CS3460: Data Structures Fall 2021
Halloween Midterm
Due Date: Thursday, October 7; 23:00 hrs
Total Points: 25

Honor Code: The following document is an individual exam. As such, this exam should be solved individually, without any discussion, exchange of electronic or hard documents, or any other malpractice that constitutes to cheating in an academic institution. Even a pleasant conversation that involves revealing how much of the test one completes constitutes a violation. If you are unsure what constitutes plagiarism, please refer to the following guidelines set forth by the Appalachian State University `http://academicintegrity.appstate.edu/`, or consult with your instructor. By continuing to participate in this mid-term, you implicitly agree to uphold the honor code. Failure to uphold the honor code will result in an **F grade** for the entire course.

Please write well-tested Java programs to solve the following problems. You are allowed to use any code that is posted on ASULearn, or code that you have written for any of the homework problems. You are not allowed to search for solutions in the Internet. You may consider to logically split your solutions into several components, and solve each of those components in Java. For example, to use a Hash Table, you may have a class that implements a single Node structure, and a class that implements a hash table that resolves collisions through chaining. For any of the problems, you are allowed to create multiple files that cater to your implementation, but please submit all the necessary Java code for a particular solution, so that they can be run simply by compiling and executing in a Java environment. If you are creating a `Node` class for linked lists and hash tables, consider naming them `ListNode` and `HashNode` respectively to avoid confusion during compilation.

Due to popular requests, especially by a character named Henry Jones, this exam is Halloween themed. I hope you all are not horror-stuck by the problems.

Good Luck!

1. **Horror Playlist (15 points):** Olivia Hess has created a horror-filled video playlist on the new video service AppTube. The playlist contains many different types of videos. For example, it could contain all videos about the works of Stephen King, or the videos could have popular horror sub-genres like zombies, monsters and aliens. There could be many videos of the same type. However, each video is associated with only one type.

   For her Halloween, she decides that time has finally come for her to watch these videos. She wants to watch all videos of the same type at the same time. She can click on any video in the playlist to start watching that video. If she is done watching that video, then that video is automatically removed from the playlist. Moreover, AppTube automatically plays the next video in the playlist only if the next video is of the same type as the one that was just watched. Otherwise she needs to click again on some other video to start watching that video. The playlist does not loop, so after watching the last video on the playlist, she may have to click on another video to start watching again. In this problem, you are given a description of the videos in the playlist and a sequence of video types that indicates the order in which to watch the videos. Your task is to compute the total number of clicks needed to watch all the videos on the playlist. Note that you may not reorder the playlist.

   The input consists of two integers n and k on the first line, where n is the total number of videos in the playlist, and k is the number of different types of videos. The second line

contains n integers $p_1, \ldots, p_n$, where p_i is type of the i^{th} video on the playlist. All p_is are between 1 and k. On the third line, there are k integers $v_1, \ldots, v_k$ that provide the order in which to watch the videos on the playlist. Please output a single number that indicates the total number of clicks needed to watch the entire playlist. A sample input and output is provided below.

```
Sample Input (see horrorplaylist.in1):

14 3
1 1 3 3 3 1 1 1 3 3 2 2 3 3
1 3 2

Sample Output (see horrorplaylist.out1):

5
```

In the sample input above, there are 14 videos in the playlist and 3 different types of videos. The second line contains the types of videos in the playlist. So the first two videos in the playlist are of type 1, the next three videos are of type 3 and so on. The third line indicates the order in which Olivia watches these videos. So she watches all videos of type 1 first, then type 3 and finally type 2. It will take two clicks to watch all videos of type 1, 2 clicks to watch all videos of type 3, and only 1 click to watch all videos of type 2. So the total number of clicks is 5.

For full credit, your program should work for $k \leq 10^6$ and $n \leq 10^7$. Please name your program `HorrorPlaylist.java`.

Hints for Points: The instructor has compassionately left two hints with his teaching assistant Maclean Frazier. The hints provide brief descriptions of algorithms to solve the above problem. The first hint contains descriptions of a slower solution, while the second hint contains descriptions of a faster solution. You may email Maclean at *frazierms@appstate.edu* to obtain these hints. However, two points will be taken off for every hint. Moreover, you cannot obtain the second hint without obtaining the first hint, meaning it will cost you a total of 4 points to get the second hint. So understand the following when asking for hints.

(a) You email Maclean (not me) to obtain these hints.

(b) 2 points are taken off for each hint.

(c) You cannot get the second hint without spending two points on the first hint.

If you have implemented your program correctly, it may still be possible to get a decent score on this problem with the hints. Or you may try to solve this without the hints. The choice is yours.

2. **Candy Collector (10 points):** Your professor's many nephews and nieces are going from door to door collecting their treats dressed as witches, warlocks and wizards. Being well versed in everything about candies, they have meticulously created a list of all the different types of candies being produced in the entire world! As they go from door to door collecting their candies, they receive many different types of candies. They may get multiple candies of the same type. They wonder if they got at least one candy of every type.

The first line of input contains two numbers k and n, where k is the number of types of candy, and n is the total number of candies collected. The next n lines contain a single integer that indicates a type of candy collected. As output please print a message whether they were able to collect all k types of candy or not. If they were not able to collect all k types of candies, please indicate how many types of candies they are missing. Two sample inputs and outputs are shown below, and also provided in files `candycollector.in1` (and its corresponding output `candycollector.out1`) and `candycollector.in2` (and its output `canducollector.out2`). Please format your output exactly as seen in these files. We will be using the `diff` utility to compare the output of your program with the expected outputs.

Sample Input (see candycollector.in1):

```
5 10
1
2
1
1
1
3
4
3
2
5
```

Sample Output (see candycollector.out1):

```
Yay! all 5 type(s) of candies collected.
```

Sample Input (see candycollector.in2):

```
5 10
1
2
1
1
1
3
4
3
2
4
```

Sample Output (see candycollector.out2):

```
Missing 1 type(s) of candy.
```

Please name your program `CandyCollector.java`. For full credit, your program should work for $k \leq 10^6$ and $n \leq 10^7$. Unfortunately, no hints are available for this problem.

Submission: Please submit all the files as a single zip archive `mid-term.zip` through ASULearn. The zip archive should only contain the individual files of the assignment, and these files should not be inside folders. Moreover, please do not include other extraneous files. Only include all files that belong to your solution.

Input/ Output Instructions: For all programs, until and otherwise stated, we will be taking the input from standard input (`System.in`) and will be sending the output to standard output (`System.out`). Since we will be using an automatic grading system, please make sure that your output format exactly matches the description above. So make sure that there are no extraneous output in terms of spaces, newlines and other characters.

Notes on Coding: Please do not include user-defined packages in your code. Your code should run in the Unix/Linux machine using the commands `javac` and `java`.

Please note that you are **not allowed to use Java Collections** which contain pre-defined libraries for many of the data structures that we learn in class. The purpose of these assignments is to build these data structures from first principles. Programs that includes these objects will receive 0 points.