RAID Project

Submission Date: 9th of December 11:59 PM

1 Introduction

1.1 Submission

Submit a **folder** that is **only** containing your Java source files (*.java) to the course's Homework folder.

Full path: **F:\COURSES\UGRADS\COMP130\Homework**

Note: COMP 131 students, please submit to the folder under the COMP 130 directory.

Please use the following naming convention for the submitted folders:

YourPSLetter_CourseCode_Surname_Name_HWNumber_Semester

Example folder names:

- **PSA_COMP130_Surname_Name_HW4_F18**
- **PSB_COMP131_Surname_Name_HW4_F18**

Additional notes:

- Using Java libraries and code which are not covered in lectures and problem sessions are **NOT ALLOWED**, failing to do so may be **penalized**.
- Using the naming convention properly is important, failing to do so may be **penalized**.
- **Do not** use Turkish characters when naming files or folders.
- Submissions with unidentifiable names will be **disregarded** completely. (ex. "homework1", "project" etc.)
- Please write your name into the Java source file where it is asked for.
- If you are resubmitting to update your solution, simply append **v#** where # denotes the resubmission version. (i.e. **v2**)

1.2 Academic Honesty

Koç University's *Statement on Academic Honesty* holds for all the homeworks given in this course. Failing to comply with the statement will be penalized accordingly. If you are unsure whether your action violates the code of conduct, please consult your instructor.

1.3 Aim of the Project

The aim of the project is to allow you to practice knowledge on Strings, Arrays and Image Manipulation. In this project, you are asked to develop a Java program which reads an image data and stores it using arrays with which then you will simulate RAID 0, 1 and 10 configuration.

1.4 Given Code

In the code given to you, you will see that there are parts already implemented in the project. **Do not** change anything in the code if it is indicated to you with a comment. The code given to you has something called **JavaDoc** comments above all the methods. These comments allow you to view various information about the method when you mouse over the name of the method. Below are the methods given to you in the code with their explanation.

1.4.1 Given Methods

Your program shall implement and call the following methods:

- **public GImage createRaidImage(int raid)** - given the RAID configuration, the method creates a new GImage object using the specified RAID configuration.
- **public void failRaid(int raid)** - given the RAID configuration, the method simulates the failing of a disc in a RAID configuration.
- **public void createRaid0Discs(GImage image)** - given an image, the method simulates saving of the image data on RAID0 discs which consists of 2 discs (disc1 and disc2)
- **public void createRaid1Discs(GImage image)** - given an image, the method simulates saving of the image data on RAID1 discs which consists of 2 discs (disc1 and disc2)
- **public void createRaid10Discs(GImage image)** - given an image, the method simulates saving of the image data on RAID10 discs which consists of 4 discs (disc1, disc2, disc3 and disc4).
- **public void writeImageFile (int[][] pixels, String raidType)** - given 2D array of image pixels constructed after applying a RAID configuration and the name of the RAID configuration, the method shall save the image pixels into appropriate file (e.g. RAID1.txt file for image pixels in RAID1 configuration).

You should implement the details of the above given methods with the given signature. You should also use (call) them in appropriate methods. You will be **penalized** in case you fail to implement and call these methods.

You can create additional methods. However, you shouldn't replace the above methods with the new ones.

The Java project given to you also contains two images that you can work with.

1.4.2 Given Constants

Constants are given at the bottom of the project. All constants provide **JavaDoc** comments above them. Please read these to understand the purpose of using them. **Do not** use another variable or a static value for something if there is a constant variable defined for that purpose.

1.5 Further Questions

For further questions **about the project** you may contact **Ayça Tüzmen** at [atuzmen@ku.edu.tr]. Note that it may take up to 24 hours before you receive a response so please ask your questions **before** it is too late. No questions will be answered when there is **less than two days** left for the submission.

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10

Figure 1: Image Pixels

2 Project Tasks

The project consists of 5 parts. In part 1, the program shall read an image and convert its data into a multidimensional array. In part 2, the program shall simulate saving the contents of the multidimensional array to imaginary discs using RAID 0, 1 and 10 algorithms. In Part 3, the program shall simulate the scenario where an imaginary disk crashes and the program shall corrupt the contents of a disc in the RAID configuration. In Part 4, the program shall reconstruct a new image using the discs in the RAID configuration. In Part 5, the program shall display the original image and reconstructed new image from RAID discs.

2.1 Part 1 - Creating 2D Pixel Data from an Image (5 points)

In Part 1, the program shall ask the user to enter the name of an image file, and convert the pixel values of the image into a two-dimensional array. The java project given to you contains two images. The user can select the name of one of them. The program shall prompt user to re-enter until a valid file name is entered. Once the user enters an image name that exists in the Java project, the program shall create a two dimensional array consisting of int pixel values. Assume that the pixel values of an image is represented as shown in Figure 1.
Hint: Use getPixelArray() method.

2.2 Part 2 - Storing in RAID Configuration (40 points)

The program shall display a menu in the console and allow the user to select the RAID configuration that will be used to store the image data. The menu shall display (Figure 2):

```
This program simulates the saving of an image on Raid Volumes
Specify the name of the image file:HW.png
Select 1, 0, or 10: 0
```

Figure 2: Menu

2.2.1 RAID 0 - Strip Data in Two Discs

The program shall simulate saving the given image data using *RAID 0*.

RAID 0 allows the data in the RAID volume to be striped across the array's members. Striping divides data into units and distributes those units across the members without creating data redundancy, but improving read/write performance (Figure 3). In this part, the pixel values captured in Part 1, needs to be converted from 2D array (int [][]) into **two separate one dimensional (int[])**

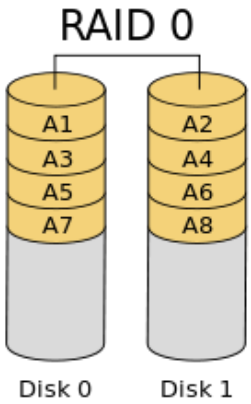

Figure 3: RAID0 Configuration.

arrays. Each one dimensional array will simulate a single disc. Your program should **strip pixel values** from the 2D array into **two** one dimensional arrays.

For example, assume the pixel values of the image data is stored in the 2D array as such:

int [][] pixel =
{ {-1,-4,-9213102,-10857935,-10857936,-10857937,-10857938},
{-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933} }

The program shall store these pixel values as such:
int [] disc1 =
{-1,-92453102,-10857936,-10857938,-1,-12357939,-10832932}
int [] disc2 =
{-4,-10857935,-10857937,-5,-92453102,,-10567931,-10857933}

2.2.2 RAID1 - Mirroring Data in Two Discs

The program shall simulate saving the given image data using *RAID1*.

RAID 1 allows the data in RAID volume to be **mirrored** across the RAID array's members. Mirroring is the term used to describe the key feature of RAID1, which writes duplicate data to each member; therefore, creating data redundancy and increasing fault tolerance (Figure 4).

The pixel values captured in Part 1, needs to be converted from 2D array (int [][]) into **two separate one dimensional (int[]) arrays**. Each one dimensional array will simulate a single disc. Your program should **mirror pixel values** from the 2D array into **two** one dimensional arrays.

For example, assume the pixel values the image data is stored in the 2D array as such:
int [][] pixel =
{ {-1,-4,-9213102,-10857935,-10857936,-10857937,-10857938},
{-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933} }

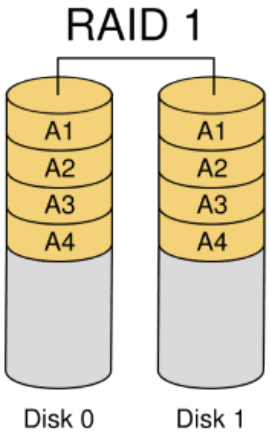

Figure 4: RAID1 Configuration.

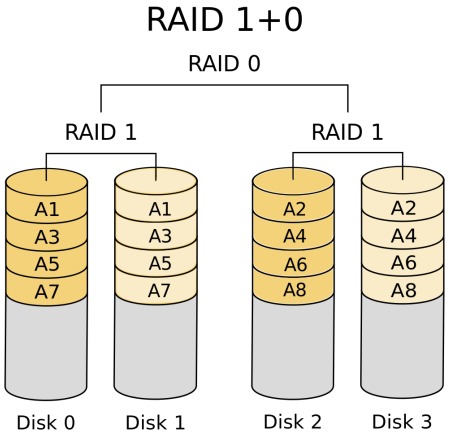

Figure 5: RAID10 Configuration.

The program shall store these pixel values as such:
int [] disc1 =
{1,-4,-9213102,-10857935,-10857936,-10857937,-10857938,-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933}
int [] disc2 =
{1,-4,-9213102,-10857935,-10857936,-10857937,-10857938,-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933}

2.2.3 RAID 10

The program shall simulate saving of the given image data using *RAID10*.

Using *RAID10*, information is striped across a two disk arrays for system performance. Each of the discs in the arrays has a mirror for fault tolerance. RAID10 provides the performance benefits of RAID 0 and the redundancy of RAID 1. However, it requires double the hard drives (Figure 5).

The pixel values captured in Part 1, needs to be converted from 2D array (int [][]) into **four separate one dimensional (int[]) arrays**. Each one dimensional array will simulate a single disc. Your program should **mirror and strip pixel values** from the 2D array into **four one dimensional arrays**.

For example, assume the pixel values of the image data is stored in the 2D array as such:
int [][] pixel = {
{-1,-4,-9213102,-10857935,-10857936,-10857937,-10857938},
{-5,-1,-92453102,-12357939,-10567931,-10832932,-10857933} }

The program shall store these pixel values as such:
int [] disc1 =
{-1,-92453102,-10857936,-10857938,-1,-12357939,-10832932 }
int [] disc2 =
{-1,-92453102,-10857936,-10857938,-1,-12357939,-10832932 }
int [] disc3 =
{-4,-10857935,-10857937,-5,-92453102,,-10567931,-10857933}
int [] disc4 =
{-4,-10857935,-10857937,-5,-92453102,,-10567931,-10857933}

2.3 Part 3 - Failing a Disc (10 points)

The program shall simulate the failure of a single disc in a RAID configuration. It should **randomly decide** which disc to fail. Once decided on the disc, it will randomly **corrupt** the content of the failing disc.

If the user selected **RAID0 or RAID1** as the configuration, your program should fail **one out of the two discs** in RAID0 or RAID1 configuration (i.e. disc1 or disc2). For example, if disc1 is chosen, the program shall corrupt the contents of disc1.

If the user selected RAID10 as the configuration, the program shall randomly fail only **one the discs out of four discs** in RAID 10. For example, if disc4 is chosen, the program shall corrupt the contents of disc4 only.

A disc will be corrupted by replacing the pixel value at every **5th** pixel in the disc by a **BLACK** and **Transparent** color pixel.

Hint: Use **GImage.createRGBPixel(0, 0, 0, 0)** method.

For example, if the pixel values on disc1 are as such before failing:
int [] disc1 =
{ {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-9211079,-1,-1,-1,-1,-1,-1},
{-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-8096967,-6515638,-6515638,-1,-1,-1,-1,-1} }

After failure, the corrupted pixel values on disc1 shall look as such:
int [] disc1AfterFailure =
{ {-16777216,-1,-1,-1,-1,-16777216,-1,-1,-1,-1,-16777216,-9211079,-1,-1,-1,-16777216,-1,-1},
{-1,-1,-16777216,-1,-1,-1,-1,-16777216,-1,-1,-8096967,-6515638,-16777216,-1,-1,-1,-1,-16777216} }

```
0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -9211079 -1 -1 -1 0 -1 -1
-1 -1 0 -1 -1 -1 -1 0 -1 -1 -8096967 -6515638 0 -1 -1 -1 -1 0
-1 -1 -1 -1 0 -1 -1 -1 -1 0 -4868782 -4868782 -8094902 -1 0 -1 -1 -1
-1 0 -1 -1 -1 -1 0 -1 -9211079 -3750301 -3750301 0 -8094902 -1 -1 -1 0 -1
-1 -1 -1 0 -1 -1 -1 -10857935 0 -1052813 -1052813 -2699661 -8094902 0 -1 -1 -1 -1
0 -1 -1 -1 -1 0 -1 -6515638 -1052813 -2175340 0 -1052813 -6515638 -1 -1 0 -1 -1
-1 -1 0 -1 -1 -1 -10263751 0 -124 -124 -124 -124 0 -1 -1 -1 -1 0
-1 -1 -1 -1 0 -10857935 -5923766 -124 -124 0 -1052813 -124 -4344477 -5923766 0 -1 -1 -1
-1 0 -1 -1 -1 -9211079 0 -124 -1052813 -2148 -124 0 -2699661 -9211079 -1 -1 0 -1
-1 -1 -1 0 -10263759 -5923766 -124 -2148 0 -1583460 -2148 -124 -2699661 0 -1 -1 -1 -1
0 -1 -1 -1 -10263751 0 -124 -124 -2148 -528467 0 -1052813 -1052813 -8094902 -1 0 -1 -1
-1 -1 0 -1 -8096967 -2699661 -124 0 -2148 -1583460 -2148 -124 0 -5923766 -1 -1 -1 0
-1 -1 -1 -1 0 -2699661 -124 -124 -124 0 -124 -124 -124 -4872870 0 -1 -1 -1
-1 0 -1 -1 -7044022 -2699661 0 -2148 -1052813 -124 -1052813 0 -2148 -4872870 -1 -1 0 -1
-1 -1 -1 0 -8094902 -2699661 -124 -124 0 -1052813 -2148 -124 -1052813 0 -1 -1 -1 -1
0 -1 -1 -1 -8094902 0 -1052813 -1052813 -2148 -124 0 -124 -1052813 -6517670 -1 0 -1 -1
-1 -1 0 -1 -8096967 -4344477 -1052813 0 -124 -124 -1052813 -2148 0 -8096967 -1 -1 -1 0
-1 -1 -1 -1 0 -6515638 -2699661 -2148 -1052813 0 -124 -1052813 -4872870 -10263759 0 -1 -1 -1
-1 0 -1 -1 -1 -10263751 0 -1052813 -124 -1052813 -1052813 0 -8094902 -1 -1 -1 0 -1
-1 -1 -11908575 0 -10263751 -10857935 -8096967 -4344477 0 -2699661 -4872870 -8096967 -11384287 0 -8094894 -11908575 -1 -1
0 -13027048 -8094894 -528442 -1057099 0 -5399437 -9211079 -4344477 -5925781 0 -5399437 -3754620 -1057099 -528442 0 -13027048 -1
-1 -13553384 0 -1057099 -528451 -526394 -2098 0 -5925781 -5399437 -1581395 -2098 0 -528451 -1057099 -8094894 -13553384 0
-1 -13027039 -3754620 -1057099 0 -528451 -528442 -3754620 -5399437 0 -1054778 -526394 -528442 -1057099 0 -3754620 -13027039 -1
-1 0 -1581395 -1581395 -528467 -526394 0 -528451 -1581395 -1057099 -528451 0 -530507 -528467 -1581395 -1581395 0 -1
-1 -10263751 -528451 0 -1057107 -1057099 -1057099 -1581395 0 -1581395 -1581395 -1581395 -1581395 0 -1581395 -1581395 -10857935 -1
0 -10263751 -528451 -1057107 -1057099 0 -1057099 -1057099 -1057099 -1057107 0 -1057099 -1057107 -1583460 -1581395 0 -10857935 -1
-1 -9213102 0 -528451 -526394 -528442 -526394 0 -530507 -526394 -528442 -1057099 0 -1581395 -1057099 -2175340 -10857935 0
-1 -9213102 -2098 -2098 0 -2098 -2098 -526394 -526394 0 -528442 -528451 -1057107 -2175340 0 -2175323 -11908551 -1
-1 0 -2098 -2098 -2098 -2098 0 -2098 -526394 -2098 -528442 0 -2175323 -2175340 -2175323 -2175340 0 -1
-1 -11908551 -2098 0 -2098 -2098 -2098 -2098 0 -526394 -528442 -528451 -1581395 0 -2703980 -3754620 -13553367 -1
0 -13553367 -528442 -2098 -2098 0 -2098 -2098 -2098 -528442 0 -1057099 -1581395 -2703980 -3754620 0 -14079711 -1
-1 -14079711 0 -2098 -2098 -2098 -2098 0 -526394 -528442 -528451 -1057099 0 -3754620 -3754620 -4348549 -14606064 0
-1 -14606056 -1581395 -2098 0 -2098 -2098 -526394 -528442 0 -1057099 -1581395 -2175323 -3754620 0 -5925781 -14081776 -1
-1 0 -5399437 -2098 -2098 -2098 0 -528442 -528442 -528451 -1057099 0 -2703980 -3754620 -4348549 -11908575 0 -1
-1 -1 -9211021 0 -2098 -2098 -526394 -528442 0 -1057099 -1057107 -2177371 -2703980 0 -5399437 -15198192 -1 -1
0 -1 -9211021 -2098 -2098 0 -526394 -528442 -528442 -528451 0 -2703980 -3754620 -3754620 -5399437 0 -1 -1
-1 -1 0 -2098 -2098 -2098 -526394 0 -530507 -1057099 -1581395 -2177371 0 -3754620 -5399437 -14606056 -1 0
-1 -1 -8094894 -2098 0 -2098 -528442 -528442 -528451 0 -2175323 -2177371 -3754620 -3754620 0 -15198192 -1 -1
-1 0 -8094894 -2098 -2098 -2098 0 -528442 -528451 -1057099 -2175323 0 -3754620 -3754620 -5399437 -15198192 0 -1
-1 -1 -8094894 0 -2098 -2098 -528442 -1054778 0 -1057099 -1581395 -2177371 -2703980 0 -5925781 -15198192 -1 -1
0 -1 -9213102 -2098 -2098 0 -526394 -528442 -528451 -1057099 0 -2177371 -3754620 -3754620 -5925781 0 -1 -1
-1 -1 0 -2098 -2098 -2098 -528442 0 -528451 -1057099 -2175323 -2703980 0 -3754620 -5925781 -15198192 -1 0
-1 -1 -9213102 -2098 0 -2098 -528442 -528442 -528451 0 -2175323 -2177371 -3754620 -3754620 0 -15198192 -1 -1
-1 0 -8094894 -2098 -2098 -2098 0 -528442 -530507 -1581395 -2177371 0 -3754620 -4348549 -5925781 -15198192 0 -1
-1 -1 -8094894 0 -2098 -2098 -528442 -528442 0 -1057107 -2175323 -2177371 -3754620 0 -5925781 -15198192 -1 -1
0 -1 -9213102 -2098 -2098 0 -528442 -528451 -530507 -1057099 0 -2177371 -3754620 -4348549 -5925781 0 -1 -1
-1 -1 0 -2098 -2098 -2098 -528442 0 -1057099 -1057099 -2175323 -2703980 0 -4348549 -5925781 -14608368 -1 0
-1 -1 -8094894 -2098 0 -2098 -528442 -528451 -1057099 0 -2175323 -3754620 -3754620 -4348549 0 -15198192 -1 -1
```

Figure 6: Sample RAID1.txt.

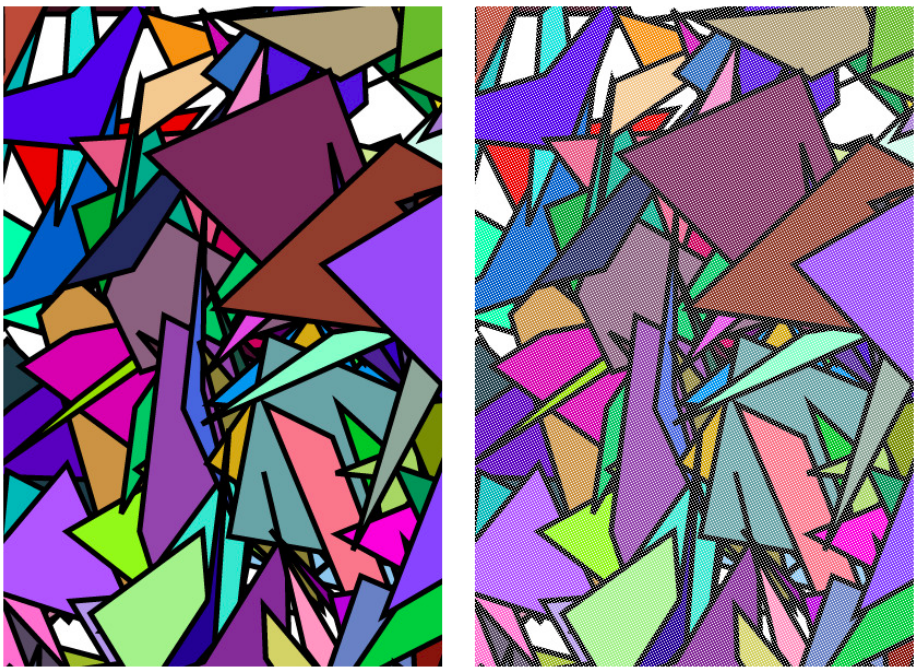

Figure 7: RAID0 Output.

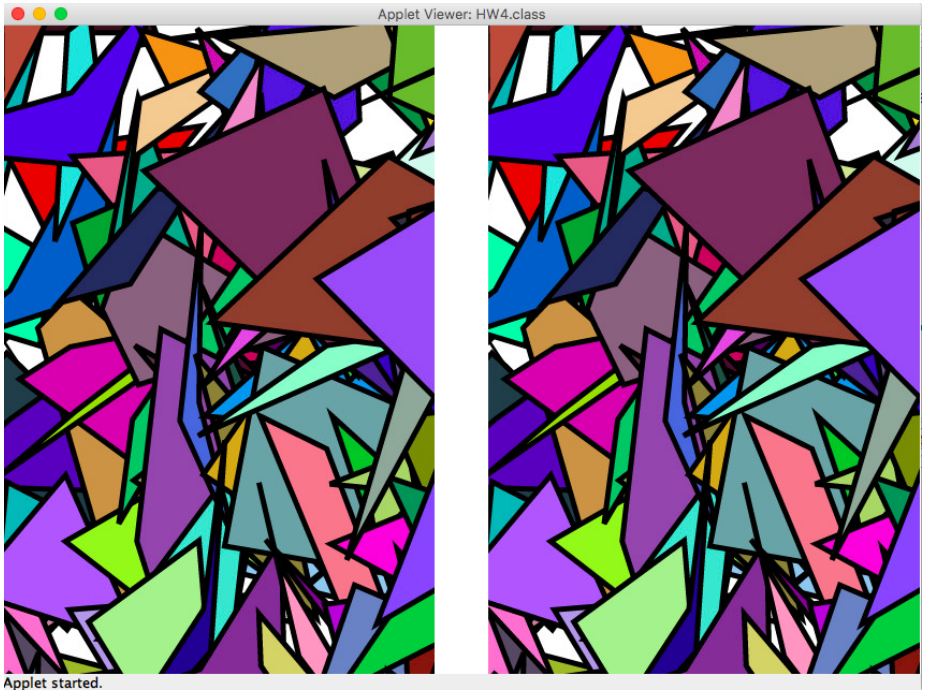

Figure 8: RAID1 Output and RAID10.

2.4 Part 4 - Creating a New Image from RAID Discs and Saving Image Data (40 points)

Your program shall construct a new image from the pixel values in the discs used in a RAID Configuration.

- If the user selected RAID0, the new image should be created using the pixel values in **two discs (both failing and non-failing discs, meaning disc1 and disc2)**.
- If the user selected RAID1, the new image should be created using **one of the discs (non-failing disc)** in RAID1.
- If the user selected RAID10, the new image should be created using **two discs (non-failing discs)** in RAID10.

The program shall save the new image construct on a file. The pixel values of the newly constructed image shall be saved into a .txt file (Figure 6).

- Create and save into **RAID0.txt** file the pixel values of the RAID0 image.
- Create and save into **RAID1.txt** file the pixel values of the RAID1 image.
- Create and save into **RAID10.txt** file the pixel values of the RAID10 image.

Hint: Use **BufferedReader rd = new BufferedReader(new FileReader(fileName))** syntax.

2.5 Part 5 - Displaying the original and recovered image (5 points)

Your program should display the original and the recovered image from RAID configuration on the canvas next to each other. A sample output is shown in (Figure 7 and Figure 8).

2.6 End of Project

Your project ends here. You may continue to tinker with the code to implement any desired features and discuss them with your section leader. However, **do not** include any additional features that you implement after this point in to your submission.