);
              // Update the window
              window.display();
          }

          return 0;
      }
```

Checkers.hpp

```
// Copyright [2023] <Kristina Russell>
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

class Checkers : public sf::Drawable {
 private:
  std::vector<std::vector<char>> board;
  std::vector<std::vector<char>> hBoard;
  std::vector<std::vector<char>> blackLoc;
  std::vector<std::vector<char>> redLoc;

  sf::Sprite pSprite;
  sf::Texture pTexture;
  virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
const;
 public:
    Checkers();
    void clicked(sf::Vector2i);
    friend std::istream& operator>>(std::istream& in, Checkers& check);
};
```

Checker.cpp

```cpp
// Copyright [2023] <Kristina Russell>
#include "checkers.hpp"

Checkers::Checkers() {
    std::fstream file;
    file.open("boardPiecesStart.txt");
    if (!file) {
        throw std::runtime_error("Error board file could not be opened!");
    }
    file >> *this;
    file.close();
}

std::istream& operator>>(std::istream& in, Checkers& check) {
    int i = 0;
    std::string line;
    std::vector<char> temp;

    for (int y = 0; y < 8; y++) {
       temp.push_back('-');
    }
    for (int y = 0; y < 8; y++) {
       check.blackLoc.push_back(temp);
       check.redLoc.push_back(temp);
    }
    temp.clear();
    temp.push_back('\n');
    check.blackLoc.push_back(temp);
    check.redLoc.push_back(temp);
    temp.clear();
    // read in the lvl txt file line by line, breaking it down to chars
    // i is the column, j is the row
    while (!in.eof()) {
        in >> line;
        for (int j = 0; j < 8; j++) {
            // store where the chips inital locations are & overwrite
            // the player char for storage in lvl
            if (line[j] == 'b') {
                check.blackLoc[i][j] = 'b';
                line[j] = ',';
            } else if (line[j] == 'r') {
                check.redLoc[i][j] = 'r';
                line[j] = ',';
            }
            temp.push_back(line[j]);
        }
        check.board.push_back(temp);
        temp.clear();
        line.clear();
        i++;
    }

    return in;
}
```

```cpp

void Checkers::draw(sf::RenderTarget& target, sf::RenderStates states) const {
    sf::Texture highlightTile;
    if (!highlightTile.loadFromFile("highlight.png")) {
        throw std::runtime_error("Error loading highlight texture in draw()");
    }
    sf::Texture redTile;
    if (!redTile.loadFromFile("redTile.png")) {
        throw std::runtime_error("Error loading red tile texture in draw()");
    }
    sf::Texture blackTile;
    if (!blackTile.loadFromFile("blackTile.png")) {
        throw std::runtime_error("Error loading black tile texture in draw()");
    }
    sf::Texture redPawn;
    if (!redPawn.loadFromFile("redpawn.png")) {
        throw std::runtime_error("Error loading red pawn texture in draw()");
    }
    sf::Texture redKing;
    if (!redKing.loadFromFile("redking.png")) {
        throw std::runtime_error("Error loading red king texture in draw()");
    }
    sf::Texture blackPawn;
    if (!blackPawn.loadFromFile("blackpawn.png")) {
        throw std::runtime_error("Error loading black pawn texture in draw()");
    }
    sf::Texture blackKing;
    if (!blackKing.loadFromFile("blackking.png")) {
        throw std::runtime_error("Error loading black king texture in draw()");
    }
    sf::Sprite sprite;
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            switch (board[i][j]) {
            case '.':
                sprite.setTexture(redTile);
                sprite.setPosition(j*64, i*64);
                target.draw(sprite);
                break;
            case ',':
                sprite.setTexture(blackTile);
                sprite.setPosition(j*64, i*64);
                target.draw(sprite);
                break;
            default:
                break;
            }
            switch (hBoard[i][j]) {
            case 'h':
                sprite.setTexture(highlightTile);
                sprite.setPosition(j*64, i*64);
```

```cpp
                    target.draw(sprite);
                    break;
                case 'x':
                    break;
                default:
                    break;
                }
                switch (redLoc[i][j]) {
                    case 'r':
                        sprite.setTexture(redPawn);
                        sprite.setPosition(j*64, i*64);
                        target.draw(sprite);
                        break;
                    case 'R':
                        sprite.setTexture(redKing);
                        sprite.setPosition(j*64, i*64);
                        target.draw(sprite);
                        break;
                    default:
                        break;
                }
                switch (blackLoc[i][j]) {
                    case 'b':
                        sprite.setTexture(blackPawn);
                        sprite.setPosition(j*64, i*64);
                        target.draw(sprite);
                        break;
                    case 'B':
                        sprite.setTexture(blackKing);
                        sprite.setPosition(j*64, i*64);
                        target.draw(sprite);
                        break;
                    default:
                        break;
                }
            }
        }
    }
    void Checkers::clicked(sf::Vector2i clickPos) {
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                if (blackLoc[i][j] == 'b' || blackLoc[i][j] == 'B' || redLoc[i][j] == 'r' || redLoc[i][j] == 'R') {
                    if (i*64 <= clickPos.y && ((i+1)*64) >= clickPos.y) {
                        // std::cout << "Y: " << i*64 << " " << clickPos.y << " " << (i+1)*64 << std::endl;
                        if (j*64 <= clickPos.x && ((j+1)*64) >= clickPos.x) {
                            if (hBoard[i][j] == 'h') {
                                hBoard[i][j] = 'x';
                            }
                            // std::cout << "X: " << j*64 << " " << clickPos.x << " " << (j+1)*64 << std::endl;
                            board[i][j] = 'h';
                        }
                    }
                }
            }
        }
```

```
}
```

1. PS5 - DNA Alignment

2. Summary:

   In this assignment we were faced with having to find the optimal sequence alignment of two strings. We were given text files that contained strings that were comprised of a combination of letters (A T G C) that are found in DNA strings. The problem lies in having to figure out which combination out of the thousands (possibly even more in some cases) of different sequences that we could construct would be the most "efficient." In order to decide what was the most efficient, the idea of an edit distance was introduced. The stipulations associated with the edit distance was that if a gap was needed to be inserted, that would increase the alignments score by 2 points, if two of the characters mismatch, it would increase by 1 point, and if the characters matched there would be no increase in points. The more points one alignment has than another, the less efficient it is, meaning that the alignment with the lowest score would be considered the most efficient. I used the Needleman-Wunsch method to determine the alignment. To put it simply, I created a matrix of the alignment scores, and then once that matrix was filled out, I traced back through the matrix to find the optimal distance, which results in the correct alignment string.

3. Algorithms/Data Structures/OO Designs Utilized:

   I made use of a vector of vectors of integers in order to store the scores for the different combination in a matrix. I used the Needleman-Wunsch algorithm as well.

4. Knowledge Gained from the Assignment:

   I learned of the Needleman-Wunsch algorithm and that it was possible to use the actual layout of a matrix in order to help optimize finding certain answers.

5. Evidence of Code Running:

Here is a test I ran on my code

Inputs:

string 1: aggggggggbeeeea
string 2: bgfq

Expected output:

```
a- 2 g- 2 g- 2 g- 2 g- 2 g- 2
g- 2 g- 2 g- 2 bb 0 e- 2 e- 2
eg 1 ef 1 aq 1
```

What happened:

```
a - 2
g - 2
g - 2
g - 2
g - 2
g - 2
g - 2
g - 2
```

```
g - 2
b b 0
e - 2
e - 2
e g 1
e f 1
a q 1

Execution time is 0.006495 seconds
Edit distance = 25
```

6. Any Incomplete Portions of the Assignment:

   I don't believe there were any portions that were excluded. I had a bit of trouble simply understanding the pdf the assignment was from, but after doing some research and taking my time to slowly read through everything, I was able to successfully do everything.

7. Source Code:

Makefile

```
// Copyright [2023] <Kristina Russell>
#include "EDistance.hpp"
#include <SFML/System.hpp>

int main(int argc, char** argv) {
    sf::Clock clock;
    sf::Time t;
    std::ifstream file;
    file.open(argv[1]);
          if (!file.is_open()) {
              std::cout << "File couldn't be found/opened!!!" << std::endl;
              exit(1);
          }

          std::string str1, str2;
          file >> str1;
          file >> str2;
          // extra dash so it's like the table in the example
          str1.insert(str1.begin(), '-');
          str2.insert(str2.begin(), '-');
          EDistance DNA(str1, str2);
          int x = DNA.optDistance();
          std::string y = DNA.alignment();
          std::cout << y << std::endl;
          t = clock.getElapsedTime();
          std::cout << "Execution time is " << t.asSeconds() << " seconds \n";
          std::cout << "Edit distance = " << x << std::endl;
      }
```

Main.cpp

```cpp
// Copyright [2023] <Kristina Russell>
#include "EDistance.hpp"
#include <SFML/System.hpp>

int main(int argc, char** argv) {
    sf::Clock clock;
    sf::Time t;
    std::ifstream file;
    file.open(argv[1]);
          if (!file.is_open()) {
              std::cout << "File couldn't be found/opened!!!" << std::endl;
              exit(1);
          }

          std::string str1, str2;
          file >> str1;
          file >> str2;
          // extra dash so it's like the table in the example
          str1.insert(str1.begin(), '-');
          str2.insert(str2.begin(), '-');
          EDistance DNA(str1, str2);
          int x = DNA.optDistance();
          std::string y = DNA.alignment();
          std::cout << y << std::endl;
          t = clock.getElapsedTime();
          std::cout << "Execution time is " << t.asSeconds() << " seconds
   \n";
          std::cout << "Edit distance = " << x << std::endl;
      }
```

EDistance.hpp

```
// Copyright [2023] <Kristina Russell>
#include <vector>
#include <iostream>
#include <string>
#include <fstream>
#include <algorithm>

class EDistance {
 public:
          EDistance(std::string x, std::string y);
          static int penalty(char a, char b);
          static int min(int a, int b, int c);
          int optDistance();
          std::string alignment();

          friend std::ostream& operator <<(std::ostream&, EDistance&);

       private:
          std::vector<std::vector<int>> matrix;
          std::string x;
          std::string y;
      };
```

EDistance.cpp

```cpp
// Copyright [2023] <Kristina Russell>
#include "EDistance.hpp"

EDistance::EDistance(std::string str1, std::string str2) {
    x = str1;
    y = str2;

    std::vector<int> temp;
    // size + 1 for the needleman-wunsch algo to work
          for (unsigned int i = 0; i < (x.size() + 1); i++) {
              temp.push_back(-1);
          }
          for (unsigned int i = 0; i < (y.size() + 1); i++) {
              matrix.push_back(temp);
          }
      }

      int EDistance::penalty(char a, char b) {
          return (a != b);
      }

      int EDistance::min(int a, int b, int c) {
          int x = std::min({a, b, c});
          return x;
      }

      int EDistance::optDistance() {
          // filling left most column / top row
          for (unsigned int i = 0; i < matrix.size(); i++)
              matrix[i][0] = i*2;

          for (unsigned int i = 0; i < matrix[0].size(); i++)
              matrix[0][i] = i*2;

          // making the similarity matrix
          int match = 0, insert = 0, del = 0;
          for (unsigned int i = 1; i < matrix.size(); i++) {
              for (unsigned int j = 1; j < matrix[0].size(); j++) {
                  match = matrix[i-1][j-1] + penalty(x[j], y[i]);
                  del = matrix[i-1][j] + 2;
                  insert = matrix[i][j-1] + 2;
                  matrix[i][j] = min(match, del, insert);
              }
          }

          // traceback to find edit distance
          int i = y.size();
          int j = x.size();
          int distance = 0;
          while (i > 0 || j > 0) {
              if ( (i > 0 && j > 0) && (matrix[i][j] == matrix[i-1][j-1] + penalty(x[j], y[i])) ) {
                  distance += penalty(x[j], y[i]);
```

```
                i--;
                j--;
            } else if (i > 0 && (matrix[i][j] == matrix[i-1][j] + 2)) {
                distance += 2;
                i--;
            } else {
                distance += 2;
                j--;
            }
        }

        // return the edit distance
        return distance;
    }

    std::string EDistance::alignment() {
        std::string alignment;
        int j = x.size();
        int i = y.size();
        // similar to the previous traceback
        // 2 = misaligned, which is also why a '-' is inserted
        while (i > 0 || j > 0) {
            // looks at the sizes of j (x size) and i (y size), making sure they're
            // greater than 0, as well as checking that the val of the matrix at i and j
            // is equal to the val of the matrix located at i-1 and j- 1 + the penalty(1 or 0)
            if ( (i > 0 && j > 0) &&
            (matrix[i][j] == matrix[i-1][j-1] + penalty(x[j], y[i])) ) {
                // need to convert to a string for storing
                std::string pen = std::to_string(penalty(x[j], y[i]));
                alignment.insert(alignment.begin(), pen[0]);
                alignment.insert(alignment.begin(), ' ');
                alignment.insert(alignment.begin(), y[i]);
                alignment.insert(alignment.begin(), ' ');
                alignment.insert(alignment.begin(), x[j]);
                alignment.insert(alignment.begin(), '\n');
                i--;
                j--;
            // make sure the val at matrix i, j is equal to the val at the location
            // of the matrix one vertically before ij if we add 2
            // checks for misalignment within string x
            } else if (i > 0 && (matrix[i][j] == matrix[i-1][j] + 2)) {
                alignment.insert(alignment.begin(), '2');
                alignment.insert(alignment.begin(), ' ');
                alignment.insert(alignment.begin(), y[i]);
                alignment.insert(alignment.begin(), ' ');
                alignment.insert(alignment.begin(), '-');
                alignment.insert(alignment.begin(), '\n');
                i--;
            } else { // misaliognment in string y
                alignment.insert(alignment.begin(), '2');
                alignment.insert(alignment.begin(), ' ');
                alignment.insert(alignment.begin(), '-');
                alignment.insert(alignment.begin(), ' ');
                alignment.insert(alignment.begin(), x[j]);
                alignment.insert(alignment.begin(), '\n');
                j--;
            }
        }
        // used to prevent the extra character at the
        // end of the string from printing
        alignment[alignment.size() - 1] = '\0';
        return alignment;
    }

    std::ostream& operator <<(std::ostream& out, EDistance& dna) {
        for (unsigned int i = 0; i < dna.matrix.size(); i++) {
            for (unsigned int j = 0; j < dna.matrix[0].size(); j++) {
             out << dna.matrix[i][j];
            }
        }
        return out;
    }
```

1. PS6 - RandWriter

2. Summary:
   The goal of this assignment was to create a program that could be seeded and given a number of k-grams to generate text that varied on levels of sophistication. The way this type of program should work is through something known as a Markov model. This model helps determine what characters or group of characters is going to come next by splitting up the text being given to it as a seed by k-gram groups. A k-gram group is simply how many letters are taken together at a time to build a tree of probability for lack of a better term. The program takes the grouping of characters from the text, looks at what comes next, and stores how many times that particular grouping is following by that particular next set.
   Although I did not complete this assignment I assume it could've been completed utilizing maps to store the characters and how many times they occur, and then creating some sort of weighted selection based off of how likely things are to occur in order to help the generator pick words or characters that are more likely to follow each other.

3. Algorithms/Data Structures/OO Designs Utilized:
   Like I said previously, I did not complete this assignment, but I assume a map of maps of chars(or strings) and integers would have been of much use.

4. Knowledge Gained from the Assignment:
   I now know that it is possible to generate text based off of a Markov model and its intriguing to think about how many applications this method has. Being able to apply probability in my coding is an extremely useful facet, especially when I've learned to break it down using data structures such as maps or trees.

5. Evidence of Code Running:
   N/A

6. Any Incomplete Portions of the Assignment:
   I didn't end up submitting any code for this assignment.

7. Source Code:

Makefile

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic
LIBS = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
lboost_unit_test_framework
DEPS = randwriter.hpp

all: Checkers

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

lint:
	cpplint *.cpp *.hpp

Checkers: main.o randwriter.o
	$(CC) $(CFLAGS) -o $@ $^ $(LIBS)

clean:
	rm *.o Checkers
```

Main.cpp
n/a

Randwriter.cpp
n/a

Randwriter.hpp
n/a

1. PS7 - Kronos Log Parsing

2. Summary:

   Devices such as a Kronos InTouch have boot sequences with codes that signify when the boot process has started and when it has completed. These codes are stored in log files among thousands of other codes the system's device may have to initialize and run through on startup or as it runs its various processes. In this assignment, we were tasked with working with the log files a Kronos device produces in order to create a report style file that would contain all the areas from the log where the device started or completed its boot. The format of our report on the content of the log file was supposed to be done with each startup displaying, and whether it completed (and how much time it required to complete in milliseconds) or if it failed.

   In order to work with these extremely long log files (400,000+ lines), I had to utilize the regex library. Regex allows a user to parse through text and search for matching or similar text. In my particular case, I felt as though Regex provided enough flexibility that I did not need any sort of class, simply some functions to handle/utilize the Gregorian date and time boost libraries. I was able to open and search through the log files with a regex pattern string ("log.c.166" for startup, "oejs.AbstractConnector:StartedSelectChannelConnector" for completion). I set up my regex search to find lines containing either the startup or completion pattern, and to capture the date, time, and line number associated with each message and output it to the report in the correct format.

3. Algorithms/Data Structures/OO Designs Utilized:

   I used many boost libraries such as the regex and the date and time libraries, Gregorian and posix_time. The libraries present users such as myself with a vast array of classes and functions that can be utilized in almost every way a programmer could need (and if not, a programmer could always use these classes/functions as a basis and add their own twist). These premade classes allowed me to quickly feed the date and time into a preexisting function that could do the hassle of time conversion for me.

4. Knowledge Gained from the Assignment:

   I think this assignment went on to show that there were multiple ways to create a solution, whether it be heavily relying on libraries, or crafting something of your own to fulfill your needs. For much of this assignment, I found that I wasn't necessarily stuck on the "how" portion of crafting a solution, but more so overwhelmed with how many different tools I had at my disposal. With the formatting of regex, I could've taken many different paths in order to achieve the same end result.

5. Evidence of Code Running:

   Here is one of my output reports:

   `Device1_intouch.log.rpt`

```
=-=-=-=-= Device boot =-=-=-=-=
435369 (logs/device1_intouch.log): 2014-03-25 19:11:59 Boot Started
435759 (logs/device1_intouch.log): 2014-03-25 19:15:02.369 Boot Completed
SUCCESS: 183369 ms

=-=-=-=-= Device boot =-=-=-=-=
436500 (logs/device1_intouch.log): 2014-03-25 19:29:59 Boot Started
436859 (logs/device1_intouch.log): 2014-03-25 19:32:44.036 Boot Completed
SUCCESS: 165036 ms

=-=-=-=-= Device boot =-=-=-=-=
440719 (logs/device1_intouch.log): 2014-03-25 22:01:46 Boot Started
440791 (logs/device1_intouch.log): 2014-03-25 22:04:27.514 Boot Completed
SUCCESS: 161514 ms

=-=-=-=-= Device boot =-=-=-=-=
440866 (logs/device1_intouch.log): 2014-03-26 12:47:42 Boot Started
441216 (logs/device1_intouch.log): 2014-03-26 12:50:29.824 Boot Completed
SUCCESS: 167824 ms

=-=-=-=-= Device boot =-=-=-=-=
442094 (logs/device1_intouch.log): 2014-03-26 20:41:34 Boot Started
442432 (logs/device1_intouch.log): 2014-03-26 20:44:13.235 Boot Completed
SUCCESS: 159235 ms

=-=-=-=-= Device boot =-=-=-=-=
443073 (logs/device1_intouch.log): 2014-03-27 14:09:01 Boot Started
443411 (logs/device1_intouch.log): 2014-03-27 14:11:42.500 Boot Completed
SUCCESS: 161500 ms
```

6. Any Incomplete Portions of the Assignment:
   Although it was only bonus, I didn't end up searching for additional services start and completion sequences. I think it would have been fairly straightforward as I would have already had the structure for searching and finding the estimated time between start and completion set up.

7. Source Code:

Makefile

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic
LIBS = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboost_unit_test_framework -lboost_date_time
DEPS =

all: ps7

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

lint:
    cpplint *.cpp *.hpp

ps7: main.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIBS)

clean:
    rm *.o ps7
```

Main.cpp

```
#include <regex>
#include <string>
#include <fstream>
#include <iostream>
#include "boost/date_time/gregorian/gregorian.hpp"
#include "boost/date_time/posix_time/posix_time.hpp"
boost::posix_time::time_duration getElapsed (std::string dT1, std::string dT2){
    // take the date and time string, find the space so we can keep only the dates
    // std::cout << "REACHED w/ " << dT1 << " and " << dT2 << std::endl;
          boost::posix_time::ptime t1(boost::posix_time::time_from_string(dT1));
          boost::posix_time::ptime t2(boost::posix_time::time_from_string(dT2));
          boost::posix_time::time_duration duration = t2 - t1;
          return (duration);
      }

      int main(int argc, char* argv[]){
          std::ifstream inputLog;
          std::ofstream outputReport;

          // takes the log file name, gets rid of the log, replace with rpt
          std::string generatedFile = argv[1];
          for(int i = 0; i < 3; i++){
              generatedFile.pop_back();
          }
          generatedFile.append("rpt");
          inputLog.open(argv[1]);
          outputReport.open(generatedFile);

          if(!inputLog.is_open()){
              std::cout << "The log file " << argv[1] << " could not be found/open for reading!!!";
              return 1;
          }
          if(!outputReport.is_open()){
              std::cout << "The report file " << generatedFile << " could not be found/open for reading!!!";
              return 1;
          }
          // setting a lineNum int for incrementation with each line search of the file
          // a variable to store each line from the log file
          // smatches to store the results from the regex searches
          // regex to detect the date and time that comes before the start string
          // regex to detect the date and time that comes before the completed string
          int lineNum = 1;
          bool state = 1;
          std::string lineContents, dateTime1, dateTime2;
          std::smatch dT;
          std::regex started("(.*): [(]log.c.166[)] server started");
          std::regex completed("(.*):INFO:oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080");
```

```
        while(getline(inputLog, lineContents)){
            // a regex search for the start string; outputs failure if startups are back to back or hit eof without completion
            if(std::regex_search(lineContents, dT, started)){
                if(!state){
                    outputReport << "FAILURE TO BOOT" << std::endl << std::endl;
                }
                outputReport << "=-=-=-=-= Device boot =-=-=-=-=" << std::endl;
                outputReport << lineNum << " (" << argv[1] << "): " << dT[1] << " Boot Started" << std::endl;
                // std::cout << "  LOGGED" << std::endl;
                dateTime1 = dT[1];
                state = 0;
            }
            else if(std::regex_search(lineContents, dT, completed)){
                outputReport << lineNum << " (" << argv[1] << "): " << dT[1] << " Boot Completed" << std::endl;
                outputReport << "SUCCESS: ";
                dateTime2 = dT[1];
                boost::posix_time::time_duration yuh = getElapsed(dateTime1, dateTime2);
                outputReport << yuh.total_milliseconds() << " ms" << std::endl << std::endl;
                // std::cout << "  LOGGED" << std::endl;
                state = 1;
            }
            lineNum++;
        }

        // accounting for EOF
        if(!state){
            outputReport << "FAILURE TO BOOT" << std::endl << std::endl;
        }

        // std::cout << "RUH ROH RAGGY" << std::endl;

        inputLog.close();
        outputReport.close();

        return 0;

    }
```