Computing IV Sec 202: Project Portfolio

Camden Andersson

Spring 2024

Contents

Time to Complete Portfolio: 2 hours

1 PS0: Hello SFML

1.1 What I accomplished

The assignment aimed to familiarize me with the setup and use of a build environment for software development with the Simple and Fast Multimedia Library (SFML). The primary objective was to configure a development environment using GCC and Make, and to craft a simple SFML application that demonstrated the correct setup by displaying a basic graphical window. This window initially featured a simple green circle to ensure everything was functioning correctly.

In my solution, I built upon the basic requirements by adding interactive and dynamic elements to the application. I introduced a sprite to the SFML window, then I programmed this sprite to respond to user inputs, allowing it to move across the window based on keystroke commands. This enhancement not only tested my ability to manipulate graphical objects within SFML but also provided a practical demonstration of handling user interactions within a graphical application. The implementation involved using SFML's graphics and window management features to control the sprite's position dynamically and to update the display in real-time. This extension of the demo code allowed me to explore more of SFML's capabilities, particularly in terms of animation and input handling.

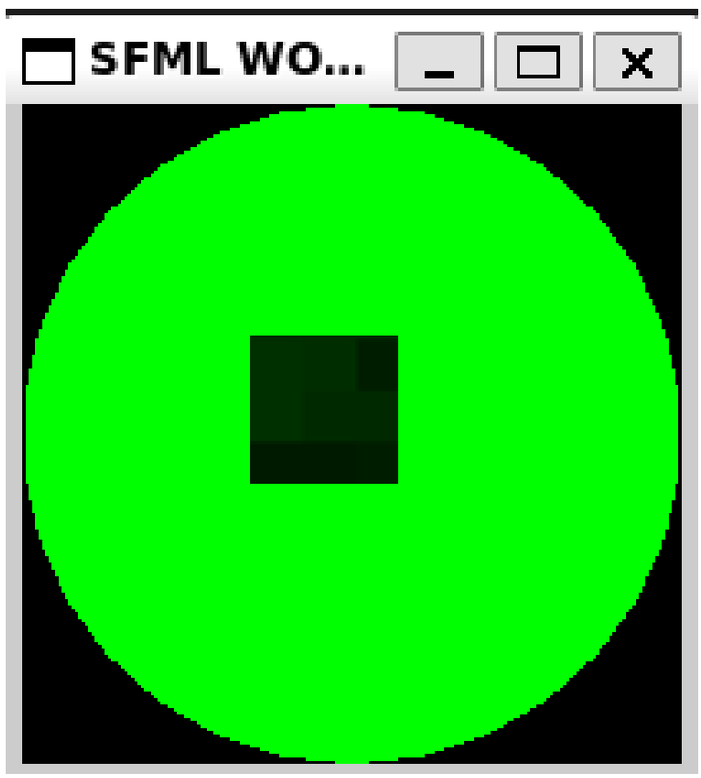

Figure 1: Window produced showcasing the sprite and circle

1.2 What I already knew

C++ itself was the only knowledge I had going in, as the project itself was about learning the very basics of SFML, which up to the point of taking the class I was completely unfamiliar with. Additionally, this was the first time since Computing II we had to use Makefiles, which was a large re-adjustment.

1.3 What I learned

I learned the very basics of SFML, such as creating a window loop and taking input from the keyboard. I also re-learned how to use and set up Makefiles for future projects, and what is expected of us in our code format form the use of CPPLINT.

1.4 Challenges

This was the first project, so naturally it was no challenge and was completed in minimal time.

1.5 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
# Your .hpp files
DEPS =
# Your compiled .o files
OBJECTS =
# The name of your program
PROGRAM = sfml-app

.PHONY: all clean lint


all: $(PROGRAM)

# Wildcard recipe to make .o files from corresponding .cpp file
%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

clean:
    rm *.o $(PROGRAM)

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

```
// Copyright Camden Andersson 2023

#include <SFML/Graphics.hpp>

int main() {
    sf::RenderWindow window(sf::VideoMode(200, 200), "SFML WORKS!");
    sf::CircleShape shape(100.f);
    shape.setFillColor(sf::Color::Green);

    sf::Texture texture;

    if (!texture.loadFromFile("sprite.png")) {
        std::exit(1);
    }

    sf::Sprite wood;
    wood.setTexture(texture);
    wood.setTextureRect(sf::IntRect(15, 15, 45, 45));

    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window.close();
```

```
        }

        window.clear();
        window.draw(shape);
        window.draw(wood);
        window.display();
    }


    return 0;
}
```

2 PS1: LFSR

2.1 What I accomplished

In PS1a: LFSR/PhotoMagic, you're tasked with developing a 16-bit Fibonacci LFSR simulation within the PhotoMagic namespace, focusing on bit-level operations in C++. This involves creating the FibLFSR class, which uses XOR operations across three "tap" positions to generate pseudo-random bit sequences and functions to step through the LFSR operations. The assignment requires implementing and testing these functionalities using the Boost test framework, culminating in a static library PhotoMagic.a that includes the LFSR implementation. The project also tests your skills in build process management using Makefiles and code validation through testing.

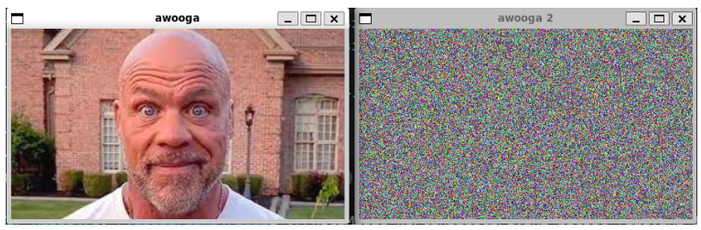

Figure 2: Windows produced from the encryption

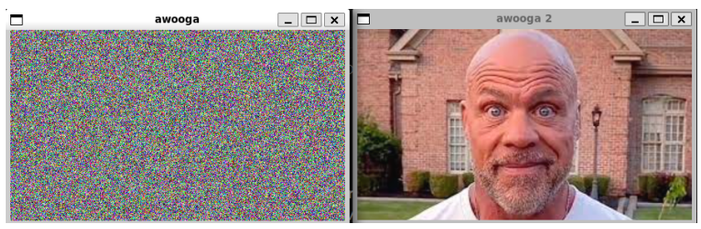

Figure 3: Windows produced from the decryption

My solution focused on the efficient and effective implementation of a 16-bit Fibonacci Linear Feedback Shift Register (LFSR) encapsulated within the FibLFSR class in the PhotoMagic namespace. Opting to represent the LFSR with a single integer, my approach made use of bitwise operations to manipulate bits directly, which is both computationally efficient and straightforward. This integer-based representation enabled simple implementation of essential operations such as shifts and XORs, which are vital for the LFSR's functionality.

The core functionality of my FibLFSR class includes a step function that performs one iteration of the LFSR sequence. In this function, specific bits at predefined "tap" positions are XORed together to generate a new bit, which is then used to update the LFSR's state, mimicking the feedback process found in Fibonacci LFSRs. Additionally, a generate function was implemented to produce a sequence of bits, useful for tasks requiring multiple bits, such as encrypting an image by altering pixel values. To facilitate debugging and visualization, I also overloaded the stream insertion operator to display the LFSR's current state.

This project not only demonstrated my capability to handle low-level bit manipulations in C++ but also reinforced my skills in using development tools like Makefiles for building processes and the Boost testing framework for validating the functionality of my code. This comprehensive approach provided a robust foundation for applying the LFSR to practical encryption tasks, showcased by encrypting images where the LFSR sequence influenced the RGB values of pixels, effectively altering the visual content.

2.2 What I already knew

I already knew the bitshift operations we were using (OR, XOR, AND), etc. These are vital to use in the step() function. The basics of SFML that were learned last project were put to use immediately in this project, as you need to display two windows showing the input and output of encrypting your image with your LFSR.

2.3 What I learned

I learned what a FibLFSR is, how to implement one, and what it can be used for. Additionally, I learned how to use the boost library to test your implementation to check my codes validity. This is a key skill that was learned during this assignment, showing its importance later on.

2.4 Challenges

Learning what a FibLFSR is. It took me a while to fully understand the concept of what it was, but after understanding it, the implementation wasn't so difficult. Learning and testing using the boost library was also somewhat challenging as well, as you need to think carefully about how to implement your tests to get them to make any sense. I would like to look more into the boost library at its use, while seeming daunting initially, showed the importance of testing your solution.

2.5 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
# Your .hpp files
DEPS = FibLFSR.hpp PhotoMagic.hpp
# Your compiled .o files
OBJECTS = FibLFSR.o PhotoMagic.o
# The name of your program
PROGRAM = PhotoMagic

SLIB = PhotoMagic.a

.PHONY: all clean lint

# Wildcard recipe to make .o files from corresponding .cpp file
#%.o: %.cpp $(DEPS)
#   $(CC) $(CFLAGS) -c $<

all: $(PROGRAM) PhotoMagic.a test

main.o: main.cpp FibLFSR.hpp PhotoMagic.hpp
    $(CC) $(CFLAGS) -c $< -o $@

PhotoMagic.o: PhotoMagic.cpp PhotoMagic.hpp
    $(CC) $(CFLAGS) -c $< -o $@

FibLFSR.o: FibLFSR.cpp FibLFSR.hpp
    $(CC) $(CFLAGS) -c $< -o $@

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

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

test.o: test.cpp FibLFSR.hpp
    $(CC) $(CFLAGS) -c $< -o $@

PhotoMagic.a: PhotoMagic.o FibLFSR.o
    ar rcs PhotoMagic.a FibLFSR.o PhotoMagic.o

```

```


clean:
    rm *.o *.a $(PROGRAM) test *.gch

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

```
// Copyright 2024 Camden Andersson
// pixels.cpp:
// using SFML to load a file, manipulate its pixels, write it to disk



// g++ -o pixels pixels.cpp -lsfml-graphics -lsfml-window
#include <iostream>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "PhotoMagic.hpp"
int main(int argc, char* argv[]) {
    sf::Image image;
    if (!image.loadFromFile(argv[1]))
        return -1;

    std::cout << "loaded image" << std::endl;

    sf::Vector2u size = image.getSize();
    sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "awooga");
    sf::RenderWindow window2(sf::VideoMode(size.x, size.y), "awooga 2");

    std::cout << "windows made" << std::endl;

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

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

    // transform image
    std::string seedSTR = argv[3];
    PhotoMagic::FibLFSR seed(seedSTR);
    PhotoMagic::FibLFSR* ptr = &seed;
    PhotoMagic::transform(image, ptr);
    std::cout << "seed loaded into transform" << std::endl;

    sf::Texture texture2;
    texture2.loadFromImage(image);

    sf::Sprite sprite2;
    sprite2.setTexture(texture2);

    std::cout << "texture loaded and sprites created, drawing window:" <<
   std::endl;

    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(sprite2);
        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;
}
```

```
// Copyright 2023 Camden Andersson
#ifndef PHOTOMAGIC_HPP
#define PHOTOMAGIC_HPP

#include "FibLFSR.hpp"

using PhotoMagic::FibLFSR;

namespace PhotoMagic {
// Transforms image using FibLFSR
void transform(sf::Image&, FibLFSR*);
// Display an encrypted copy of the picture, using the LFSR to do the
    encryption
}
#endif
```

```
// Copyright 2023 Camden Andersson
#include "PhotoMagic.hpp"

namespace PhotoMagic {
// Transforms image using FibLFSR
// take the iamge as an input, along with the fib lsfr.
// go pixel by pixel stepping once to transform each pixel

void transform(sf::Image& image, FibLFSR* lfsr) {
  int vert, hori;
  sf::Color pixel;

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

  vert = imageSize.x;
  hori = imageSize.y;
    // look at pdf for explanation on what to do
  for (int i = 0; i < hori; i++) {
      for (int j = 0; j < vert; j++) {
          pixel = image.getPixel(j, i);
```

```
            pixel.r = pixel.r ^ lfsr->generate(8);
            pixel.g = pixel.g ^ lfsr->generate(8);
            pixel.b = pixel.b ^ lfsr->generate(8);
            image.setPixel(j, i, pixel);
        }
    }
}
// Display an encrypted copy of the picture, using the LFSR to do the encryption
}  //  namespace PhotoMagic
```

```
// Copyright 2024 Camden Andersson
#ifndef FibLFSR_HPP
#define FibLFSR_HPP

#include <iostream>
#include <string>
#include <sstream>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

namespace PhotoMagic {
class FibLFSR {
 public:
        // Constructor to create LFSR with the given initial seed
        explicit FibLFSR(std::string seed);
        // Simulate one step and return the new bit as 0 or 1
        int step();
        // Simulate k steps and return a k-bit integer
        int generate(int k);
        // friend output operator
        friend std::ostream& operator<<(std::ostream& out, const FibLFSR& lfsr);
        // Transforms image using FibLFSR
        void transform(sf::Image&, FibLFSR*);
        // Display an encrypted copy of the picture, using the LFSR to do the encryption
 private:
        // Any fields that you need
        // stored lfsr as an int
        unsigned int storedLFSR = 0;
};
}  // namespace PhotoMagic
#endif
```

```
// Copyright 2024 Camden Andersson
#include "FibLFSR.hpp"



/*
Take the seed as an input of 16 bits
Xor bits 15, 13, 12 and 10 , output to bit 1
*/

namespace PhotoMagic {
FibLFSR::FibLFSR(std::string seed) {
    storedLFSR = std::stoi(seed, 0, 2);

```

```
    // for testing
    // std::cout << storedLFSR;
}

int FibLFSR::step() {
    int newBit = ((((((storedLFSR >> 15) & 1) ^ ((storedLFSR >> 13) & 1)))
    ^ ((storedLFSR >> 12) & 1)) ^ ((storedLFSR >> 10) & 1));
    storedLFSR = storedLFSR << 1;
    storedLFSR = storedLFSR | newBit;

    return newBit;
}

int FibLFSR::generate(int k) {
    int result = 0;

    for (int i = 0; i < k; ++i) {
        result = (result << 1) | step();
    }
    return result;
}

std::ostream& operator<<(std::ostream& out, const PhotoMagic::FibLFSR& lfsr)
    {
    for (int i = 15; i >= 0; --i) {
        out << (((lfsr.storedLFSR) >> i) & 1);
    }

    return out;
}
}  // namespace PhotoMagic
```

```
// Copyright 2022
// By Dr. Rykalova
// Editted by Dr. Daly
// test.cpp for PS1a
// updated 1/8/2024

#include <iostream>
#include <string>
#include <sstream>
#include "FibLFSR.hpp"


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

using PhotoMagic::FibLFSR;

BOOST_AUTO_TEST_CASE(testStepInstr) {
  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(testGenerateInstr) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE_EQUAL(l.generate(9), 51);
}

// new test cases

BOOST_AUTO_TEST_CASE(testOutput) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE_EQUAL(l.generate(9), 51);
  std::cout << l << std::endl;
}


BOOST_AUTO_TEST_CASE(testGenStep) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE_EQUAL(l.generate(6), 6);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
}

BOOST_AUTO_TEST_CASE(testGenSStream) {
  FibLFSR l("1011011000110110");
  std::stringstream ss;
  l.generate(8);
  ss << l;
  BOOST_REQUIRE_EQUAL(ss.str(), "0011011000011001");
}

BOOST_AUTO_TEST_CASE(testDifferentSeed) {
  FibLFSR l("1011010000000001");
  BOOST_REQUIRE_EQUAL(l.generate(6), 17);
}

BOOST_AUTO_TEST_CASE(testListStep) {
  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_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
}
```

3 PS2: Pythagoras Tree

3.1 What I accomplished

In PS2: Recursive Graphics (Pythagoras Tree), the assignment tasked me with creating a program that plots a Pythagoras tree, a fractal constructed from squares forming right triangles, illustrating the Pythagorean theorem. This mathematical model, named after Pythagoras, was invented by Albert E Bosman in 1942, and the assignment required implementing this using the SFML (Simple and Fast Multimedia Library). My program needed to take two command-line arguments: the length of one side of the base square (L) and the depth of the recursion (N).

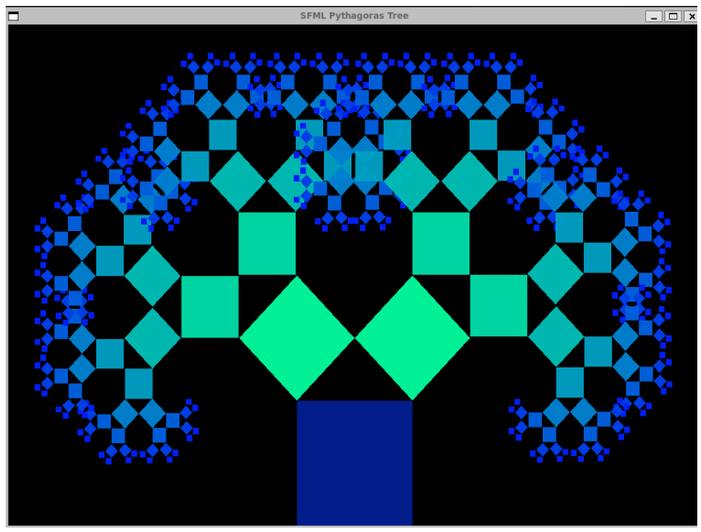

Figure 4: Window produced from running the program

In my solution, I focused on developing a recursive function to draw this fractal tree. I utilized the sf::Drawable class for drawing each part of the tree and handled the recursive drawing by calling a helper function from the main recursive function pTree(). For each recursion level, two smaller squares were drawn at right angles to each other, adhering to the mathematical principles outlined for the tree construction. I managed the geometry by calculating the side lengths of the child squares using trigonometric functions, specifically using cosine and sine based on a 45-degree angle.

I employed SFML's sf::RectangleShape (instead of sf::ConvexShape) to render squares and handled transformations such as rotation and scaling based on the depth of recursion to maintain the fractal's structure. Moreover, I ensured that the drawing's orientation and scaling were accurate by setting the shapes' origins appropriately, which was critical for the correct application of rotation transformations. The SFML window was sized dynamically based on the input parameter L to prevent the fractal from spilling over the window boundaries, optimizing the space usage according to the tree size. For extra credit, I incorporated multiple colors to differentiate between levels of the tree and allowed modifications of the inner triangle's angle to explore different fractal patterns. This extension provided a rich, visually engaging representation of the Pythagoras tree, showcasing my ability to integrate mathematical concepts with graphic programming in C++.

3.2 What I already knew

Before starting this assignment, I was already familiar with the mathematical concepts of sine and cosine functions, as its a crucial concept in geometry. This foundational knowledge was crucial for handling the rotations of squares in the project. Additionally, my prior experience with SFML from previous projects proved invaluable, as it directly influenced how graphical objects were manipulated and drawn in the application.

3.3 What I learned

Through this project, I deepened my understanding of applying the cos() and sin() functions to achieve rotation effects in graphical programming. Moreover, I learned how to use

recursion effectively to construct a Pythagoras Tree, which introduced me to the concept of plane fractals—a term and mathematical model I had not encountered before.

3.4 Challenges

The most significant challenge I faced was grasping the complexities of rotational transformations for each square in the fractal structure. To overcome this, I found it extremely helpful to sketch out the transformations (as directed in class), which clarified the relationships and positioning of the squares. This visualization was instrumental in adjusting my newLeft and newRight variables accurately, ensuring the fractal's correct creation.

3.5 Codebase

```makefile
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboost_unit_test_framework
# Your .hpp files
DEPS = PTree.hpp
# Your compiled .o files
OBJECTS = PTree.o
# The name of your program
PROGRAM = PTree

.PHONY: all clean lint

all: $(PROGRAM)

all: $(PROGRAM)

# Wildcard recipe to make .o files from corresponding .cpp file
%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

clean:
    rm *.o $(PROGRAM)

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

```cpp
// Copyright 2023 Camden Andersson
#include <string>
#include "PTree.hpp"

int main(int argc, char* argv[]) {
    int sideLength = std::atoi(argv[1]);
    int recursionAmt = std::atoi(argv[2]);

    sf::RenderWindow window(sf::VideoMode((sideLength * 6),
    (sideLength * 4)), "SFML Pythagoras Tree");

    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window.close();
```

```
        }
        PTree pTree;

        window.clear();
        pTree.pTree(window, (sideLength * 2.5), (sideLength * 3), sideLength, 0, recursionAmt);

        window.display();
    }



    return 0;
}
```

```
// Copyright 2023 Camden Andersson

#include <iostream>
#include <cmath>
#include <SFML/Graphics.hpp>

class PTree: public sf::RectangleShape {
 public:
    void pTree(sf::RenderTarget& window, int x, int y,
    int L, double angle, int depth);
};
```

```
// Copyright 2023 Camden Andersson
#include "PTree.hpp"

const double ANGLE = M_PI / 4.0;  // 45 degrees in radians

void PTree::pTree(sf::RenderTarget& window, int x, int y,
int L, double angle, int depth) {
    if (depth <= 0) {
        return;
    }

    sf::RectangleShape rect(sf::Vector2f(L, L));
    // rect.setFillColor(sf::Color(0,0,0,255));
    rect.setPosition(x, y);
    // rect.setSize(sf::Vector2f(L,L));
    rect.setRotation(angle * (180 / M_PI));
    //  rect.setOutlineColor(sf::Color::White);
    //  rect.setOutlineThickness(1);
    rect.setFillColor(sf::Color
    { 0xff9ff * (static_cast<sf::Uint32>(depth) * 2)});
    window.draw(rect);

    //  Look at the 4 points of the previous rect you drew
    sf::Vector2f vTL = rect.getTransform().transformPoint(rect.getPoint(0));
    sf::Vector2f vTR = rect.getTransform().transformPoint(rect.getPoint(1));
    //  sf::Vector2f vBR = rect.getTransform().transformPoint(rect.getPoint(2));
    //  sf::Vector2f vBL = rect.getTransform().transformPoint(rect.getPoint(3));
    //  int newRightX = vTL.x - (L / sqrt(2)) * cos(angle + (M_PI / 4));
    //  int newRightY = vTL.y - (L / sqrt(2)) * sin(angle + (M_PI / 4));

```

```
    //  pTree(window, newRightX, newRightY, L / sqrt(2), (angle - (M_PI / 4)
    ), depth - 1);

    int newLeftX = vTL.x - (L / sqrt(2)) * cos(angle + (M_PI / 4));
    int newLeftY = vTL.y - (L / sqrt(2)) * sin(angle + (M_PI / 4));

    int newRightX = vTR.x + (L) * sin(angle);
    int newRightY = vTR.y - (L) * cos(angle);

    pTree(window, newLeftX, newLeftY, L / sqrt(2),
    (angle - (M_PI / 4)), depth - 1);

    pTree(window, newRightX, newRightY, L / sqrt(2),
    (angle + (M_PI / 4)), depth - 1);
}
```

4 PS3: Sokoban

4.1 What I accomplished

In PS3b: Sokoban, the assignment was to enhance a basic Sokoban game by integrating gameplay mechanics that allow a player to move and interact within a predefined grid environment, using keyboard inputs (WASD or arrow keys). The gameplay involved moving the player character around the grid to push boxes into designated storage areas, with walls and other boxes potentially blocking movement. The game is won when all boxes are successfully placed in their respective storage areas. The task also included implementing functions to handle the game's winning condition, player movement, and game resetting.

Figure 5: Window produced from running the program

My solution for PS3b focused on expanding the Sokoban game's interactivity and playability. I implemented the movePlayer method in the Sokoban class, which accepted a direction input and moved the player accordingly, considering the presence of walls and the ability to push boxes into open spaces. This required careful handling of grid-based movement and collision detection to ensure the game mechanics worked as intended.

I also implemented the isWon method to check if all boxes were correctly placed in their storage locations, effectively determining the end of the game. This is also where my lambda function was located. To manage game resets, I incorporated functionality to revert the game to its initial state whenever the player pressed 'R', or to go back one move once you press 'U', which was particularly useful during development and testing for quickly retrying game scenarios.

Handling the player's movement involved calculating the potential new position based on the current direction and checking if the space was free, blocked by a wall, or had a box that could be pushed. If a box was in the player's path and could be moved (i.e., the space beyond the box was free), I updated the positions of both the player and the box. Otherwise, the move was blocked.

I also implemented robust collision detection, which helped to manage the various game states effectively. This included ensuring that the game accurately recognized win conditions and prevented illegal moves, such as pushing multiple boxes or moving into walls. Additionally, I had to carefully design the user interface to provide clear feedback on game actions, such as movement success or failure, and game reset or win scenarios. This was complemented by visual cues and messages to enhance the player's experience and engagement with the game.

4.2 What I already knew

Prior to starting Sokoban, I was already well-acquainted with the Sokoban game concept, having played similar games before. This familiarity significantly eased the implementation process, as I had a clear understanding of the game mechanics and objectives. Additionally, I was versed in handling data structures in both column and row major order, choosing to use column major order for this project due to its ease of use in the context of the game's grid-based logic.

4.3 What I learned

Throughout the development of Sokoban, I gained valuable insights into the complexities of game management and code maintainability. One of the key lessons was the importance of writing clean and concise code. Managing the intricacies of game logic alongside the graphical representation of the game underscored how crucial it is to maintain a well-organized codebase. This experience deepened my understanding of the interplay between game logic and its visual representation, shedding light on how these two aspects are integrated in real-time applications.

4.4 Challenges

The primary challenge in this project was managing and navigating my own code as I integrated more features. The complexity increased significantly with the addition of numerous nested if-else statements, which made the codebase cumbersome to modify and maintain. Although the project itself presented manageable difficulties, the real test was ensuring that the expanding code remained functional and accessible for future adjustments. This experience highlighted the necessity of adopting strategies to keep the codebase manageable despite its growing complexity

4.5 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
# Your .hpp files
DEPS = Sokoban.hpp
# Your compiled .o files
OBJECTS = Sokoban.o
# The name of your program
PROGRAM = Sokoban

.PHONY: all clean lint

all: $(PROGRAM) test Sokoban.a

# Wildcard recipe to make .o files from corresponding .cpp file
%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

Sokoban.a: Sokoban.o
    ar rcs Sokoban.a Sokoban.o

test: test.o Sokoban.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test.o: test.cpp Sokoban.hpp
```

```
    $(CC) $(CFLAGS) -c $< -o $@

clean:
    rm *.o $(PROGRAM) *.a

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

```
// Copyright 2024 Camden Andersson

#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Audio/Sound.hpp>
#include <SFML/Audio/SoundBuffer.hpp>
#include "Sokoban.hpp"

int main(int argc, char* argv[]) {
    // Create the sokoban object
    SB::Sokoban sokoban;
    sokoban.setLevelName(argv[1]);
    std::ifstream file;
    file.open(argv[1]);
    file >> sokoban;

    file.close();
    sf::SoundBuffer buffer;
    sf::Sound sound;
    buffer.loadFromFile("sound.wav");
    sound.setBuffer(buffer);

    // Play the game (draw the level for now)
    sf::RenderWindow window(sf::VideoMode(64 * sokoban.width(),
        64 * sokoban.height()), "Sokoban Game");
    bool soundPlayed = false;  // Variable to track if the sound has been played
    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window.close();

            window.clear();

            SB::Direction playerDirection;
            if (event.type == sf::Event::KeyReleased) {
                sf::Keyboard::Key key = event.key.code;
                std::cout << "Key released: " << event.key.code << std::endl;
                if (sokoban.isWon()) {
                    if (key == sf::Keyboard::R)
                        sokoban.resetLevel();
                } else {
                    switch (key) {
                    case sf::Keyboard::W:
                    case sf::Keyboard::Up:
                        playerDirection = SB::Up;
                        sokoban.movePlayer(playerDirection);
                        break;
                    case sf::Keyboard::A:
```

```
                    case sf::Keyboard::Left:
                        playerDirection = SB::Left;
                        sokoban.movePlayer(playerDirection);
                        break;
                    case sf::Keyboard::S:
                    case sf::Keyboard::Down:
                        playerDirection = SB::Down;
                        sokoban.movePlayer(playerDirection);
                        break;
                    case sf::Keyboard::D:
                    case sf::Keyboard::Right:
                        playerDirection = SB::Right;
                        sokoban.movePlayer(playerDirection);
                        break;
                    case sf::Keyboard::R:
                        sokoban.resetLevel();
                        break;
                    case sf::Keyboard::U:
                        sokoban.resetLevelBack();
                    default:
                        break;
                    }
                }
            }
            window.draw(sokoban);

            if (sokoban.isWon()) {
                if (!soundPlayed) {
                    sound.play();
                    soundPlayed = true;  // Mark the sound as played
                }
                // Set up the font
                sf::Font font;
                if (!font.loadFromFile("arial.ttf")) {
                    return 1;  // Exit if the font file is not found
                }
                // Create a text object
                sf::Text text("You Won!", font, 50);
                text.setFillColor(sf::Color::Green);
                // Center the text
                sf::FloatRect textRect = text.getLocalBounds();
                text.setOrigin(textRect.left + textRect.width / 2.0f,
                    textRect.top + textRect.height / 2.0f);
                text.setPosition(sf::Vector2f(window.getSize().x / 2.0f,
                window.getSize().y / 2.0f));

                // Draw the text
                window.draw(text);
            }
            // display window
            window.display();
        }
    }
}
```

```
// Copyright 2024 Camden Andersson
#include <fstream>
#include <string>
#include <iostream>
#include <vector>
```

```
#include <algorithm>
#include <SFML/Graphics.hpp>
namespace SB {
enum Direction {Up, Down, Left, Right};
class Sokoban : public sf::Drawable {
 public:
     Sokoban();
     //  void fillTextures();
     void setInitialPlayerLoc();
     int width(void) const;
     int height(void) const;
     sf::Vector2i playerLoc() const;
     void movePlayer(enum Direction _direction);
     bool isWon() const;
     void setPlayerX(int x);
     void setPlayerY(int y);
     void setLevelName(std::string _levelName);
     std::string getLevelName(void);
     char getGridData(int x, int y);
     char getGridOriginalData(int x, int y);
     void setGridData(int x, int y, char c);
     void resetLevel();
     void fillOriginalGrid();
     bool isBoxOrStoredBox(int x, int y);
     bool isWall(int x, int y);
     bool isOrigSpaceStorage(int x, int y);
     void fillPreviousGrid();
     void resetLevelBack();
     friend std::ifstream& operator>>(std::ifstream& is, SB::Sokoban& grid);

 protected:
        virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const;

 private:
     std::vector<std::vector<char>> gridData;  // The 2d grid
     std::vector<std::vector<char>> gridDataOriginal;  // unmodified grid to allow for a reset
     std::vector<std::vector<char>> gridDataPrevious;  // The 2d grid before we move
     std::string levelName;
     int internalWidth;
     int internalHeight;
     Direction lastDirection;
     int playerX;
     int playerY;
     sf::Texture texturePlayer05;
     sf::Texture texturePlayer08;
     sf::Texture texturePlayer17;
     sf::Texture texturePlayer20;
     sf::Texture textureBlock06;
     sf::Texture textureCrate03;
     sf::Texture textureEnviroment03;
     sf::Texture textureGround01;
     sf::Texture textureGround04;
};
}  // namespace SB
```

```
// Copyright 2024 Camden Andersson
#include "Sokoban.hpp"
```

```

namespace SB {
//  make it take in an ifstream and output an ifstream
std::ifstream& operator>>(std::ifstream& is, SB::Sokoban& grid) {
    // is.open(grid.levelName);
    if (!is.is_open()) {
        std::cerr << "Error opening file." << std::endl;
        return is;
    }

    // Read the width and height
    is >> grid.internalWidth >> grid.internalHeight;

    // Resize the grid
    grid.gridData.resize(grid.internalWidth, std::vector<char>(grid.internalHeight));

    // Read the grid data
    for (int i = 0; i < grid.internalWidth; ++i) {
        for (int j = 0; j < grid.internalHeight; ++j) {
            is >> grid.gridData[i][j];

            if (grid.gridData[i][j] == '@') {
                grid.playerX = j;
                grid.playerY = i;
            }
        }
    }

    //  Load in all the textures and set initial player loc
    //  grid.fillTextures();  // now you can use them in draw()
    grid.fillOriginalGrid();
    //  grid.setInitialPlayerLoc();
    return is;
}


void Sokoban::setLevelName(std::string _levelName) {
    levelName = _levelName;
}
// returns height due to how columns are read
int Sokoban::width(void) const {
    return internalHeight;
}
// returns width due to how columns are read
int Sokoban::height(void) const {
    return internalWidth;
}

sf::Vector2i Sokoban::playerLoc() const {
    // returns player position (x,y)
    return sf::Vector2i(playerX, playerY);
}

bool Sokoban::isWall(int x, int y) {
    if ((getGridData(x, y)) == '#')
        return true;
    else
        return false;
```

```
}


bool Sokoban::isBoxOrStoredBox(int x, int y) {
    if ((getGridData(x, y) == 'A') || (getGridData(x, y) == '1'))
        return true;
    else
        return false;
}

bool Sokoban::isOrigSpaceStorage(int x, int y)  {
        if (getGridOriginalData(x, y) == 'a')
            return true;
        else
            return false;
}

void Sokoban::fillPreviousGrid() {
        // Initialize and copy gridData to another vector
        gridDataPrevious.resize(gridData.size());
        for (size_t i = 0; i < gridData.size(); ++i) {
            gridDataPrevious[i].resize(gridData[i].size());
            for (size_t j = 0; j < gridData[i].size(); ++j) {
                gridDataPrevious[i][j] = gridData[i][j];
            }
        }
}


void Sokoban::resetLevelBack() {
        // Put the original grid data back to one previous step
        for (int i = 0; i < internalWidth; ++i) {
            for (int j = 0; j < internalHeight; ++j) {
                gridData[i][j] = gridDataPrevious[i][j];

                if (gridData[i][j] == '@') {
                    playerX = j;
                    playerY = i;
                }
            }
        }
    }


void Sokoban::movePlayer(enum Direction _direction) {
        std::cout << "Player Moved: " << std::endl;

        // This is where the player is currently at
        sf::Vector2i currLoc = playerLoc();

        // See if the move is possible
        bool bMoveOk = true;
        if ((_direction == Right) && (currLoc.x == internalHeight - 1))
            bMoveOk = false;
        else if ((_direction == Left) && (currLoc.x == 0))
            bMoveOk = false;
        else if ((_direction == Up) && (currLoc.y == 0))
            bMoveOk = false;
        else if ((_direction == Down) && (currLoc.y == internalWidth-1))
```

```
            bMoveOk = false;
        if (!bMoveOk) {
            std::cout << "Player move not possible." << std::endl;
            return;
        }

        // TO DO: make sure a move is possible ie a player can't go off the board
        int nextX = 0;
        int nextY = 0;
        int nextNextX = 0;
        int nextNextY = 0;

        // Before you move, store the current state so you cand undo one step with the Y key.
        fillPreviousGrid();
        // Make a move
        switch (_direction) {
        case Up:
        case Down:

            if (_direction == Up) {
                nextY = currLoc.y - 1;
                nextNextY = currLoc.y - 2;
            } else {
                nextY = currLoc.y + 1;
                nextNextY = currLoc.y + 2;
            }
            // If the nextY is not a wall or nothing, then move
            if (!isWall(currLoc.x, nextY)) {
                // If the next spot up is a box or stored box
                if (isBoxOrStoredBox(currLoc.x, nextY)) {
                    // See if the next next spot is wall or nothing
                    if (!isWall(currLoc.x, nextNextY)) {
                        //  adding check for if a box is after the box you are pushing
                        if (!isBoxOrStoredBox(currLoc.x, nextNextY)) {
                            // Move the player
                            setPlayerY(nextY);
                            setGridData(currLoc.x, nextY, '@');

                            // See if the box has moved into a storage spot
                            if (getGridData(currLoc.x, nextNextY) == 'a') {
                                // Move the box
                                setGridData(currLoc.x, nextNextY, '1');

                                // Replace the players old location with a '.' or 'a'
                                if (isOrigSpaceStorage(currLoc.x, currLoc.y))
                                setGridData(currLoc.x, currLoc.y, 'a');
                                else
                                    setGridData(currLoc.x, currLoc.y, '.');
                        } else {
                            if (!isWall(currLoc.x, nextNextY)) {
                                setGridData(currLoc.x, nextNextY, 'A');
                                setGridData(currLoc.x, currLoc.y, '.');
                            }
                        }
```

```
                        }
                    }
                } else {  // It's not a box, just move
                    // Set the new y location
                    setPlayerY(nextY);
                    setGridData(currLoc.x, nextY, '@');

                    // Replace the players old location with a .
                    if (isOrigSpaceStorage(currLoc.x, currLoc.y))
                        setGridData(currLoc.x, currLoc.y, 'a');
                    else
                        setGridData(currLoc.x, currLoc.y, '.');
                }
            }
            break;

        case Left:
        case Right:

            if (_direction == Left) {
                nextX = currLoc.x - 1;
                nextNextX = currLoc.x - 2;;
            } else {
                nextX = currLoc.x + 1;
                nextNextX = currLoc.x + 2;
            }
            if (!isWall(nextX, currLoc.y)) {
                // If the next spot left is a box or stored box
                if (isBoxOrStoredBox(nextX, currLoc.y)) {
                    // See if the next next spot is wall
                    if (!isWall(nextNextX, currLoc.y)) {
                        if (!isBoxOrStoredBox(nextNextX, currLoc.y)) {
                            // Move the player
                            setPlayerX(nextX);
                            setGridData(nextX, currLoc.y, '@');

                            // See if the box has moved into a storage spot
                            if (getGridData(nextNextX, currLoc.y) == 'a') {
                                // Move the box
                                setGridData(nextNextX, currLoc.y, '1');

                                // Replace the players old location with a '.' or 'a'
                                if (isOrigSpaceStorage(currLoc.x, currLoc.y))
                                    setGridData(currLoc.x, currLoc.y, 'a');
                                else
                                    setGridData(currLoc.x, currLoc.y, '.');
                        } else {
                            if (!isWall(nextNextX, currLoc.y)) {
                                setGridData(nextNextX, currLoc.y, 'A');

                                // Replace the players old location with a '.' or 'a'
                                if (isOrigSpaceStorage(currLoc.x, currLoc.y))
                                    setGridData(currLoc.x, currLoc.y, 'a');
                                else
                                    setGridData(currLoc.x, currLoc.y, '.');
```

```
                            }
                        }
                    }
                    }
                } else {  // It's not a box, just move
                    // Set the new y location
                    setPlayerX(nextX);
                    setGridData(nextX, currLoc.y, '@');

                    // Replace the players old location with a '.' or 'a'
                    if (isOrigSpaceStorage(currLoc.x, currLoc.y))
                        setGridData(currLoc.x, currLoc.y, 'a');
                    else
                        setGridData(currLoc.x, currLoc.y, '.');
                }
        }
            break;


        default:
            break;
        }
        lastDirection = _direction;
        std::cout << "Player is at: " << playerX << ", " << playerY << std::endl;
}

// uses algorithm and lambda functions
bool Sokoban::isWon() const {
        // When there are no more unstored boxes 'A', you won
        bool bIsWon = true;
        for (int i = 0; i < internalWidth; ++i) {
            for (int j = 0; j < internalHeight; ++j) {
                if (gridData[i][j] == 'a')
                    bIsWon = false;
            }
        }
        if (bIsWon) {
            return true;
        } else {
        auto it = std::find_if(gridData.begin(), gridData.end(), [](const auto& row) {
            return std::find(row.begin(), row.end(), 'A') != row.end();
            });
        return it == gridData.end();
    }
}


std::string Sokoban::getLevelName(void) {
    return levelName;
}

void Sokoban::setPlayerX(int x) {
    if (x >= 0)
        playerX = x;
}
    void Sokoban::setPlayerY(int y) {
    if (y >= 0)
```

```
        playerY = y;
}

void Sokoban::resetLevel() {
        // Put the original grid data back into the grid data
        // that will be drawn
        for (int i = 0; i < internalWidth; ++i) {
            for (int j = 0; j < internalHeight; ++j) {
                gridData[i][j] = gridDataOriginal[i][j];

                if (gridData[i][j] == '@') {
                    playerX = j;
                    playerY = i;
                }
            }
        }
}

void Sokoban::fillOriginalGrid() {
        // Initialize and copy gridData to another vector
        gridDataOriginal.resize(gridData.size());
        for (size_t i = 0; i < gridData.size(); ++i) {
            gridDataOriginal[i].resize(gridData[i].size());
            for (size_t j = 0; j < gridData[i].size(); ++j) {
                gridDataOriginal[i][j] = gridData[i][j];
            }
        }
}

Sokoban::Sokoban() {
    //  make sure all textures are valid
    texturePlayer05.loadFromFile("player_05.png");
    texturePlayer08.loadFromFile("player_08.png");
    texturePlayer17.loadFromFile("player_17.png");
    texturePlayer20.loadFromFile("player_20.png");
    textureBlock06.loadFromFile("block_06.png");
    textureCrate03.loadFromFile("crate_03.png");
    textureEnviroment03.loadFromFile("environment_03.png");
    textureGround01.loadFromFile("ground_01.png");
    textureGround04.loadFromFile("ground_04.png");
}

//  moved to constructor instead
//  void Sokoban::fillTextures() {
//  }

char Sokoban::getGridData(int x, int y) {
    return gridData[y][x];
}

char Sokoban::getGridOriginalData(int x, int y) {
    return gridDataOriginal[y][x];
}

void Sokoban::setGridData(int x, int y, char c) {
    gridData[y][x] = c;
}

void Sokoban::setInitialPlayerLoc() {
```

```
    char cCurr;
    for (int i = 0; i < internalWidth; ++i) {
        for (int j = 0; j < internalHeight; ++j) {
            cCurr = gridData[i][j];
            if (cCurr == '@') {
                    playerX = j;
                    playerY = i;
            }
        }
    }
}

void Sokoban::draw(sf::RenderTarget& target, sf::RenderStates states) const
    {
    sf::Sprite tPlayer;
    sf::Sprite tCrate;
    char cCurr;

    for (int i = 0; i < internalWidth; ++i) {
        for (int j = 0; j < internalHeight; ++j) {
            cCurr = gridData[i][j];
            sf::Sprite sprite;
            switch (cCurr) {
            case '#':
                sprite.setTexture(textureBlock06);
                sprite.setPosition(j * 64, i*64);
                target.draw(sprite, states);
                break;
            case '.':
                sprite.setTexture(textureGround01);
                sprite.setPosition(j * 64, i*64);
                target.draw(sprite, states);
                break;
            case '@':
                sprite.setTexture(textureGround01);
                sprite.setPosition(j * 64, i * 64);
                target.draw(sprite, states);
                switch (lastDirection) {
                case Up:
                    tPlayer.setTexture(texturePlayer08);
                    break;
                case Right:
                    tPlayer.setTexture(texturePlayer17);
                    break;
                case Left:
                    tPlayer.setTexture(texturePlayer20);
                    break;
                default:  // Down
                    tPlayer.setTexture(texturePlayer05);
                }
                tPlayer.setPosition(j * 64, i * 64);
                target.draw(tPlayer, states);
                break;
            case 'A':
                sprite.setTexture(textureGround01);
                sprite.setPosition(j * 64, i*64);
                target.draw(sprite, states);

                tCrate.setTexture(textureCrate03);
```

```
                tCrate.setPosition(j * 64, i*64);
                target.draw(tCrate, states);
                break;
            case 'a':
                sprite.setTexture(textureGround04);
                sprite.setPosition(j * 64, i*64);
                target.draw(sprite, states);
                break;
            case '1':
                sprite.setTexture(textureGround01);
                sprite.setPosition(j * 64, i*64);
                target.draw(sprite, states);

                tCrate.setTexture(textureCrate03);
                tCrate.setPosition(j * 64, i*64);
                target.draw(tCrate, states);
            }
        }
    }
}
}  // namespace SB
```

```
// Copyright 2024 Camden Andersson
#include <iostream>
#include <fstream>
#include "Sokoban.hpp"

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

using SB::Sokoban;

BOOST_AUTO_TEST_CASE(testInstantWin) {
  std::ifstream file;
  file.open("level1.lvl");
  SB::Sokoban sb;
  file >> sb;
  BOOST_REQUIRE_EQUAL(false, sb.isWon());
}

BOOST_AUTO_TEST_CASE(testCantMove) {
  std::ifstream file;
  file.open("level1.lvl");
  SB::Sokoban sb;
  file >> sb;
  sb.movePlayer(SB::Direction::Right);
  int startX = sb.playerLoc().x;
  int startY = sb.playerLoc().y;
  sb.movePlayer(SB::Direction::Up);
  int endX = sb.playerLoc().x;
  int endY = sb.playerLoc().y;
  BOOST_REQUIRE_EQUAL(startX, endX);
  BOOST_REQUIRE_EQUAL(startY, endY);
}

BOOST_AUTO_TEST_CASE(testBoxWall) {
  std::ifstream file;
  file.open("level3.lvl");
  SB::Sokoban sb;
```

```
  file >> sb;
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  int startX = sb.playerLoc().x;
  int startY = sb.playerLoc().y;
  sb.movePlayer(SB::Direction::Right);
  int endX = sb.playerLoc().x;
  int endY = sb.playerLoc().y;
  BOOST_REQUIRE_EQUAL(startX, endX);
  BOOST_REQUIRE_EQUAL(startY, endY);
}

BOOST_AUTO_TEST_CASE(testBoxBox) {
  std::ifstream file;
  file.open("level2.lvl");
  SB::Sokoban sb;
  file >> sb;
  int startX = sb.playerLoc().x;
  int startY = sb.playerLoc().y;
  sb.movePlayer(SB::Direction::Up);
  int endX = sb.playerLoc().x;
  int endY = sb.playerLoc().y;
  BOOST_REQUIRE_EQUAL(startX, endX);
  BOOST_REQUIRE_EQUAL(startY, endY);
}

BOOST_AUTO_TEST_CASE(testMoveOffScreen) {
  std::ifstream file;
  file.open("level4.lvl");
  SB::Sokoban sb;
  file >> sb;
  sb.movePlayer(SB::Direction::Down);
  sb.movePlayer(SB::Direction::Down);
  sb.movePlayer(SB::Direction::Down);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  int startX = sb.playerLoc().x;
  int startY = sb.playerLoc().y;
  sb.movePlayer(SB::Direction::Right);
  int endX = sb.playerLoc().x;
  int endY = sb.playerLoc().y;
  BOOST_REQUIRE_EQUAL(startX, endX);
  BOOST_REQUIRE_EQUAL(startY, endY);
}

BOOST_AUTO_TEST_CASE(testMoreBoxes) {
  std::ifstream file;
  file.open("level5.lvl");
  SB::Sokoban sb;
  file >> sb;
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Up);
```

```
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Down);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Up);
  BOOST_REQUIRE_EQUAL(true, sb.isWon());
}

BOOST_AUTO_TEST_CASE(testMoreStorage) {
  std::ifstream file;
  file.open("level6.lvl");
  SB::Sokoban sb;
  file >> sb;
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Right);
  sb.movePlayer(SB::Direction::Down);
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Up);
  sb.movePlayer(SB::Direction::Left);
  sb.movePlayer(SB::Direction::Left);
  sb.movePlayer(SB::Direction::Left);
  sb.movePlayer(SB::Direction::Left);
  sb.movePlayer(SB::Direction::Down);
  sb.movePlayer(SB::Direction::Left);
  sb.movePlayer(SB::Direction::Up);
  BOOST_REQUIRE_EQUAL(true, sb.isWon());
}
```

5 PS4: NBody

5.1 What I accomplished

For PS4: N-Body Simulation, the assignment involved developing a physics-based simulation to model the gravitational interactions among celestial bodies within a universe. This task extended the CelestialBody class to handle dynamic physics calculations, including updating velocities based on gravitational forces. A crucial component was the implementation of a step method in the Universe class, which progresses the simulation based on time increments and updates the positions and velocities of all celestial bodies accordingly.

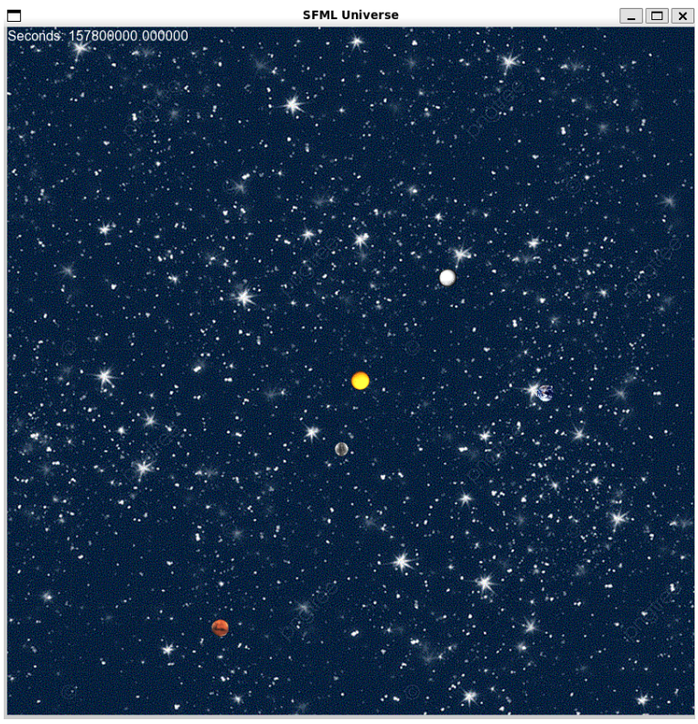

Figure 6: Window produced from running the program

My solution for PS4b was centered on accurately simulating the orbital mechanics of celestial bodies using Newton's laws of motion and gravitation. I extended the CelestialBody class to include methods for calculating gravitational forces exerted on each body, which involved computing pairwise forces and their components in both x and y directions. These calculations were based on the mass of the bodies and their distances from each other, employing the gravitational constant to determine the magnitude of the force.

In the Universe class, I implemented the step method to advance the simulation for a specified time step. This method calculated the net forces on each body, determined their accelerations, and updated their velocities and positions accordingly. The simulation employed a leapfrog integration scheme to ensure numerical stability and accuracy over time.

To manage memory efficiently and ensure robustness, I used smart pointers for handling the lifetimes of CelestialBody objects. This approach prevented memory leaks and ensured that objects were properly cleaned up, which was critical given the dynamic nature of the simulation where bodies could be added or removed based on various simulation conditions.

For extra credit, I designed my own simulation scenario, which featured a celestial body in an unusual orbit around the sun, demonstrating the flexibility and accuracy of the physics engine in handling non-standard orbital dynamics. Additionally, I integrated a time counter into the simulation, which displayed the elapsed time in seconds within the universe, enhancing the interactivity and user engagement by providing a real-time update of the simulation's progress.

One of the technical challenges was ensuring the accuracy of the simulation, particularly in maintaining the precision of calculations over long periods. This required careful implementation of the physics equations and attention to the cumulative errors that can occur in numerical simulations. The project not only deepened my understanding of astrophysical simulations but also enhanced my skills in managing complex object-oriented programming structures and memory management in a high-performance computing context.

5.2 What I already knew

Before stating on PS4, I already possessed a solid foundation in kinematics, thanks to my previous physics courses. This background was instrumental in formulating the simulation's dynamics, as it allowed me to effectively apply principles of motion and force to the celestial bodies in the virtual universe.

5.3 What I learned

During this project, I significantly deepened my understanding of how physics calculations integrate with graphical representations, building on the concepts explored in previous projects like PS3. This project highlighted the crucial link between the backend physics calculations and the frontend graphical display, which is fundamental in real-world applications such as games or simulation software. It provided me with a practical context for applying theoretical knowledge in a way that visually demonstrates complex concepts.

5.4 Challenges

The most significant challenge in this project was implementing the step() function, which involved coordinating the physics calculations with data input from various files to update the state of the universe accurately. While the integration of these elements was complex, the project was somewhat more straightforward compared to previous ones, such as PS3, because it involved fewer conditional logic blocks. The primary difficulty lay in ensuring that the physics behaved as expected across different scenarios and inputs. Additonally, I did not pass the testLeapFrog in the auto grader, as the decimal was not precise enough. I'm not entirely sure why that is, as I used doubles for all my internal calculations as requested, so it shouldn't have been an issue.

5.5 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
# Your .hpp files
DEPS = Universe.hpp CelestialBody.hpp
# Your compiled .o files
OBJECTS = Universe.o CelestialBody.o
# The name of your program
PROGRAM = NBody

.PHONY: all clean lint

all: $(PROGRAM) test NBody.a

# Wildcard recipe to make .o files from corresponding .cpp file
%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

NBody.a: Universe.o CelestialBody.o
    ar rcs NBody.a Universe.o CelestialBody.o

test: test.o Universe.o CelestialBody.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test.o: test.cpp Universe.hpp CelestialBody.hpp
    $(CC) $(CFLAGS) -c $< -o $@
```

```

clean:
    rm *.o $(PROGRAM) *.a test

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

```
// Copyright 2024 Camden Andersson

#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Audio/Sound.hpp>
#include <SFML/Audio/SoundBuffer.hpp>

#include "CelestialBody.hpp"
#include "Universe.hpp"


int main(int argc, char* argv[]) {
    // Load the file
    // std::ifstream file("kevin.txt");
    // if (!file.is_open()) {
    //    std::cerr << "Failed to open file" << std::endl;
    //    return 1;
    // }

    // Load an instance of the Universe class
    NB::Universe universe;
    std::cin >> universe;
    //  std::cout << universe;
    sf::Font textCounterFont;
    if (!textCounterFont.loadFromFile("arial.ttf")) {
        std::cout << "No arrial.ttf file found." << std::endl;
    }
    sf::Text textForCounter;
    textForCounter.setFont(textCounterFont);
    textForCounter.setCharacterSize(16);
    sf::Clock clock;



    // Set the screen size and load the Universe->CelestialBody textures
    int screenWidth = 800;
    int screenHeight = 800;
    universe.setScreenDimensions(screenWidth, screenHeight);
    universe.setMaxDimension();
    universe.loadTextures();

    // Setup the times
    double deltaT = std::stod(argv[2]);
    double T = std::stod(argv[1]);
    // double deltaT = 25000.00;
    // double T = 157788000.0;

    // NOTE: 1 day = 86400 sec
    //  1 year = 3153600 sec
    // double deltaT = 86400;
    // double T = 3153600

```

```
    // Initialize the current time
    double dCurrT = 0.0 + deltaT;
    std::string seconds = "Seconds: ";

    sf::SoundBuffer buffer;
    sf::Sound sound;
    buffer.loadFromFile("backgroundmusic.wav");
    sound.setBuffer(buffer);
    sound.play();
    sound.setLoop(true);

    // Create the window itself
    sf::RenderWindow window(sf::VideoMode(screenWidth, screenHeight), "SFML Universe");
    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed) {
                std::cout << universe;
                sound.stop();
                window.close();
            }
        }
        if (dCurrT < T) {
            // Do the physics to compute the forces and positions
            universe.step(deltaT);
            dCurrT = dCurrT + deltaT;
            // Display the results
            // temporary disabling for submission  std::cout << "time: " << dCurrT << std::endl;
        }
        textForCounter.setString(seconds + std::to_string(dCurrT));
        window.clear();
        window.draw(universe);
        window.draw(textForCounter);
        window.display();
    }

    return 0;
}
```

```
// Copyright 2024 Camden Andersson
#pragma once

#include "CelestialBody.hpp"
namespace NB {
class Universe : public sf::Drawable {
 private:
    std::vector<std::shared_ptr<CelestialBody>> bodies;
    int numParticles;
    double _radius;
    sf::Texture backgroundTexture;
    int screenWidth;
    int screenHeight;
 protected:
    virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const;  // override;
 public:
    Universe();
    explicit Universe(const std::string filename);
```

```
    void loadTextures();
    void setScreenDimensions(int width, int height);
    void setMaxDimension();
    void step(double deltaT);
    friend std::istream& operator>>(std::istream& is, Universe& universe);
    friend std::ostream& operator<<(std::ostream& os, const Universe& universe);
    CelestialBody& operator[](size_t index);
    double radius() const;
    int numPlanets() const;
};
}  // namespace NB
```

```
// Copyright 2024 Camden Andersson

#include "Universe.hpp"
namespace NB {
// Base Constructor
Universe::Universe() {
    numParticles = 0;
    _radius = 0.0;
}
// is this
Universe::Universe(const std::string filename) {
    std::ifstream file(filename);
    // same as the input operator
    file >> numParticles >> _radius;
    // bodies.clear();  // Clear any existing bodies
    // bodies.reserve(numParticles);  // Reserve space for the bodies
    // bodies.resize(numParticles);
    for (int i = 0; i < numParticles; ++i) {
        auto body = std::make_shared<CelestialBody>();
        file >> *body;  // Use the CelestialBody operator to read the body data
        bodies.push_back(body);
    }
}

void Universe::setScreenDimensions(int width, int height) {
    screenWidth = width;
    screenHeight = height;

    // Now set that dimension in all the CelestialBodies
    for (size_t i = 0; i < bodies.size(); i++) {
        bodies[i]->setScreenDimensions(width, height);
    }
}

void Universe::setMaxDimension() {
    double dMax = _radius;
    // Now set that dimension
    for (size_t i = 0; i < bodies.size(); i++) {
        bodies[i]->setMaxDimension(dMax);
    }
}

void Universe::loadTextures() {
    // EC: load the background texture
    if (!backgroundTexture.loadFromFile("background.gif")) {
        // Error handling if the file fails to load
```

```
        std::cerr << "Failed to load texture: background.gif " << std::endl;
    }
    backgroundTexture.setSmooth(true);  // Enable smooth resizing
    // Load the CelestialBody textures
    for (size_t i = 0; i < bodies.size(); i++) {
        bodies[i]->loadTextures();
    }
}

void Universe::draw(sf::RenderTarget& target, sf::RenderStates states) const {
    // Resize the texture and draw the background
    sf::Sprite sprite(backgroundTexture);
    sprite.setScale(screenWidth / backgroundTexture.getSize().x,
    screenWidth / backgroundTexture.getSize().y);
    sprite.setPosition(0.0f, 0.0f);
    target.draw(sprite, states);
    // Iterate through all the Celestial Bodies and draw
    for (const auto& body : bodies) {
        body->draw(target, states);
    }
}

std::istream& operator>>(std::istream& is, Universe& universe) {
    is >> universe.numParticles >> universe._radius;
    universe.bodies.clear();  // Clear any existing bodies
    universe.bodies.reserve(universe.numParticles);  // Reserve space for the bodies
    for (int i = 0; i < universe.numParticles; ++i) {
            auto body = std::make_shared<CelestialBody>();
            is >> *body;  // Use the CelestialBody operator to read the body data
            universe.bodies.push_back(body);
    }
    return is;
}

std::ostream& operator<<(std::ostream& os, const Universe& universe) {
    os << universe.numParticles << std::endl;  // removed all old formating
    os << universe._radius << std::endl;
    for (const auto& body : universe.bodies) {
        os << *body;
    }
    return os;
}

double Universe::radius() const {
    return _radius;
}

int Universe::numPlanets() const {
    return bodies.size();
}

NB::CelestialBody& Universe::operator[](size_t index) {
    if (index >= bodies.size()) {
        // Error handling if index is out of bounds
        std::cerr << "Failed to load Celestial body at " << index << "." << std::endl;
```

```
    }
    return *bodies[index];
}


void Universe::step(double deltaT) {
        double G = 6.67e-11;
        double deltaX, deltaY, r, F, Fx, Fy, ax, ay, vx, vy, px, py;
        // Go body by body
       for (size_t i = 0; i < bodies.size(); i++) {
            // For every body (except itself), use superpositon to compute F
            Fx = 0.0;
            Fy = 0.0;
            // For each body, compute the Force btwn that and the other spk
            // Of course, exclude the body you are on
            for (size_t j = 0; j < bodies.size(); j++) {
                if (j != i) {
                    deltaX = bodies[j]->position().x - bodies[i]->position().x;
                    deltaY = bodies[j]->position().y - bodies[i]->position().y;
                    r = pow(deltaX * deltaX + deltaY * deltaY, 0.5);
                    F = (G * bodies[i]->mass() * bodies[j]->mass()) / (r * r);

                    // Sum up the components with the previous force
                    Fx = Fx + (F * deltaX / r);
                    Fy = Fy + (F * deltaY / r);
                }
            }

            // Now that you have the total force...
            ax = Fx / bodies[i]->mass();
            ay = Fy / bodies[i]->mass();

            vx = bodies[i]->getVelocity().x + (deltaT * ax);
            vy = bodies[i]->getVelocity().y + (deltaT * ay);
            sf::Vector2f newV = sf::Vector2f(static_cast<float>(vx), static_cast<float>(vy));
            bodies[i]->setVelocity(newV);  // store the new velocity

            px = bodies[i]->position().x + (deltaT * vx);
            py = bodies[i]->position().y + (deltaT * vy);
            sf::Vector2f newPos = sf::Vector2f(static_cast<float>(px), static_cast<float>(py));
            bodies[i]->setPosition(newPos);  // store the new position
        }
    }

}  // namespace NB
```

```
// Copyright 2024 Camden Andersson
#pragma once

#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <memory>
#include <SFML/Graphics.hpp>
```

```

namespace NB {
class CelestialBody : public sf::Drawable {
 private:
    double xPos;
    double yPos;
    double xVelocity;
    double yVelocity;
    double _mass;
    std::string filename;
    sf::Texture texture;
    int screenWidth;
    int screenHeight;
    double maxDimension;

 public:
    CelestialBody() {}
    CelestialBody(double x, double y, double vx, double vy, double m, const std::string& file);
    void setPosition(sf::Vector2f pos);
    sf::Vector2f position() const;
    sf::Vector2f velocity() const;
    double mass() const;
    void loadTextures();
    void setScreenDimensions(int width, int height);
    void setMaxDimension(double _maxDimension);
    void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
    friend std::istream& operator>>(std::istream& is, CelestialBody& body);
    friend std::istream& operator>>(std::istream& is, std::shared_ptr<CelestialBody>& body);
    friend std::ostream& operator<<(std::ostream& os, const CelestialBody& body);
    void setVelocity(sf::Vector2f vel);
    sf::Vector2f getVelocity();
};
}  // namespace NB
```

```
// Copyright 2024 Camden Andersson
#include "CelestialBody.hpp"
namespace NB {

CelestialBody::CelestialBody(double x, double y, double vx,
    double vy, double m, const std::string& file)  :
    xPos(x), yPos(y), xVelocity(vx), yVelocity(vy), _mass(m), filename(file) {
    if (!texture.loadFromFile(filename)) {
        std::cerr << "Failed to load texture: " << filename << std::endl;
    }
}

void CelestialBody::setPosition(sf::Vector2f pos) {
    xPos = pos.x;
    yPos = pos.y;
}

void CelestialBody::setVelocity(sf::Vector2f vel) {
    xVelocity = vel.x;
    yVelocity = vel.y;
}
```

```

sf::Vector2f CelestialBody::getVelocity() {
    return sf::Vector2f(static_cast<float>(xVelocity), static_cast<float>(yVelocity));
}

sf::Vector2f CelestialBody::position() const {
    return sf::Vector2f(static_cast<float>(xPos), static_cast<float>(yPos));
}

sf::Vector2f CelestialBody::velocity() const {
    return sf::Vector2f(static_cast<float>(xVelocity), static_cast<float>(yVelocity));
}

double CelestialBody::mass() const {
    return _mass;
}

void CelestialBody::setScreenDimensions(int width, int height) {
    screenWidth = width;
    screenHeight = height;
}

void CelestialBody::setMaxDimension(double _maxDimension) {
    maxDimension = 2.2*_maxDimension;
    // make it a little bigger than 2x the largest distance
}

void CelestialBody::loadTextures() {
    if (!texture.loadFromFile(filename)) {
        std::cerr << "Failed to load texture: " << filename << std::endl;
    }
}

void CelestialBody::draw(sf::RenderTarget& target, sf::RenderStates states) const {
    sf::Sprite sprite;
    sprite.setTexture(texture);
    // Get drawing coordinates
    double drawX = xPos * (screenWidth / maxDimension) + (screenWidth / 2.0);
    double drawY = -1 * yPos * (screenWidth / maxDimension) + (screenHeight / 2.0);
    sprite.setPosition(static_cast<float>(drawX), static_cast<float>(drawY));
    target.draw(sprite, states);
}

std::istream& operator>>(std::istream& is, CelestialBody& body) {
    return is >> body.xPos >> body.yPos >> body.xVelocity >>
    body.yVelocity >> body._mass >> body.filename;
}

std::ostream& operator<<(std::ostream& os, const CelestialBody& body) {
    // removed all old formating
    os << body.xPos << " " << body.yPos << " "
       << body.xVelocity << " " << body.yVelocity << " "
       << body._mass << " "
```

```
        << body.filename << std::endl;
    return os;
}
}  // namespace NB
```

```
// Copyright 2024 Camden Andersson
#include <iostream>
#include <fstream>
#include "Universe.hpp"

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

using NB::Universe;

BOOST_AUTO_TEST_CASE(testNumPlanets) {
  NB::Universe un("planets.txt");
  BOOST_REQUIRE_EQUAL(un.numPlanets(), 5);
}

BOOST_AUTO_TEST_CASE(testRadius) {
  NB::Universe un("planets.txt");
  BOOST_REQUIRE_EQUAL(un.radius(), 2.50e+11);
}

BOOST_AUTO_TEST_CASE(testBracket0) {
  NB::Universe un("planets.txt");
  BOOST_REQUIRE_NO_THROW(un[0]);
}

BOOST_AUTO_TEST_CASE(testBracketEnd) {
  NB::Universe un("planets.txt");
  BOOST_REQUIRE_NO_THROW(un[4]);
}

BOOST_AUTO_TEST_CASE(testFormat) {
  NB::Universe un("planets.txt");
  std::stringstream ss;
  ss << un;
  std::string s = "5\n";
  s = s + "2.5e+11\n";
  s = s + "1.496e+11 0 0 29800 5.974e+24 earth.gif\n";
  s = s + "2.279e+11 0 0 24100 6.419e+23 mars.gif\n";
  s = s + "5.79e+10 0 0 47900 3.302e+23 mercury.gif\n";
  s = s + "0 0 0 0 1.989e+30 sun.gif\n";
  s = s + "1.082e+11 0 0 35000 4.869e+24 venus.gif\n";
  BOOST_REQUIRE_EQUAL(ss.str(), s);
}

BOOST_AUTO_TEST_CASE(testHardcoded) {
  NB::Universe un("planets.txt");
  // std::cout << un;
  std::ifstream ifs("test.txt");
  // declare your celestialBOdy objects for comparison.
  NB::CelestialBody a;
  NB::CelestialBody b;
  NB::CelestialBody c;
  NB::CelestialBody d;
  NB::CelestialBody e;
```

```
  // load in from sstream
  ifs >> a;
  ifs >> b;
  ifs >> c;
  ifs >> d;
  ifs >> e;
  // test the input for each celestialBody object.
  BOOST_REQUIRE_EQUAL(a.mass(), un[0].mass());
  BOOST_REQUIRE_EQUAL(a.position().x, un[0].position().x);
  BOOST_REQUIRE_EQUAL(a.position().y, un[0].position().y);
  BOOST_REQUIRE_EQUAL(a.velocity().x, un[0].velocity().x);
  BOOST_REQUIRE_EQUAL(a.velocity().y, un[0].velocity().y);
  BOOST_REQUIRE_EQUAL(b.mass(), un[1].mass());
  BOOST_REQUIRE_EQUAL(b.position().x, un[1].position().x);
  BOOST_REQUIRE_EQUAL(b.position().y, un[1].position().y);
  BOOST_REQUIRE_EQUAL(b.velocity().x, un[1].velocity().x);
  BOOST_REQUIRE_EQUAL(b.velocity().y, un[1].velocity().y);
  BOOST_REQUIRE_EQUAL(c.mass(), un[2].mass());
  BOOST_REQUIRE_EQUAL(c.position().x, un[2].position().x);
  BOOST_REQUIRE_EQUAL(c.position().y, un[2].position().y);
  BOOST_REQUIRE_EQUAL(c.velocity().x, un[2].velocity().x);
  BOOST_REQUIRE_EQUAL(c.velocity().y, un[2].velocity().y);
  BOOST_REQUIRE_EQUAL(d.mass(), un[3].mass());
  BOOST_REQUIRE_EQUAL(d.position().x, un[3].position().x);
  BOOST_REQUIRE_EQUAL(d.position().y, un[3].position().y);
  BOOST_REQUIRE_EQUAL(d.velocity().x, un[3].velocity().x);
  BOOST_REQUIRE_EQUAL(d.velocity().y, un[3].velocity().y);
  BOOST_REQUIRE_EQUAL(e.mass(), un[4].mass());
  BOOST_REQUIRE_EQUAL(e.position().x, un[4].position().x);
  BOOST_REQUIRE_EQUAL(e.position().y, un[4].position().y);
  BOOST_REQUIRE_EQUAL(e.velocity().x, un[4].velocity().x);
  BOOST_REQUIRE_EQUAL(e.velocity().y, un[4].velocity().y);
}

BOOST_AUTO_TEST_CASE(step25000) {
    Universe un("planets.txt");
    un.step(25000);
    un.step(25000);
    un.step(25000);
    double d = un[0].position().x;
    BOOST_REQUIRE_EQUAL(d, std::stod("149577777152"));
}
```

6 PS5: DNA Sequence Alignment

6.1 What I accomplished

In PS5: DNA Sequence Alignment, the task was to develop a program that computes the optimal sequence alignment of two DNA strings, using a method known as dynamic programming. This approach is critical in computational biology for determining the functional similarities between genes by analyzing their sequence alignment and calculating their edit distance, which accounts for mutations like insertions, deletions, and substitutions.

Figure 7: Window produced from running the program

My solution involved creating a robust algorithm to align two genetic sequences optimally by minimizing a cost function based on edit distances. This function penalizes gaps and mismatches between sequences, which models the likelihood of evolutionary mutations. I implemented this using a dynamic programming approach, where I constructed a matrix to store the minimal edit distances for substrings of the two DNA sequences, thus avoiding redundant recalculations that a naive recursive approach would entail.

For extra credit, I created an alternate min3() function that was slightly more efficient than the std version of the same function, and this was demonstrated in the seconds it took to execiute the program.

The main idea in this project was managing the complexity of dynamic programming in the context of biological data, which required careful consideration of the biological significance of each alignment decision. The matrix-based implementation needed to handle various scenarios efficiently, such as aligning long sequences with many mutations, which tested the limits of both space and time complexity of my approach. This project not only deepened my understanding of computational biology and dynamic programming but also honed my skills in designing efficient algorithms for complex data-driven problems.

From the project documentation and my README file, it's clear that the dynamic programming approach was essential for handling the computational demands of DNA sequence alignment. The matrix used in my solution efficiently calculated the optimal edit distances, and the results were thoroughly documented in the README file, demonstrating successful alignment under various test cases. This included handling large datasets like the ecoli sequences, where my implementation was able to process substantial amounts of data efficiently .

6.2 What I already knew

Before starting PS5: DNA Sequence Alignment, I was already familiar with using matrices, having utilized them extensively in previous projects. This prior experience was beneficial as it provided me with a strong foundation in handling matrix operations, which are crucial for implementing dynamic programming solutions in computational biology, espially with the Needleman-Wunsch method.

6.3 What I learned

During this project, I learned to implement the Needleman-Wunsch method. Applying this algorithm in code proved to be a challenging yet enlightening experience, as it involved translating theoretical concepts into practical programming constructs. This process deepened my understanding of how dynamic programming can be applied to solve complex problems in genomics.

6.4 Challenges

The primary challenge I encountered was managing the large matrices generated by the Needleman-Wunsch algorithm, as well a understanding the algorithm itself at first. The complexity of the algorithm itself compounded these difficulties, as any errors in the initial implementation magnified the challenges during testing. Navigating these large data structures required meticulous attention to detail and rigorous debugging to ensure the accuracy and efficiency of the sequence alignment.

6.5 Codebase

```makefile
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -lboost_unit_test_framework
# Your .hpp files
DEPS = EDistance.hpp
# Your compiled .o files
OBJECTS = EDistance.o
# The name of your program
PROGRAM = EDistance

.PHONY: all clean lint

all: $(PROGRAM) test EDistance.a

# Wildcard recipe to make .o files from corresponding .cpp file
%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

EDistance.a: EDistance.o
    ar rcs EDistance.a EDistance.o

test: test.o EDistance.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test.o: test.cpp EDistance.hpp
    $(CC) $(CFLAGS) -c $< -o $@

clean:
    rm *.o $(PROGRAM) *.a test

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

```cpp
// Copyright Camden Andersson 2024
#include "EDistance.hpp"
int main(int argc, char* argv[]) {
    // Load in the text file with the 2 strings
    std::string x, y;
    std::cin >> x >> y;
    // std::ifstream inputFile(fileName);
    // if (inputFile.is_open()) {
    //    inputFile >> x >> y;
    //    inputFile.close();
    // } else {
    //    std::cerr << "Unable to open file";
```

```
    //    return 1;
    // }

    // Start the clock, run the alignment code
    // sf::Clock clock;
    // EDistance ed(x, y);
    // std::cout << "opt[][] is initialized and seeded.\n";
    // std::cout << ed.printDynamicArray();

    // Start the clock, run the alignment code
    sf::Clock clock;
    EDistance ed(x, y);
    // std::cout << "opt[][] is initialized and seeded.\n";
    // std::cout << "x is " << x.length() << " chars\n";
    // std::cout << "y is " << y.length() << " chars\n";
    // std::cout << "\n";
    // At this point, the 2d array should be initialized

    // Compute the Edit Distance
    int distance = ed.optDistance();
    // std::cout << "opt[0][0] is the edit distance.\n";
    // std::cout << ed.printDynamicArray();
    std::string alignment = ed.alignment();
    sf::Time elapsed = clock.getElapsedTime();

    // Display the results
    std::cout << "Edit distance = " << distance << std::endl;
    std::cout << alignment;  // << std::endl;
    std::cout << "Execution time is " << elapsed.asSeconds() << " seconds"
   << std::endl;
    return 0;
}
```

```
// Copyright Camden Andersson 2024
#ifndef EDISTANCE_HPP
#define EDISTANCE_HPP

#include <vector>
#include <string>
#include <iostream>
#include <fstream>
#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>

class EDistance {
 public:
    EDistance(const std::string& x, const std::string& y);
    ~EDistance();
    int optDistance();
    std::string alignment();
    std::string printDynamicArray();
    static int penalty(char a, char b);
    static int min3(int a, int b, int c);
    static int min3std(int a, int b, int c);
    static int min3opt(int a, int b, int c);
 private:
    std::string x;  // the first DNA string
    std::string y;  // the second DNA string
    std::vector<std::vector<int>> opt;  // the n x m array to hold
```

```
    alignments
};
#endif /* EDISTANCE_HPP */
```

```
// Copyright Camden Andersson 2024
#include "EDistance.hpp"
#include <sstream>

EDistance::EDistance(const std::string& x, const std::string& y) : x(x), y(y
    ) {
    opt.resize(x.size() + 1, std::vector<int>(y.size() + 1, 0));

    // Resize the 2d array, and then seed the boundaries with a penalty of 2
    // The 2 increases as you move left and up
    for (size_t i = 0; i <= x.size(); ++i) {
        opt[i][y.size()] = 2 * (x.size() - i);
    }

    for (size_t j = 0; j <= y.size(); ++j) {
        opt[x.size()][j] = 2 * (y.size() - j);
    }
}

EDistance::~EDistance() {
    // Deallocate memory if allocated using new
    // not using new, so we can leave this as-is
}

int EDistance::penalty(char a, char b) {
    return (a == b) ? 0 : 1;
}

int EDistance::min3(int a, int b, int c) {
    // return min3std(a, b, c);
    return min3opt(a, b, c);
}

int EDistance::min3std(int a, int b, int c) {
    // Comparison of three items a, b, c
    return std::min(std::min(a, b), c);
}

int EDistance::min3opt(int a, int b, int c) {
    // An alternative method
    return a < b ? (a < c ? a : c) : (b < c ? b : c);
}

int EDistance::optDistance() {
    // We start at the bottom of the 2d matrix and work our way up
    // Right to left, then bottom to top
    for (int i = x.size() - 1; i >= 0; --i) {
        for (int j = y.size() - 1; j >= 0; --j) {
            opt[i][j] = min3(opt[i + 1][j + 1] + penalty(x[i], y[j]),
                opt[i + 1][j] + 2,
                opt[i][j + 1] + 2);
        }
    }
    // Our Edit Distance is in [0][0]
    return opt[0][0];
}
```

```

std::string EDistance::alignment() {
    std::string align;
    size_t i = 0;
    size_t j = 0;
    // Traverse the path and print out the string
    while (i < x.size() || j < y.size()) {
        if (i < x.size() && j < y.size() && opt[i][j] == opt[i + 1][j + 1] + penalty(x[i], y[j])) {
            align += x[i];
            align += ' ';
            align += y[j];
            align += ' ';
            align += std::to_string(penalty(x[i], y[j]));
            align += '\n';
            ++i;
            ++j;
        } else if (i < x.size() && opt[i][j] == opt[i + 1][j] + 2) {
            align += x[i];
            align += ' ';
            align += '-';
            align += ' ';
            align += "2\n";
            ++i;
        } else {
            align += '-';
            align += ' ';
            align += y[j];
            align += ' ';
            align += "2\n";
            ++j;
        }
    }

    return align;
}

// This will print the full array out
std::string EDistance::printDynamicArray() {
    std::string strOut = "";
    // size_t a = x.size();
    // size_t b = y.size();

    std::stringstream ss;
    for (const auto& row : opt) {
        for (int value : row) {
            ss << value << " ";
        }
        ss << "\n";
    }

    std::string optAsString = ss.str();
    return optAsString;
}
```

```
// Copyright Camden Andersson 2024

#include <sstream>
#include "EDistance.hpp"

```

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

// example10 and check output
BOOST_AUTO_TEST_CASE(testOutput) {
    std::string x, y;
    // example10
    x = "AACAGTTACC";
    y = "TAAGGTCA";
    EDistance ed(x, y);
    ed.optDistance();
    std::string alignment = ed.alignment();
    std::stringstream ss;
    std::stringstream s;
    boost::erase_all(alignment, "\n");
    ss << alignment;
    s << "A T 1";
    s << "A A 0";
    s << "C - 2";
    s << "A A 0";
    s << "G G 0";
    s << "T G 1";
    s << "T T 0";
    s << "A - 2";
    s << "C C 0";
    s << "C A 1";
    BOOST_REQUIRE_EQUAL(ss.str(), s.str());
}
// check if the cost = 7
BOOST_AUTO_TEST_CASE(testDistance) {
    std::string x, y;
    // example10
    x = "AACAGTTACC";
    y = "TAAGGTCA";
    EDistance ed(x, y);
    int distance = ed.optDistance();
    BOOST_REQUIRE_EQUAL(7, distance);
}
```

7 PS6: Random Writer

7.1 What I accomplished

For PS6: Random Writer, the task involved creating a probabilistic model based on k-grams (fixed sequences of k characters) to generate text that mimics the style of a given input text. This project was based on the principles of Markov chains, as proposed by Claude Shannon, to statistically model letter sequences and generate plausible random text.

Figure 8: Window produced from running the program

My solution centered around building a Markov model of order k, where each k-gram's occurrence in the text determined the likelihood of subsequent characters. I started by parsing the input text to identify all unique k-grams and recording the frequency of each character that followed each k-gram. This data was stored in a structured format that allowed quick access and modification, facilitating the generation of text based on the observed probabilities.

For the implementation, I utilized a map to associate each k-gram with its subsequent characters and their respective frequencies. This setup enabled me to efficiently query the next likely character given any k-gram, ensuring that the generated text reflected the statistical properties of the source text. I used my lambda in the freq(string, char) function, using findif(). To handle the text generation, I developed a function that, starting from a randomly chosen k-gram, used the model to produce text of a specified length by continuously appending characters based on the defined probabilities.

The most important part was managing the large amount of data involved in storing the frequencies for all possible k-grams, especially when considering higher orders of k. The complexity of the algorithm increased with k, as did the memory requirements, since each increase in k exponentially grew the number of possible k-grams.

The project results demonstrated the efficacy of the Markov model in generating text that, while nonsensical, bore a stylistic resemblance to the source material. The README also detailed the performance metrics, showing the execution time and memory usage, which were crucial for evaluating the scalability and efficiency of the algorithm.

7.2 What I already knew

Before starting, my knowledge of Markov models was minimal. I had only heard of them briefly and had no practical experience implementing them. This lack of familiarity meant that I was approaching this project with fresh eyes, ready to learn about a new and powerful statistical tool used across various fields of study.

7.3 What I learned

Through this project, I gained a comprehensive understanding of Markov models and their application in generating text. This was a fascinating exploration into how seemingly random elements can be structured in a way that produces surprisingly coherent and reasonable outputs. The project was a significant step up in complexity from earlier assignments, offering a deeper insight into probabilistic models and their practical implementations in computational tasks.

7.4 Challenges

The main challenge I faced was grasping the concept of a Markov model and applying it effectively to the task of text generation. Understanding the underlying principles of how previous states (or k-grams) influence the probability of subsequent events required a shift in my usual approach to programming. This conceptual hurdle was significant, as there was no straightforward method to simplify the complexity of Markov models. Developing a solid understanding of this model was crucial and proved to be the most challenging aspect of the project.

7.5 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
# Your .hpp files
DEPS = RandWriter.hpp
# Your compiled .o files
OBJECTS = RandWriter.o
# The name of your program
PROGRAM = TextWriter

.PHONY: all clean lint

all: $(PROGRAM) test TextWriter.a

# Wildcard recipe to make .o files from corresponding .cpp file
%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): TextWriter.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

TextWriter.a: RandWriter.o
    ar rcs TextWriter.a RandWriter.o

test: test.o RandWriter.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test.o: test.cpp RandWriter.hpp
    $(CC) $(CFLAGS) -c $< -o $@

clean:
    rm *.o $(PROGRAM) *.a test

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

```
// Copyright Camden Andersson 2024
#include <iostream>
#include "RandWriter.hpp"
#include "TextWriter.hpp"
```

```
// Copyright Camden Andersson 2024
#pragma once

#ifndef RANDWRITER_HPP
#define RANDWRITER_HPP

#include <string>
#include <map>
#include <fstream>

class RandWriter {
 public:
    RandWriter(const std::string& text, size_t k);

    size_t orderK() const;

```

```
    int freq(const std::string& kgram) const;

    int freq(const std::string& kgram, char c) const;

    char kRand(const std::string& kgram);

    std::string generate(const std::string& kgram, size_t L);

    friend std::ostream& operator<<(std::ostream& os, const RandWriter& rw);

 private:
    size_t k;
    std::string alphabet;
    std::map<std::string, std::map<char, int>> symbolTable;

    std::string circularText(const std::string& text, size_t k);

    void buildSymbolTable(const std::string& text);

    std::string activeAlphabet(const std::string& text);
};

class TextWriter {
 public:
    void TextWriter_main(int k, int L, const std::string& filename);
};

#endif /* RANDWRITER_HPP */
```

```
// Copyright Camden Andersson 2024
#include "RandWriter.hpp"
#include <iostream>
#include <random>
#include <exception>
#include <algorithm>

RandWriter::RandWriter(const std::string& text, size_t k) : k(k) {
    alphabet = activeAlphabet(text);
    if (text.length() < k) {
       throw std::invalid_argument("Constructor Error: text < k");
    }
    // Testing out circular text
    std::string circText = circularText(text, k);
    // std::cout << circText << std::endl;

    buildSymbolTable(circularText(text, k));
}

size_t RandWriter::orderK() const {
    return k;
}

std::map<char, int> findEntry(const std::map<std::string,
std::map<char, int>>& symbolTable, const std::string& key) {
    auto it = symbolTable.find(key);
    if (it != symbolTable.end()) {
        return it->second;
    } else {
        // Return an empty map if the key is not found
        return {};
```

```
    }
}

std::string RandWriter::circularText(const std::string& text, size_t k) {
    return text + text.substr(0, k);
}

void RandWriter::buildSymbolTable(const std::string& text) {
    for (size_t i = 0; i < text.length() - k; ++i) {
        std::string kgram = text.substr(i, k);
        char nextChar = text[i + k];
        symbolTable[kgram][nextChar]++;
    }
}

std::string RandWriter::activeAlphabet(const std::string& text) {
    std::vector<bool> found(128, false);
    std::string result;
    for (char c : text) {
        if (!found[c]) {
            found[c] = true;
            result += c;
        }
    }
    return result;
}

std::ostream& operator<<(std::ostream& os, const RandWriter& rw) {
    os << "Order: " << rw.orderK() << std::endl;
    os << "Alphabet: " << rw.alphabet << std::endl;
    for (const auto& entry : rw.symbolTable) {
        os << "K-gram: " << entry.first << std::endl;
        for (const auto& freqPair : entry.second) {
            os << "  Next char: " << freqPair.first <<
            " Frequency: " << freqPair.second << std::endl;
        }
    }
    return os;
}

char RandWriter::kRand(const std::string& kgram) {
    std::random_device rd;
    std::mt19937 gen(rd());

    // Rewrite
    std::vector<int> f;
    std::vector<char> possibleChars;
    int total = 0;
    int n = 0;

    if (kgram.size() != k) {
       throw std::length_error("kRand Length Error");
    }
    // Find the possible letters that follow kgram
    for (char c : alphabet) {
        n = freq(kgram, c);
        if (n > 0) {
            f.push_back(n);
            total += n;
```

```
            possibleChars.push_back(c);
        }
    }

    // TO DO: Throw an exception here
    if (possibleChars.size() == 0) {
       // std::cout << "EXCEPTION, will crash!" << std::endl;
       throw std::length_error("possibleChars.size() == 0");
    }

    // Build the percentages
    std::vector<double> percentages;
    for (size_t i = 0; i < f.size(); i++) {
        f[i] = static_cast<int>((static_cast<double>(f[i]) / static_cast<double>(total)) * 100.0);
    }
    std::discrete_distribution<> d(f.begin(), f.end());

    // Use the random generator
    int randomNum = d(gen);
    // char cReturn = possibleChars[randomNum];
    return possibleChars[randomNum];
}

int RandWriter::freq(const std::string& kgram) const {
    // Note that for this function you can't use the map's
    // count() function as that just returns 1 or 0.
    if (kgram.size() != k) {
       throw std::length_error("freq Length Error");
    }
    std::map<char, int> entry = findEntry(symbolTable, kgram);

    int nNumOccurences = 0;
    for (const auto& pair : entry) {
        // Sum up all the frequencies in the last part of the map
        nNumOccurences += pair.second;
    }

    return nNumOccurences;
}
/*
int RandWriter::freq(const std::string& kgram, char c) const {
    if (kgram.size() != k) {
       throw std::length_error("freq Length Error (2)");
    }
    if (symbolTable.count(kgram) && symbolTable.at(kgram).count(c)) {
        return symbolTable.at(kgram).at(c);
    }
    return 0;
}
*/
int RandWriter::freq(const std::string& kgram, char c) const {
    if (kgram.size() != k) {
        throw std::length_error("freq Length Error (2)");
    }
    if (symbolTable.count(kgram)) {
        const auto& charMap = symbolTable.at(kgram);

        auto it = std::find_if(charMap.begin(), charMap.end(), [c](const std
```

```
    ::pair<char, int>& p) {
            return p.first == c;
        });

        return (it != charMap.end()) ? it->second : 0;
    }
    return 0;
}



std::string RandWriter::generate(const std::string& kgram, size_t L) {
    if (kgram.size() != k) {
       throw std::length_error("generate Length Error");
    }
    std::string result = kgram;

    for (size_t i = k; i < L; ++i) {
        std::string sz = result.substr(result.length() - k);
        char nextChar = kRand(sz);
        result += nextChar;
    }
    return result;
}



std::string read_file_to_string(const std::string& filename) {
    std::ifstream file(filename);
    if (!file.is_open()) {
        std::cerr << "Failed to open file: " << filename << std::endl;
        return "";
    }

    std::string file_contents;
    file.seekg(0, std::ios::end);
    file_contents.reserve(file.tellg());
    file.seekg(0, std::ios::beg);

    file_contents.assign((std::istreambuf_iterator<char>(file)),
        std::istreambuf_iterator<char>());

    return file_contents;
}

std::string replace_newlines_with_spaces(const std::string& inText) {
    std::string result = inText;
    size_t startPos = 0;
    while ((startPos = result.find('\n', startPos)) != std::string::npos) {
        result.replace(startPos, 1, " ");
        startPos += 1;  // Move past the replaced character
    }
    return result;
}

void TextWriter::TextWriter_main(int k, int L, const std::string& filename)
    {
    // Open the file that contains the text & display it
    // std::string rawText = read_file_to_string(filename);
```

```
    //  if (!rawText.empty()) {
    //    std::cout << "File contents:\n" << rawText << std::endl;
    //  }
    std::string text = replace_newlines_with_spaces(filename);

    // Build the Markov model and display the frequencies
    RandWriter rw(text, k);
    // std::cout << rw << std::endl;

    // Set the seed kgram and generate the text
    std::string kgram = text.substr(0, k);
    std::cout << rw.generate(kgram, L) << std::endl;
}
```

```
// Copyright Camden Andersson 2024
#include "RandWriter.hpp"

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

BOOST_AUTO_TEST_CASE(wrongLength) {
    BOOST_REQUIRE_THROW(RandWriter r("a", 2), std::invalid_argument);
}

BOOST_AUTO_TEST_CASE(generateTest) {
    RandWriter rw("abcdabcdabcd", 2);
    std::string generated = rw.generate("ab", 5);
    BOOST_REQUIRE_EQUAL(generated.size(), 5);
    BOOST_REQUIRE(generated.substr(0, 2) == "ab");
}

BOOST_AUTO_TEST_CASE(wrongDistributionTest) {
    RandWriter rw("aaaaabbbbcc", 2);
    std::map<char, int> charCount;
    for (int i = 0; i < 1000; ++i) {
        std::string generated = rw.generate("aa", 4);
        charCount[generated[2]]++;
    }
    BOOST_REQUIRE_CLOSE(static_cast<double>(charCount['a']) / 1000.0, 0.74,
   10.0);
    BOOST_REQUIRE_CLOSE(static_cast<double>(charCount['b']) / 1000.0, 0.26,
   10.0);
}
BOOST_AUTO_TEST_CASE(passTestAmount) {
    BOOST_REQUIRE_EQUAL(1, 1);
}
```

8 PS7: Kronos Log Parsing

8.1 What I accomplished

For PS7: Kronos Time Clock, the project involved parsing and analyzing log files from the Kronos InTouch time clock device using regular expressions. The primary task was to detect and report on the device's boot-up sequences, identifying successful startups and any failures by examining specific log entries that marked the start and end of each boot process.

My approach was to develop a robust parser that utilized regular expressions to sift through the log files and extract relevant information about the device's startup sequences. The implementation involved two main regular expressions: one to detect the initiation of a boot sequence and another to identify its completion. These expressions were designed to capture the necessary timestamps to calculate the duration of each boot process.

To handle the log analysis, I implemented a function that read through the entire log file line by line, applying the regular expressions to determine the start and end of each boot sequence. If a boot sequence was initiated but did not have a corresponding completion before the next sequence began or the file ended, it was reported as a failure. Conversely, successful boot sequences were recorded with their duration calculated using the Boost date/time library to ensure precision.

The implementation also involved managing the pairing of start and end log entries without any overlap or nesting, which was crucial for accurate reporting. Each identified sequence was output with either a success message and the elapsed time or a failure notice, as specified in the project requirements.

For extra credit, I enhanced the output by generating a summary header that provided an overview of all boot sequences, including their start and completion times. This summary was designed to give a quick snapshot of the device's operational history over the period covered by the logs, making it easier for technicians to identify patterns or recurring issues.

This project not only reinforced my skills in using regular expressions but also in applying them to a real-world problem where precision and accuracy are critical. The ability to parse complex log files and extract meaningful information from them is a valuable skill in many software engineering and data analysis roles.

8.2 What I already knew

Before starting on PS7: Kronos Time Clock, I was already quite experienced with log parsing, having previously utilized various tools and programs for this purpose. This background gave me a solid foundation and a comfort level with the task of extracting and analyzing information from log files, making the initial aspects of this project feel quite familiar.

8.3 What I learned

Through this project, I learned to use and understand regular expressions, a tool I had not previously been exposed to. Discovering the power and versatility of regular expressions was enlightening; they proved to be incredibly effective for pattern matching and data extraction in complex text files. This skill has significantly broadened my capabilities in data processing and will undoubtedly be valuable in future projects.

8.4 Challenges

The primary challenge in this project arose during the extra credit portion. While the base assignment was relatively straightforward and within my comfort zone due to my prior experience with log parsing, extending the functionality for the extra credit proved to be more demanding. Despite these challenges, the project required less time and effort overall compared to previous assignments, demonstrating a good balance between complexity and manageability. Unfortunately, the tests for log3 beginning and end and the log6 beginning and end were unable to pass, despite many attempts to fix, so ultimately credit was lost for that. I believe that there was some sort of formatting issue, but I couldn't find the cause in time.

Following is the .rpt log file produced for log 6. As you can see, it incorporates the extra credit (both the header and the extra services).

```
Device Boot Report

InTouch log file: device6_intouch.log
Lines Scanned: 90671

Device boot count: initiated = 14, completed: 12


=== Device boot ===
2(device6_intouch.log): 2014-04-03 20:27:48 Boot Start
161(device6_intouch.log): 2014-04-03 20:31:01 Boot Completed
    Boot Time: 193000ms

Services
    ProtocolService
        Start:148(device6_intouch.log)
        Completed:155(device6_intouch.log)
        ElapsedTime:13020
    OfflineSmartviewService
        OfflineSmartviewServiceStart:139(device6_intouch.log)
        Completed:140(device6_intouch.log)
        ElapsedTime:11
    BiometricService
        BiometricServiceStart:134(device6_intouch.log)
        Completed:136(device6_intouch.log)
        ElapsedTime:176
    SoftLoadService
        Start:148(device6_intouch.log)
        Completed:152(device6_intouch.log)
        ElapsedTime:2932
    GateService
        GateServiceStart:132(device6_intouch.log)
        Completed:133(device6_intouch.log)
        ElapsedTime:1
    StateManager
        StateManagerStart:137(device6_intouch.log)
        Completed:138(device6_intouch.log)
        ElapsedTime:1660
    AVFeedbackService
        AVFeedbackServiceStart:130(device6_intouch.log)
        Completed:131(device6_intouch.log)
        ElapsedTime:27
    ReaderDataService
        ReaderDataServiceStart:141(device6_intouch.log)
        Completed:142(device6_intouch.log)
        ElapsedTime:11
    Logging
        LoggingStart:18(device6_intouch.log)
        Completed:19(device6_intouch.log)
        ElapsedTime:271
    DatabaseInitialize
        DatabaseInitializeStart:20(device6_intouch.log)
        Completed:44(device6_intouch.log)
        ElapsedTime:32365
    MessagingService
        MessagingServiceStart:45(device6_intouch.log)
        Completed:47(device6_intouch.log)
        ElapsedTime:5488
    DatabaseThreads
```

```

    DiagnosticsServiceProtocolServiceWATCHDOGSoftLoadServiceDiagnosticsServiceStart
    :148(device6_intouch.log)
        Start:148(device6_intouch.log)
        Completed:151(device6_intouch.log)
        ElapsedTime:398
    ConfigurationService
        ConfigurationServiceStart:114(device6_intouch.log)
        Completed:115(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:116(device6_intouch.log)
        Completed:117(device6_intouch.log)
        ElapsedTime:1143
    StagingService
        StagingServiceStart:122(device6_intouch.log)
        Completed:123(device6_intouch.log)
        ElapsedTime:3290
    WATCHDOG
        Start:148(device6_intouch.log)
        Completed:150(device6_intouch.log)
        ElapsedTime:190
    DiagnosticsService
        Completed:149(device6_intouch.log)
        ElapsedTime:206
    HealthMonitorService
        HealthMonitorServiceStart:109(device6_intouch.log)
        Completed:110(device6_intouch.log)
        ElapsedTime:247
    BellService
        BellServiceStart:128(device6_intouch.log)
        Completed:129(device6_intouch.log)
        ElapsedTime:2
    Persistence
        PersistenceStart:111(device6_intouch.log)
        Completed:113(device6_intouch.log)
        ElapsedTime:22278
    ThemingService
        ThemingServiceStart:118(device6_intouch.log)
        Completed:119(device6_intouch.log)
        ElapsedTime:0
    LandingPadService
        LandingPadServiceStart:120(device6_intouch.log)
        Completed:121(device6_intouch.log)
        ElapsedTime:2
    PortConfigurationService
        PortConfigurationServiceStart:124(device6_intouch.log)
        Completed:125(device6_intouch.log)
        ElapsedTime:54
    DeviceIOService
        DeviceIOServiceStart:126(device6_intouch.log)
        Completed:127(device6_intouch.log)
        ElapsedTime:26

=== Device boot ===
82079(device6_intouch.log): 2014-04-09 14:51:15 Boot Start
82303(device6_intouch.log): 2014-04-09 14:54:39 Boot Completed
    Boot Time: 204000ms

```

```
Services
    ThemingService
        ThemingServiceStart:82224(device6_intouch.log)
        Completed:82225(device6_intouch.log)
        ElapsedTime:1
    OfflineSmartviewService
        OfflineSmartviewServiceStart:82222(device6_intouch.log)
        Completed:82223(device6_intouch.log)
        ElapsedTime:12
    BiometricService
        BiometricServiceStart:82217(device6_intouch.log)
        Completed:82219(device6_intouch.log)
        ElapsedTime:19
    ReaderDataService
        ReaderDataServiceStart:82215(device6_intouch.log)
        Completed:82216(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:82099(device6_intouch.log)
        Completed:82100(device6_intouch.log)
        ElapsedTime:591
    DatabaseInitialize
        DatabaseInitializeStart:82101(device6_intouch.log)
        Completed:82127(device6_intouch.log)
        ElapsedTime:35131
    MessagingService
        MessagingServiceStart:82128(device6_intouch.log)
        Completed:82130(device6_intouch.log)
        ElapsedTime:6351
    DatabaseThreads
    ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceDiagnosticsServiceStart
    :82276(device6_intouch.log)
        Start:82276(device6_intouch.log)
        Completed:82281(device6_intouch.log)
        ElapsedTime:2836
    ConfigurationService
        ConfigurationServiceStart:82197(device6_intouch.log)
        Completed:82198(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:82201(device6_intouch.log)
        Completed:82202(device6_intouch.log)
        ElapsedTime:895
    StateManager
        StateManagerStart:82220(device6_intouch.log)
        Completed:82221(device6_intouch.log)
        ElapsedTime:2336
    AVFeedbackService
        AVFeedbackServiceStart:82209(device6_intouch.log)
        Completed:82210(device6_intouch.log)
        ElapsedTime:43
    StagingService
        StagingServiceStart:82211(device6_intouch.log)
        Completed:82212(device6_intouch.log)
        ElapsedTime:8587
    WATCHDOG
        Start:82276(device6_intouch.log)
        Completed:82278(device6_intouch.log)
```

```
        ElapsedTime:284
    DiagnosticsService
        Completed:82277(device6_intouch.log)
        ElapsedTime:254
    HealthMonitorService
        HealthMonitorServiceStart:82192(device6_intouch.log)
        Completed:82193(device6_intouch.log)
        ElapsedTime:164
    BellService
        BellServiceStart:82213(device6_intouch.log)
        Completed:82214(device6_intouch.log)
        ElapsedTime:1
    Persistence
        PersistenceStart:82194(device6_intouch.log)
        Completed:82196(device6_intouch.log)
        ElapsedTime:19408
    PortConfigurationService
        PortConfigurationServiceStart:82199(device6_intouch.log)
        Completed:82200(device6_intouch.log)
        ElapsedTime:57
    DeviceIOService
        DeviceIOServiceStart:82203(device6_intouch.log)
        Completed:82204(device6_intouch.log)
        ElapsedTime:62
    LandingPadService
        LandingPadServiceStart:82205(device6_intouch.log)
        Completed:82206(device6_intouch.log)
        ElapsedTime:2
    SoftLoadService
        Start:82276(device6_intouch.log)
        Completed:82282(device6_intouch.log)
        ElapsedTime:2927
    GateService
        GateServiceStart:82207(device6_intouch.log)
        Completed:82208(device6_intouch.log)
        ElapsedTime:2

=== Device boot ===
85398(device6_intouch.log): 2014-04-10 18:13:13 Boot Start
85564(device6_intouch.log): 2014-04-10 18:16:37 Boot Completed
    Boot Time: 204000ms

Services
    SoftLoadService
        Start:85553(device6_intouch.log)
        Completed:85561(device6_intouch.log)
        ElapsedTime:2828
    GateService
        GateServiceStart:85544(device6_intouch.log)
        Completed:85545(device6_intouch.log)
        ElapsedTime:1
    OfflineSmartviewService
        OfflineSmartviewServiceStart:85540(device6_intouch.log)
        Completed:85541(device6_intouch.log)
        ElapsedTime:15
    AVFeedbackService
        AVFeedbackServiceStart:85542(device6_intouch.log)
        Completed:85543(device6_intouch.log)
        ElapsedTime:38
```

```
    StateManager
        StateManagerStart:85534(device6_intouch.log)
        Completed:85535(device6_intouch.log)
        ElapsedTime:2870
    ReaderDataService
        ReaderDataServiceStart:85538(device6_intouch.log)
        Completed:85539(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:85419(device6_intouch.log)
        Completed:85420(device6_intouch.log)
        ElapsedTime:256
    DatabaseInitialize
        DatabaseInitializeStart:85421(device6_intouch.log)
        Completed:85447(device6_intouch.log)
        ElapsedTime:35191
    MessagingService
        MessagingServiceStart:85448(device6_intouch.log)
        Completed:85450(device6_intouch.log)
        ElapsedTime:7848
    DatabaseThreads
    ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceDiagnosticsServiceStart
    :85553(device6_intouch.log)
        Start:85553(device6_intouch.log)
        Completed:85559(device6_intouch.log)
        ElapsedTime:2802
    ConfigurationService
        ConfigurationServiceStart:85517(device6_intouch.log)
        Completed:85518(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:85523(device6_intouch.log)
        Completed:85524(device6_intouch.log)
        ElapsedTime:1048
    StagingService
        StagingServiceStart:85525(device6_intouch.log)
        Completed:85526(device6_intouch.log)
        ElapsedTime:8343
    WATCHDOG
        Start:85553(device6_intouch.log)
        Completed:85555(device6_intouch.log)
        ElapsedTime:421
    DiagnosticsService
        Completed:85554(device6_intouch.log)
        ElapsedTime:255
    HealthMonitorService
        HealthMonitorServiceStart:85512(device6_intouch.log)
        Completed:85513(device6_intouch.log)
        ElapsedTime:211
    BellService
        BellServiceStart:85536(device6_intouch.log)
        Completed:85537(device6_intouch.log)
        ElapsedTime:3
    Persistence
        PersistenceStart:85514(device6_intouch.log)
        Completed:85516(device6_intouch.log)
        ElapsedTime:18367
    ThemingService
```

```
        ThemingServiceStart:85519(device6_intouch.log)
        Completed:85520(device6_intouch.log)
        ElapsedTime:0
    LandingPadService
        LandingPadServiceStart:85521(device6_intouch.log)
        Completed:85522(device6_intouch.log)
        ElapsedTime:2
    PortConfigurationService
        PortConfigurationServiceStart:85527(device6_intouch.log)
        Completed:85528(device6_intouch.log)
        ElapsedTime:52
    DeviceIOService
        DeviceIOServiceStart:85529(device6_intouch.log)
        Completed:85530(device6_intouch.log)
        ElapsedTime:24
    BiometricService
        BiometricServiceStart:85531(device6_intouch.log)
        Completed:85533(device6_intouch.log)
        ElapsedTime:12

=== Device boot ===
85957(device6_intouch.log): 2014-04-10 19:11:05 Boot Start
86123(device6_intouch.log): 2014-04-10 19:14:24 Boot Completed
    Boot Time: 199000ms

Services
    OfflineSmartviewService
        OfflineSmartviewServiceStart:86103(device6_intouch.log)
        Completed:86104(device6_intouch.log)
        ElapsedTime:41
    BiometricService
        BiometricServiceStart:86096(device6_intouch.log)
        Completed:86098(device6_intouch.log)
        ElapsedTime:13
    StateManager
        StateManagerStart:86101(device6_intouch.log)
        Completed:86102(device6_intouch.log)
        ElapsedTime:2027
    AVFeedbackService
        AVFeedbackServiceStart:86092(device6_intouch.log)
        Completed:86093(device6_intouch.log)
        ElapsedTime:32
    ReaderDataService
        ReaderDataServiceStart:86094(device6_intouch.log)
        Completed:86095(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:85978(device6_intouch.log)
        Completed:85979(device6_intouch.log)
        ElapsedTime:289
    DatabaseInitialize
        DatabaseInitializeStart:85980(device6_intouch.log)
        Completed:86004(device6_intouch.log)
        ElapsedTime:33943
    MessagingService
        MessagingServiceStart:86007(device6_intouch.log)
        Completed:86009(device6_intouch.log)
        ElapsedTime:7570
    DatabaseThreads
```

```
ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceWATCHDOGStart:86112(device6_intouch.log)
        Start:86112(device6_intouch.log)
        Completed:86117(device6_intouch.log)
        ElapsedTime:2786
    ConfigurationService
        ConfigurationServiceStart:86076(device6_intouch.log)
        Completed:86077(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:86084(device6_intouch.log)
        Completed:86085(device6_intouch.log)
        ElapsedTime:1114
    StagingService
        StagingServiceStart:86086(device6_intouch.log)
        Completed:86087(device6_intouch.log)
        ElapsedTime:7899
    DiagnosticsService
        Start:86112(device6_intouch.log)
        Completed:86114(device6_intouch.log)
        ElapsedTime:202
    WATCHDOG
        Completed:86113(device6_intouch.log)
        ElapsedTime:82
    HealthMonitorService
        HealthMonitorServiceStart:86071(device6_intouch.log)
        Completed:86072(device6_intouch.log)
        ElapsedTime:264
    BellService
        BellServiceStart:86099(device6_intouch.log)
        Completed:86100(device6_intouch.log)
        ElapsedTime:1
    Persistence
        PersistenceStart:86073(device6_intouch.log)
        Completed:86075(device6_intouch.log)
        ElapsedTime:18488
    LandingPadService
        LandingPadServiceStart:86078(device6_intouch.log)
        Completed:86079(device6_intouch.log)
        ElapsedTime:2
    ThemingService
        ThemingServiceStart:86080(device6_intouch.log)
        Completed:86081(device6_intouch.log)
        ElapsedTime:1
    PortConfigurationService
        PortConfigurationServiceStart:86082(device6_intouch.log)
        Completed:86083(device6_intouch.log)
        ElapsedTime:95
    DeviceIOService
        DeviceIOServiceStart:86088(device6_intouch.log)
        Completed:86089(device6_intouch.log)
        ElapsedTime:24
    SoftLoadService
        Start:86112(device6_intouch.log)
        Completed:86119(device6_intouch.log)
        ElapsedTime:2991
    GateService
        GateServiceStart:86090(device6_intouch.log)
```

```
        Completed:86091(device6_intouch.log)
        ElapsedTime:2

=== Device boot ===
86127(device6_intouch.log): 2014-04-10 19:18:36 Boot Start
86293(device6_intouch.log): 2014-04-10 19:21:56 Boot Completed
    Boot Time: 200000ms

Services
    SoftLoadService
        Start:86282(device6_intouch.log)
        Completed:86289(device6_intouch.log)
        ElapsedTime:2835
    GateService
        GateServiceStart:86273(device6_intouch.log)
        Completed:86274(device6_intouch.log)
        ElapsedTime:1
    OfflineSmartviewService
        OfflineSmartviewServiceStart:86269(device6_intouch.log)
        Completed:86270(device6_intouch.log)
        ElapsedTime:15
    AVFeedbackService
        AVFeedbackServiceStart:86271(device6_intouch.log)
        Completed:86272(device6_intouch.log)
        ElapsedTime:33
    StateManager
        StateManagerStart:86263(device6_intouch.log)
        Completed:86264(device6_intouch.log)
        ElapsedTime:3052
    ReaderDataService
        ReaderDataServiceStart:86267(device6_intouch.log)
        Completed:86268(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:86148(device6_intouch.log)
        Completed:86149(device6_intouch.log)
        ElapsedTime:282
    DatabaseInitialize
        DatabaseInitializeStart:86150(device6_intouch.log)
        Completed:86176(device6_intouch.log)
        ElapsedTime:34292
    MessagingService
        MessagingServiceStart:86177(device6_intouch.log)
        Completed:86179(device6_intouch.log)
        ElapsedTime:6619
    DatabaseThreads
    ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceDiagnosticsServiceStart
    :86282(device6_intouch.log)
        Start:86282(device6_intouch.log)
        Completed:86287(device6_intouch.log)
        ElapsedTime:2833
    ConfigurationService
        ConfigurationServiceStart:86246(device6_intouch.log)
        Completed:86247(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:86252(device6_intouch.log)
        Completed:86253(device6_intouch.log)
```

```
        ElapsedTime:963
    StagingService
        StagingServiceStart:86254(device6_intouch.log)
        Completed:86255(device6_intouch.log)
        ElapsedTime:8053
    WATCHDOG
        Start:86282(device6_intouch.log)
        Completed:86284(device6_intouch.log)
        ElapsedTime:178
    DiagnosticsService
        Completed:86283(device6_intouch.log)
        ElapsedTime:114
    HealthMonitorService
        HealthMonitorServiceStart:86241(device6_intouch.log)
        Completed:86242(device6_intouch.log)
        ElapsedTime:187
    BellService
        BellServiceStart:86265(device6_intouch.log)
        Completed:86266(device6_intouch.log)
        ElapsedTime:2
    Persistence
        PersistenceStart:86243(device6_intouch.log)
        Completed:86245(device6_intouch.log)
        ElapsedTime:18914
    ThemingService
        ThemingServiceStart:86248(device6_intouch.log)
        Completed:86249(device6_intouch.log)
        ElapsedTime:0
    LandingPadService
        LandingPadServiceStart:86250(device6_intouch.log)
        Completed:86251(device6_intouch.log)
        ElapsedTime:2
    PortConfigurationService
        PortConfigurationServiceStart:86256(device6_intouch.log)
        Completed:86257(device6_intouch.log)
        ElapsedTime:56
    DeviceIOService
        DeviceIOServiceStart:86258(device6_intouch.log)
        Completed:86259(device6_intouch.log)
        ElapsedTime:22
    BiometricService
        BiometricServiceStart:86260(device6_intouch.log)
        Completed:86262(device6_intouch.log)
        ElapsedTime:12

=== Device boot ===
86568(device6_intouch.log): 2014-04-10 19:32:16 Boot Start
86732(device6_intouch.log): 2014-04-10 19:35:36 Boot Completed
    Boot Time: 200000ms

Services
    OfflineSmartviewService
        OfflineSmartviewServiceStart:86712(device6_intouch.log)
        Completed:86713(device6_intouch.log)
        ElapsedTime:20
    BiometricService
        BiometricServiceStart:86707(device6_intouch.log)
        Completed:86709(device6_intouch.log)
        ElapsedTime:12
```

```
    ThemingService
        ThemingServiceStart:86705(device6_intouch.log)
        Completed:86706(device6_intouch.log)
        ElapsedTime:0
    ReaderDataService
        ReaderDataServiceStart:86703(device6_intouch.log)
        Completed:86704(device6_intouch.log)
        ElapsedTime:499
    Logging
        LoggingStart:86588(device6_intouch.log)
        Completed:86589(device6_intouch.log)
        ElapsedTime:291
    DatabaseInitialize
        DatabaseInitializeStart:86590(device6_intouch.log)
        Completed:86614(device6_intouch.log)
        ElapsedTime:33829
    MessagingService
        MessagingServiceStart:86617(device6_intouch.log)
        Completed:86619(device6_intouch.log)
        ElapsedTime:5723
    DatabaseThreads

    ProtocolServiceDiagnosticsServiceWATCHDOGSoftLoadServiceWATCHDOGStart
    :86721(device6_intouch.log)
        Start:86721(device6_intouch.log)
        Completed:86726(device6_intouch.log)
        ElapsedTime:2953
    ConfigurationService
        ConfigurationServiceStart:86685(device6_intouch.log)
        Completed:86686(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:86695(device6_intouch.log)
        Completed:86696(device6_intouch.log)
        ElapsedTime:872
    StateManager
        StateManagerStart:86710(device6_intouch.log)
        Completed:86711(device6_intouch.log)
        ElapsedTime:2273
    AVFeedbackService
        AVFeedbackServiceStart:86697(device6_intouch.log)
        Completed:86698(device6_intouch.log)
        ElapsedTime:34
    StagingService
        StagingServiceStart:86699(device6_intouch.log)
        Completed:86700(device6_intouch.log)
        ElapsedTime:7576
    DiagnosticsService
        Start:86721(device6_intouch.log)
        Completed:86723(device6_intouch.log)
        ElapsedTime:104
    WATCHDOG
        Completed:86722(device6_intouch.log)
        ElapsedTime:97
    HealthMonitorService
        HealthMonitorServiceStart:86681(device6_intouch.log)
        Completed:86682(device6_intouch.log)
        ElapsedTime:173
    BellService
```

```
        BellServiceStart:86701(device6_intouch.log)
        Completed:86702(device6_intouch.log)
        ElapsedTime:2
    Persistence
        PersistenceStart:86683(device6_intouch.log)
        Completed:86684(device6_intouch.log)
        ElapsedTime:22924
    LandingPadService
        LandingPadServiceStart:86687(device6_intouch.log)
        Completed:86688(device6_intouch.log)
        ElapsedTime:2
    PortConfigurationService
        PortConfigurationServiceStart:86689(device6_intouch.log)
        Completed:86690(device6_intouch.log)
        ElapsedTime:65
    DeviceIOService
        DeviceIOServiceStart:86691(device6_intouch.log)
        Completed:86692(device6_intouch.log)
        ElapsedTime:53
    SoftLoadService
        Start:86721(device6_intouch.log)
        Completed:86729(device6_intouch.log)
        ElapsedTime:3194
    GateService
        GateServiceStart:86693(device6_intouch.log)
        Completed:86694(device6_intouch.log)
        ElapsedTime:3

=== Device boot ===
86750(device6_intouch.log): 2014-04-10 20:06:27 Boot Start
86821(device6_intouch.log): 2014-04-10 20:09:07 Boot Completed
    Boot Time: 160000ms

Services
    OfflineSmartviewService
        OfflineSmartviewServiceStart:86801(device6_intouch.log)
        Completed:86802(device6_intouch.log)
        ElapsedTime:12
    BiometricService
        BiometricServiceStart:86797(device6_intouch.log)
        Completed:86798(device6_intouch.log)
        ElapsedTime:685
    StagingService
        StagingServiceStart:86795(device6_intouch.log)
        Completed:86796(device6_intouch.log)
        ElapsedTime:6961
    ReaderDataService
        ReaderDataServiceStart:86805(device6_intouch.log)
        Completed:86806(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:86765(device6_intouch.log)
        Completed:86766(device6_intouch.log)
        ElapsedTime:305
    DatabaseInitialize
        DatabaseInitializeStart:86767(device6_intouch.log)
        Completed:86768(device6_intouch.log)
        ElapsedTime:33168
    MessagingService
```

```
        MessagingServiceStart:86769(device6_intouch.log)
        Completed:86770(device6_intouch.log)
        ElapsedTime:5776
    DatabaseThreads
        Start:86812(device6_intouch.log)
        Completed:86820(device6_intouch.log)
        ElapsedTime:4099
    ConfigurationService
        ConfigurationServiceStart:86779(device6_intouch.log)
        Completed:86780(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:86781(device6_intouch.log)
        Completed:86782(device6_intouch.log)
        ElapsedTime:1915
    StateManager
        StateManagerStart:86799(device6_intouch.log)
        Completed:86800(device6_intouch.log)
        ElapsedTime:1793
    AVFeedbackService
        AVFeedbackServiceStart:86793(device6_intouch.log)
        Completed:86794(device6_intouch.log)
        ElapsedTime:53
    WATCHDOG
        Start:86812(device6_intouch.log)
        Completed:86815(device6_intouch.log)
        ElapsedTime:415
    DiagnosticsService
    ProtocolServiceWATCHDOGSoftLoadServiceDatabaseThreadsDiagnosticsServiceStart
    :86812(device6_intouch.log)
        Completed:86814(device6_intouch.log)
        ElapsedTime:250
    HealthMonitorService
        HealthMonitorServiceStart:86775(device6_intouch.log)
        Completed:86776(device6_intouch.log)
        ElapsedTime:575
    BellService
        BellServiceStart:86803(device6_intouch.log)
        Completed:86804(device6_intouch.log)
        ElapsedTime:2
    Persistence
        PersistenceStart:86777(device6_intouch.log)
        Completed:86778(device6_intouch.log)
        ElapsedTime:18205
    ThemingService
        ThemingServiceStart:86783(device6_intouch.log)
        Completed:86784(device6_intouch.log)
        ElapsedTime:0
    LandingPadService
        LandingPadServiceStart:86785(device6_intouch.log)
        Completed:86786(device6_intouch.log)
        ElapsedTime:2
    PortConfigurationService
        PortConfigurationServiceStart:86787(device6_intouch.log)
        Completed:86788(device6_intouch.log)
        ElapsedTime:43
    DeviceIOService
        DeviceIOServiceStart:86789(device6_intouch.log)
```

```
        Completed:86790(device6_intouch.log)
        ElapsedTime:24
    SoftLoadService
        Start:86812(device6_intouch.log)
        Completed:86817(device6_intouch.log)
        ElapsedTime:3307
    GateService
        GateServiceStart:86791(device6_intouch.log)
        Completed:86792(device6_intouch.log)
        ElapsedTime:196

=== Device boot ===
86939(device6_intouch.log): 2014-04-11 00:15:56 Boot Start
87111(device6_intouch.log): 2014-04-11 00:18:49 Boot Completed
    Boot Time: 173000ms

Services
    OfflineSmartviewService
        OfflineSmartviewServiceStart:87030(device6_intouch.log)
        Completed:87031(device6_intouch.log)
        ElapsedTime:18
    StateManager
        StateManagerStart:87028(device6_intouch.log)
        Completed:87029(device6_intouch.log)
        ElapsedTime:2752
    AVFeedbackService
        AVFeedbackServiceStart:87026(device6_intouch.log)
        Completed:87027(device6_intouch.log)
        ElapsedTime:34
    SoftLoadService
        Start:87083(device6_intouch.log)
        Completed:87099(device6_intouch.log)
        ElapsedTime:4041
    GateService
        GateServiceStart:87024(device6_intouch.log)
        Completed:87025(device6_intouch.log)
        ElapsedTime:158
    BiometricService
        BiometricServiceStart:87022(device6_intouch.log)
        Completed:87023(device6_intouch.log)
        ElapsedTime:677
    DatabaseInitialize
        DatabaseInitializeStart:86961(device6_intouch.log)
        Completed:86987(device6_intouch.log)
        ElapsedTime:37703
    MessagingService
        MessagingServiceStart:86989(device6_intouch.log)
        Completed:86991(device6_intouch.log)
        ElapsedTime:8660
    DatabaseThreads

    ProtocolServiceSoftLoadServiceDiagnosticsServiceWATCHDOGWATCHDOGStart
    :87083(device6_intouch.log)
        Start:87083(device6_intouch.log)
        Completed:87088(device6_intouch.log)
        ElapsedTime:3747
    ConfigurationService
        ConfigurationServiceStart:87006(device6_intouch.log)
        Completed:87007(device6_intouch.log)
```

```
        ElapsedTime:0
    CacheService
        CacheServiceStart:87008(device6_intouch.log)
        Completed:87009(device6_intouch.log)
        ElapsedTime:988
    StagingService
        StagingServiceStart:87016(device6_intouch.log)
        Completed:87017(device6_intouch.log)
        ElapsedTime:8398
    DiagnosticsService
        Start:87083(device6_intouch.log)
        Completed:87085(device6_intouch.log)
        ElapsedTime:240
    WATCHDOG
        Completed:87084(device6_intouch.log)
        ElapsedTime:229
    HealthMonitorService
        HealthMonitorServiceStart:87002(device6_intouch.log)
        Completed:87003(device6_intouch.log)
        ElapsedTime:219
    BellService
        BellServiceStart:87032(device6_intouch.log)
        Completed:87033(device6_intouch.log)
        ElapsedTime:2
    Persistence
        PersistenceStart:87004(device6_intouch.log)
        Completed:87005(device6_intouch.log)
        ElapsedTime:19331
    ThemingService
        ThemingServiceStart:87010(device6_intouch.log)
        Completed:87011(device6_intouch.log)
        ElapsedTime:0
    LandingPadService
        LandingPadServiceStart:87012(device6_intouch.log)
        Completed:87013(device6_intouch.log)
        ElapsedTime:10
    PortConfigurationService
        PortConfigurationServiceStart:87014(device6_intouch.log)
        Completed:87015(device6_intouch.log)
        ElapsedTime:44
    DeviceIOService
        DeviceIOServiceStart:87018(device6_intouch.log)
        Completed:87019(device6_intouch.log)
        ElapsedTime:22
    Logging
        LoggingStart:86959(device6_intouch.log)
        Completed:86960(device6_intouch.log)
        ElapsedTime:410
    ReaderDataService
        ReaderDataServiceStart:87020(device6_intouch.log)
        Completed:87021(device6_intouch.log)
        ElapsedTime:2

=== Device boot ===
87116(device6_intouch.log): 2014-04-11 13:28:25 Boot Start
87286(device6_intouch.log): 2014-04-11 13:31:12 Boot Completed
    Boot Time: 167000ms

Services
```

```
    OfflineSmartviewService
        OfflineSmartviewServiceStart:87208(device6_intouch.log)
        Completed:87209(device6_intouch.log)
        ElapsedTime:22
    BiometricService
        BiometricServiceStart:87204(device6_intouch.log)
        Completed:87205(device6_intouch.log)
        ElapsedTime:688
    StagingService
        StagingServiceStart:87198(device6_intouch.log)
        Completed:87199(device6_intouch.log)
        ElapsedTime:8737
    ReaderDataService
        ReaderDataServiceStart:87200(device6_intouch.log)
        Completed:87201(device6_intouch.log)
        ElapsedTime:3
    Logging
        LoggingStart:87136(device6_intouch.log)
        Completed:87137(device6_intouch.log)
        ElapsedTime:322
    DatabaseInitialize
        DatabaseInitializeStart:87138(device6_intouch.log)
        Completed:87164(device6_intouch.log)
        ElapsedTime:34023
    MessagingService
        MessagingServiceStart:87165(device6_intouch.log)
        Completed:87167(device6_intouch.log)
        ElapsedTime:6415
    DatabaseThreads

    ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceDiagnosticsServiceStart
    :87259(device6_intouch.log)
        Start:87259(device6_intouch.log)
        Completed:87265(device6_intouch.log)
        ElapsedTime:3552
    ConfigurationService
        ConfigurationServiceStart:87182(device6_intouch.log)
        Completed:87183(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:87194(device6_intouch.log)
        Completed:87195(device6_intouch.log)
        ElapsedTime:1096
    StateManager
        StateManagerStart:87206(device6_intouch.log)
        Completed:87207(device6_intouch.log)
        ElapsedTime:2958
    AVFeedbackService
        AVFeedbackServiceStart:87196(device6_intouch.log)
        Completed:87197(device6_intouch.log)
        ElapsedTime:55
    WATCHDOG
        Start:87259(device6_intouch.log)
        Completed:87261(device6_intouch.log)
        ElapsedTime:376
    DiagnosticsService
        Completed:87260(device6_intouch.log)
        ElapsedTime:158
    HealthMonitorService
```

```
        HealthMonitorServiceStart:87178(device6_intouch.log)
        Completed:87179(device6_intouch.log)
        ElapsedTime:152
    BellService
        BellServiceStart:87202(device6_intouch.log)
        Completed:87203(device6_intouch.log)
        ElapsedTime:2
    Persistence
        PersistenceStart:87180(device6_intouch.log)
        Completed:87181(device6_intouch.log)
        ElapsedTime:20159
    LandingPadService
        LandingPadServiceStart:87184(device6_intouch.log)
        Completed:87185(device6_intouch.log)
        ElapsedTime:2
    ThemingService
        ThemingServiceStart:87190(device6_intouch.log)
        Completed:87191(device6_intouch.log)
        ElapsedTime:1
    DeviceIOService
        DeviceIOServiceStart:87186(device6_intouch.log)
        Completed:87187(device6_intouch.log)
        ElapsedTime:28
    SoftLoadService
        Start:87259(device6_intouch.log)
        Completed:87263(device6_intouch.log)
        ElapsedTime:3323
    GateService
        GateServiceStart:87188(device6_intouch.log)
        Completed:87189(device6_intouch.log)
        ElapsedTime:415
    PortConfigurationService
        PortConfigurationServiceStart:87192(device6_intouch.log)
        Completed:87193(device6_intouch.log)
        ElapsedTime:37

=== Device boot ===
87836(device6_intouch.log): 2014-04-11 13:58:02 Boot Start
88009(device6_intouch.log): 2014-04-11 14:00:49 Boot Completed
    Boot Time: 167000ms

Services
    OfflineSmartviewService
        OfflineSmartviewServiceStart:87931(device6_intouch.log)
        Completed:87932(device6_intouch.log)
        ElapsedTime:22
    BiometricService
        BiometricServiceStart:87927(device6_intouch.log)
        Completed:87928(device6_intouch.log)
        ElapsedTime:669
    SoftLoadService
        Start:87982(device6_intouch.log)
        Completed:87989(device6_intouch.log)
        ElapsedTime:3776
    GateService
        GateServiceStart:87925(device6_intouch.log)
        Completed:87926(device6_intouch.log)
        ElapsedTime:1
    ReaderDataService
```

```
        ReaderDataServiceStart:87921(device6_intouch.log)
        Completed:87922(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:87857(device6_intouch.log)
        Completed:87858(device6_intouch.log)
        ElapsedTime:331
    DatabaseInitialize
        DatabaseInitializeStart:87859(device6_intouch.log)
        Completed:87885(device6_intouch.log)
        ElapsedTime:35524
    MessagingService
        MessagingServiceStart:87886(device6_intouch.log)
        Completed:87888(device6_intouch.log)
        ElapsedTime:6809
    DatabaseThreads

    ProtocolServiceWATCHDOGSoftLoadServiceDiagnosticsServiceDiagnosticsServiceStart
    :87982(device6_intouch.log)
        Start:87982(device6_intouch.log)
        Completed:87987(device6_intouch.log)
        ElapsedTime:3549
    ConfigurationService
        ConfigurationServiceStart:87905(device6_intouch.log)
        Completed:87906(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:87911(device6_intouch.log)
        Completed:87912(device6_intouch.log)
        ElapsedTime:1542
    StagingService
        StagingServiceStart:87915(device6_intouch.log)
        Completed:87916(device6_intouch.log)
        ElapsedTime:8060
    StateManager
        StateManagerStart:87929(device6_intouch.log)
        Completed:87930(device6_intouch.log)
        ElapsedTime:2798
    AVFeedbackService
        AVFeedbackServiceStart:87919(device6_intouch.log)
        Completed:87920(device6_intouch.log)
        ElapsedTime:37
    WATCHDOG
        Start:87982(device6_intouch.log)
        Completed:87984(device6_intouch.log)
        ElapsedTime:321
    DiagnosticsService
        Completed:87983(device6_intouch.log)
        ElapsedTime:229
    HealthMonitorService
        HealthMonitorServiceStart:87899(device6_intouch.log)
        Completed:87900(device6_intouch.log)
        ElapsedTime:166
    BellService
        BellServiceStart:87923(device6_intouch.log)
        Completed:87924(device6_intouch.log)
        ElapsedTime:203
    Persistence
        PersistenceStart:87901(device6_intouch.log)
```

```
        Completed:87904(device6_intouch.log)
        ElapsedTime:19998
    LandingPadService
        LandingPadServiceStart:87907(device6_intouch.log)
        Completed:87908(device6_intouch.log)
        ElapsedTime:2
    ThemingService
        ThemingServiceStart:87913(device6_intouch.log)
        Completed:87914(device6_intouch.log)
        ElapsedTime:1
    PortConfigurationService
        PortConfigurationServiceStart:87909(device6_intouch.log)
        Completed:87910(device6_intouch.log)
        ElapsedTime:39
    DeviceIOService
        DeviceIOServiceStart:87917(device6_intouch.log)
        Completed:87918(device6_intouch.log)
        ElapsedTime:24

=== Device boot ===
88983(device6_intouch.log): 2014-04-11 14:23:42 Boot Start
89155(device6_intouch.log): 2014-04-11 14:26:31 Boot Completed
    Boot Time: 169000ms

Services
    ThemingService
        ThemingServiceStart:89076(device6_intouch.log)
        Completed:89077(device6_intouch.log)
        ElapsedTime:0
    PortConfigurationService
        PortConfigurationServiceStart:89074(device6_intouch.log)
        Completed:89075(device6_intouch.log)
        ElapsedTime:63
    OfflineSmartviewService
        OfflineSmartviewServiceStart:89070(device6_intouch.log)
        Completed:89071(device6_intouch.log)
        ElapsedTime:22
    BiometricService
        BiometricServiceStart:89066(device6_intouch.log)
        Completed:89067(device6_intouch.log)
        ElapsedTime:688
    DatabaseInitialize
        DatabaseInitializeStart:89006(device6_intouch.log)
        Completed:89030(device6_intouch.log)
        ElapsedTime:35720
    MessagingService
        MessagingServiceStart:89033(device6_intouch.log)
        Completed:89035(device6_intouch.log)
        ElapsedTime:6065
    DatabaseThreads

    ProtocolServiceWATCHDOGSoftLoadServiceDiagnosticsServiceDiagnosticsServiceStart
    :89127(device6_intouch.log)
        Start:89127(device6_intouch.log)
        Completed:89134(device6_intouch.log)
        ElapsedTime:3717
    ConfigurationService
        ConfigurationServiceStart:89050(device6_intouch.log)
        Completed:89051(device6_intouch.log)
```

```
        ElapsedTime:0
    CacheService
        CacheServiceStart:89052(device6_intouch.log)
        Completed:89053(device6_intouch.log)
        ElapsedTime:987
    StateManager
        StateManagerStart:89068(device6_intouch.log)
        Completed:89069(device6_intouch.log)
        ElapsedTime:2935
    AVFeedbackService
        AVFeedbackServiceStart:89058(device6_intouch.log)
        Completed:89059(device6_intouch.log)
        ElapsedTime:45
    StagingService
        StagingServiceStart:89062(device6_intouch.log)
        Completed:89063(device6_intouch.log)
        ElapsedTime:7935
    WATCHDOG
        Start:89127(device6_intouch.log)
        Completed:89129(device6_intouch.log)
        ElapsedTime:258
    DiagnosticsService
        Completed:89128(device6_intouch.log)
        ElapsedTime:169
    HealthMonitorService
        HealthMonitorServiceStart:89046(device6_intouch.log)
        Completed:89047(device6_intouch.log)
        ElapsedTime:223
    BellService
        BellServiceStart:89072(device6_intouch.log)
        Completed:89073(device6_intouch.log)
        ElapsedTime:1
    Persistence
        PersistenceStart:89048(device6_intouch.log)
        Completed:89049(device6_intouch.log)
        ElapsedTime:20056
    DeviceIOService
        DeviceIOServiceStart:89054(device6_intouch.log)
        Completed:89055(device6_intouch.log)
        ElapsedTime:26
    SoftLoadService
        Start:89127(device6_intouch.log)
        Completed:89132(device6_intouch.log)
        ElapsedTime:3584
    GateService
        GateServiceStart:89056(device6_intouch.log)
        Completed:89057(device6_intouch.log)
        ElapsedTime:169
    LandingPadService
        LandingPadServiceStart:89060(device6_intouch.log)
        Completed:89061(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:89004(device6_intouch.log)
        Completed:89005(device6_intouch.log)
        ElapsedTime:311
    ReaderDataService
        ReaderDataServiceStart:89064(device6_intouch.log)
        Completed:89065(device6_intouch.log)
```

```
        ElapsedTime:2

=== Device boot ===
90112(device6_intouch.log): 2014-04-14 12:13:59 Boot Start
**** Incomplete boot ****

Services
    Logging
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    DatabaseInitialize
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    MessagingService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    HealthMonitorService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    Persistence
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    ConfigurationService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    LandingPadService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    PortConfigurationService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    CacheService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    ThemingService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    StagingService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    DeviceIOService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    BellService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
```

```
    GateService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    ReaderDataService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    BiometricService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    StateManager
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    OfflineSmartviewService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    AVFeedbackService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    DatabaseThreads
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    SoftLoadService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    WATCHDOG
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    ProtocolService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    DiagnosticsService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:

*** Services not successfully started: Logging, DatabaseInitialize,
    MessagingService, HealthMonitorService, Persistence,
    ConfigurationService, LandingPadService, PortConfigurationService,
    CacheService, ThemingService, StagingService, DeviceIOService,
    BellService, GateService, ReaderDataService, BiometricService,
    StateManager, OfflineSmartviewService, AVFeedbackService,
    DatabaseThreads, SoftLoadService, WATCHDOG, ProtocolService,
    DiagnosticsService


=== Device boot ===
90112(device6_intouch.log): 2014-04-14 12:13:59 Boot Start
**** Incomplete boot ****

```

```

=== Device boot ===
90135(device6_intouch.log): 2014-04-14 12:16:13 Boot Start
**** Incomplete boot ****

Services
    Logging
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    DatabaseInitialize
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    MessagingService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    HealthMonitorService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    Persistence
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    ConfigurationService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    LandingPadService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    PortConfigurationService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    CacheService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    ThemingService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    StagingService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    DeviceIOService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    BellService
         Start: Not started(device6_intouch.log)
         Completed: Not completed(device6_intouch.log)
         Elapsed Time:
    GateService
```

```
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    ReaderDataService
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    BiometricService
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    StateManager
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    OfflineSmartviewService
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    AVFeedbackService
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    DatabaseThreads
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    SoftLoadService
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    WATCHDOG
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    ProtocolService
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:
    DiagnosticsService
        Start: Not started(device6_intouch.log)
        Completed: Not completed(device6_intouch.log)
        Elapsed Time:

*** Services not successfully started: Logging, DatabaseInitialize,
   MessagingService, HealthMonitorService, Persistence,
   ConfigurationService, LandingPadService, PortConfigurationService,
   CacheService, ThemingService, StagingService, DeviceIOService,
   BellService, GateService, ReaderDataService, BiometricService,
   StateManager, OfflineSmartviewService, AVFeedbackService,
   DatabaseThreads, SoftLoadService, WATCHDOG, ProtocolService,
   DiagnosticsService


=== Device boot ===
90112(device6_intouch.log): 2014-04-14 12:13:59 Boot Start
**** Incomplete boot ****


```

```
=== Device boot ===
90135(device6_intouch.log): 2014-04-14 12:16:13 Boot Start
**** Incomplete boot ****


=== Device boot ===
90176(device6_intouch.log): 2014-04-14 12:18:44 Boot Start
90307(device6_intouch.log): 2014-04-14 12:21:25 Boot Completed
    Boot Time: 161000ms

Services
    ThemingService
        ThemingServiceStart:90268(device6_intouch.log)
        Completed:90269(device6_intouch.log)
        ElapsedTime:0
    PortConfigurationService
        PortConfigurationServiceStart:90266(device6_intouch.log)
        Completed:90267(device6_intouch.log)
        ElapsedTime:65
    OfflineSmartviewService
        OfflineSmartviewServiceStart:90262(device6_intouch.log)
        Completed:90263(device6_intouch.log)
        ElapsedTime:16
    BiometricService
        BiometricServiceStart:90258(device6_intouch.log)
        Completed:90259(device6_intouch.log)
        ElapsedTime:666
    DatabaseInitialize
        DatabaseInitializeStart:90198(device6_intouch.log)
        Completed:90224(device6_intouch.log)
        ElapsedTime:33067
    MessagingService
        MessagingServiceStart:90225(device6_intouch.log)
        Completed:90227(device6_intouch.log)
        ElapsedTime:3825
    DatabaseThreads

    DiagnosticsServiceProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceStart
    :90296(device6_intouch.log)
        Start:90296(device6_intouch.log)
        Completed:90303(device6_intouch.log)
        ElapsedTime:3564
    ConfigurationService
        ConfigurationServiceStart:90242(device6_intouch.log)
        Completed:90243(device6_intouch.log)
        ElapsedTime:0
    CacheService
        CacheServiceStart:90244(device6_intouch.log)
        Completed:90245(device6_intouch.log)
        ElapsedTime:981
    StateManager
        StateManagerStart:90260(device6_intouch.log)
        Completed:90261(device6_intouch.log)
        ElapsedTime:2657
    AVFeedbackService
        AVFeedbackServiceStart:90250(device6_intouch.log)
        Completed:90251(device6_intouch.log)
        ElapsedTime:49
    StagingService
```

```
        StagingServiceStart:90254(device6_intouch.log)
        Completed:90255(device6_intouch.log)
        ElapsedTime:7947
    WATCHDOG
        Start:90296(device6_intouch.log)
        Completed:90298(device6_intouch.log)
        ElapsedTime:557
    DiagnosticsService
        Completed:90297(device6_intouch.log)
        ElapsedTime:192
    HealthMonitorService
        HealthMonitorServiceStart:90238(device6_intouch.log)
        Completed:90239(device6_intouch.log)
        ElapsedTime:205
    BellService
        BellServiceStart:90264(device6_intouch.log)
        Completed:90265(device6_intouch.log)
        ElapsedTime:2
    Persistence
        PersistenceStart:90240(device6_intouch.log)
        Completed:90241(device6_intouch.log)
        ElapsedTime:18401
    DeviceIOService
        DeviceIOServiceStart:90246(device6_intouch.log)
        Completed:90247(device6_intouch.log)
        ElapsedTime:27
    SoftLoadService
        Start:90296(device6_intouch.log)
        Completed:90302(device6_intouch.log)
        ElapsedTime:3501
    GateService
        GateServiceStart:90248(device6_intouch.log)
        Completed:90249(device6_intouch.log)
        ElapsedTime:164
    LandingPadService
        LandingPadServiceStart:90252(device6_intouch.log)
        Completed:90253(device6_intouch.log)
        ElapsedTime:2
    Logging
        LoggingStart:90196(device6_intouch.log)
        Completed:90197(device6_intouch.log)
        ElapsedTime:289
    ReaderDataService
        ReaderDataServiceStart:90256(device6_intouch.log)
        Completed:90257(device6_intouch.log)
        ElapsedTime:2
```

8.5 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework -lboost_date_time
# Your .hpp files
DEPS = Kronos.hpp
# Your compiled .o files
OBJECTS = Kronos.o
# The name of your program
PROGRAM = ps7
```

```makefile

.PHONY: all clean lint

all: $(PROGRAM) Kronos.a

# Wildcard recipe to make .o files from corresponding .cpp file
%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

Kronos.a: Kronos.o
    ar rcs Kronos.a Kronos.o

clean:
    rm *.o $(PROGRAM) *.a

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

```cpp
// Copyright Camden Andersson 2024

#include <iostream>
#include <string>
#include <regex>
#include "Kronos.hpp"

int main(int argc, char* argv[]) {
    std::string filename = argv[1];
    kronos k(filename);
    k.makeBootupLog();
    // k.printBootupLog();
    k.makeSummaryLog();
    k.printSummaryLog();

    return 0;
}
```

```cpp
// Copyright Camden Andersson 2024
#pragma once

#ifndef KRONOS_HPP
#define KRONOS_HPP

#include <iostream>
#include <string>
#include <regex>
#include <boost/date_time.hpp>

class kronos {
 public:
    explicit kronos(const std::string& filename);
    void makeBootupLog();
    void makeSummaryLog();
    void printBootupLog();
    void printSummaryLog();
    std::string cleanBonus(const std::string& input);
    std::string createServiceStatusString(const std::string& filename);
 private:
```

```
    int numLinesScanned;    // the number of lines read in "filename"
    int numInitiated;       // the number of services initiated
    int numSuccess;         // the number of successful services running
    std::string logFilename;  // the filename of the log to be read
    std::string logText;    // the text from "filename"
    std::string logBoot;    // just the boot text
    std::string logFull;    // the full text for the bonus portion
};

#endif /* KRONOS_HPP */
```

```
// Copyright Camden Andersson 2024
#include "Kronos.hpp"
#include <unordered_map>

// This function reads the file and returns it as text
std::string read_file_to_string(const std::string& filename) {
    std::ifstream file(filename);
    if (!file.is_open()) {
        std::cerr << "Failed to open file: " << filename << std::endl;
        return "";
    }
    std::string file_contents;
    file.seekg(0, std::ios::end);
    file_contents.reserve(file.tellg());
    file.seekg(0, std::ios::beg);

    file_contents.assign((std::istreambuf_iterator<char>(file)),
        std::istreambuf_iterator<char>());

    return file_contents;
}

void write_string_to_file(const std::string& str, const std::string&
   filename) {
    std::ofstream file(filename);
    if (file.is_open()) {
        file << str;
        file.close();
        std::cout << "String written to file successfully." << std::endl;
    } else {
        std::cerr << "Unable to open file." << std::endl;
    }
}


// Constructor
kronos::kronos(const std::string& filename) {
    logFilename = filename;
    // Open the file that contains the text & display it
    std::string rawText = read_file_to_string(filename);
    if (!rawText.empty()) {
        logText = rawText;
    } else {
        std::cout << "File not found.\n" << std::endl;
        logText = "";
    }
    // Finish initializing the all the class members
    numLinesScanned = 0;
    numInitiated = 0;
```

```
    numSuccess = 0;
    logBoot = "";
    logFull = "";
}

void kronos::makeBootupLog() {
    std::string logShort = "";
    std::string logLong = "";
    std::string bootStartTime;
    std::string bootEndTime;
    std::regex startRegex("\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}: "
    ".*server started");
    std::regex endRegex("\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}: "
    ".*AbstractConnector:Started .*");
    std::smatch match;

    std::string line = "";
    int lineNumStart = 0;
    int lineNumEnd = 0;
    int lineNumber = 0;
    bool bootStarted = false;
    std::string bonus;
    std::istringstream iss(logText);
    while (std::getline(iss, line)) {
        lineNumber++;

        // Look for the START message
        if (std::regex_search(line, match, startRegex)) {
            // See if this is a failure
            if (!bootStarted) {
                bootStarted = true;
            } else {
                // bootStarted was true, hence this is an error
                logShort += "**** Incomplete boot ****\n\n";
                std::string sError = kronos::createServiceStatusString(
   logFilename);
                logFull += logShort + sError;
                bonus = "";
            }

            // Get the date and time
            std::regex pattern(R"((\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}))");
            std::smatch matches;
            if (std::regex_search(line, matches, pattern)) {
                // Print the matched date and time
                bootStartTime = matches[1];
            }
            numInitiated++;

            // Build the log message
            logShort += "\n=== Device boot ===\n";
            logShort += std::to_string(lineNumber) +
            "(" + logFilename + "): " + bootStartTime +  " Boot Start\n";
        }

        ////////////////////////////////////////////////////////////////
        // BONUS - log other services
        // Start Services
        if (bootStarted) {
```

```
            std::string service;
            std::regex patternService("Starting Service\\.\\s*(\\w+) (\\d+\\.\\d+)");
            std::smatch matchService;
            if (std::regex_search(line, matchService, patternService)) {
                if (matchService.size() > 2) {
                    service = matchService[1];
                    bonus += "\t" + service;
                    lineNumStart = lineNumber;
                }
            }

            // Complete service
            std::regex expr("Service\\s+started\\s+successfully\\.\\s+([A-Za-z0-9]+)"
            "\\s+([A-Za-z0-9.]+)\\s+\\(([0-9]+)\\s+ms\\)");
            std::smatch match;
            std::string sMs;
            std::string sService;
            if (std::regex_match(line, match, expr)) {
                sService = match[1];  // TO DO: check sService!!
                sMs = match[3];
                lineNumEnd = lineNumber;
                bonus += "\t\t" + sService +
                " - Start: " + std::to_string(lineNumStart) + "(" + logFilename + ")\n";
                bonus += "\t\t" + sService +
                " - Completed: " + std::to_string(lineNumEnd) + "(" + logFilename + ")\n";
                bonus += "\t\t" + sService + " - Elapsed Time: " + sMs + "\n";
            }
        }
        ////////////////////////////////////////////////////////////////

        // Look for the END message
        if (std::regex_search(line, match, endRegex)) {
            std::string s;
            // Get the date and time
            std::regex pattern(R"((\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}))");
            std::smatch matches;
            if (std::regex_search(line, matches, pattern)) {
                // Print the matched date and time
                bootEndTime = matches[1];
            }
            s = std::to_string(lineNumber) +
            "(" + logFilename + "): " + bootEndTime + " Boot Completed\n";
            logShort += s;
            numSuccess++;

            // Calculate elapsed time
            boost::posix_time::ptime startTime = boost::posix_time::time_from_string(bootStartTime);
            boost::posix_time::ptime endTime = boost::posix_time::time_from_string(bootEndTime);
            boost::posix_time::time_duration diff = endTime - startTime;
            s = "\tBoot Time: " + std::to_string(diff.total_milliseconds()) + "ms\n\n";
            logShort += s;
```

```

            // Put logShort into the logBoot result
            logBoot += logShort;

            // You've found the COMPLETION
            std::string cleanedBonus = cleanBonus(bonus);
            logFull += logShort + "Services\n" + cleanedBonus;
            bonus = "";  // clear it
            logShort = "";  // clear it
            // logLong += logShort + bonus;
            bootStarted = false;
        }
    }

    numLinesScanned = lineNumber;
}

std::string kronos::cleanBonus(const std::string& input) {
    std::istringstream iss(input);
    std::unordered_map<std::string, std::vector<std::string>> groupedLines;

    // Parse the input and group lines by the first word
    std::string line;
    while (std::getline(iss, line)) {
        // Trim leading tabs and spaces
        line.erase(line.begin(), std::find_if(line.begin(), line.end(), [](unsigned char ch) {
            return !std::isspace(ch);
            }));

        std::istringstream lineStream(line);
        std::string firstWord;
        lineStream >> firstWord;

        // Remove the first word and the '-' if present
        line = line.substr(firstWord.length());
        line.erase(std::remove_if(line.begin(), line.end(), [](unsigned char ch) {
            return std::isspace(ch) || ch == '-';
            }), line.end());

        // Store the line in the appropriate group
        groupedLines[firstWord].push_back(line);
    }

    // Format the output
    std::ostringstream oss;
    for (const auto& pair : groupedLines) {
        oss << "\t" << pair.first << std::endl;
        for (const auto& line : pair.second) {
            oss << "\t\t" << line << std::endl;
        }
        // oss << std::endl;  // Add a blank line between groups
    }

    return oss.str();
}

std::string kronos::createServiceStatusString(const std::string & filename)
```

```
    {
        std::ostringstream oss;
        oss << "Services\n";
        std::string services[] = {
            "Logging", "DatabaseInitialize", "MessagingService", "HealthMonitorService",
            "Persistence", "ConfigurationService", "LandingPadService", "PortConfigurationService",
            "CacheService", "ThemingService", "StagingService", "DeviceIOService", "BellService",
            "GateService", "ReaderDataService", "BiometricService",
            "StateManager", "OfflineSmartviewService",
            "AVFeedbackService",
            "DatabaseThreads", "SoftLoadService",
            "WATCHDOG", "ProtocolService", "DiagnosticsService"
        };

        for (const auto& service : services) {
            oss << "\t" << service << "\n";
            oss << "\t\t Start: Not started(" << filename << ")\n";
            oss << "\t\t Completed: Not completed(" << filename << ")\n";
            oss << "\t\t Elapsed Time: \n";
        }

        oss << "\n*** Services not successfully started: Logging, "
        "DatabaseInitialize, MessagingService, HealthMonitorService, "
        "Persistence, ConfigurationService, LandingPadService, "
        "PortConfigurationService, CacheService, ThemingService, "
        "StagingService, DeviceIOService, BellService, GateService, "
        "ReaderDataService, BiometricService, StateManager, OfflineSmartviewService, "
        "AVFeedbackService, DatabaseThreads, SoftLoadService, WATCHDOG, ProtocolService, "
        "DiagnosticsService\n\n";

        std::string result = oss.str();
        size_t pos = result.find("device5_intouch.log");
        if (pos != std::string::npos) {
            result.replace(pos, std::strlen("device5_intouch.log"), filename);
        }

        return result;
}

void kronos::makeSummaryLog() {
    // Start the header, for the bonus points
    std::string sHeader;
    sHeader = "Device Boot Report\n\n";
    sHeader += "InTouch log file: " + logFilename + "\n";
    sHeader += "Lines Scanned: " + std::to_string(numLinesScanned) + "\n\n";
    sHeader += "Device boot count: initiated = "
    + std::to_string(numInitiated) + ", completed: " + std::to_string(numSuccess);

    // Make the full log
    logFull = sHeader + "\n\n" + logFull;
}

```

```
void kronos::printBootupLog() {
    // std::cout << logBoot << std::endl;

    std::string filename = logFilename + ".rpt";
    write_string_to_file(logBoot, filename);
}

void kronos::printSummaryLog() {
    // std::cout << logFull << std::endl;

    std::string filename = logFilename + ".rpt";
    write_string_to_file(logFull, filename);
}
```