CS3460: Data Structures Fall 2021
Final Exam
Due Date: Friday, December 03; 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

Good Luck!

1. **Magic Squares (15 points):** Following the success of the magic cube, Mr. Rubik invented its planar version, called magic squares. This is a sheet composed of 8 equal-sized squares:

1	2	3	4
8	7	6	5

   In this task we consider the version where each square has a different color. Colors are denoted by the first 8 positive integers. A sheet configuration is given by the sequence of colors obtained by reading the colors of the squares starting at the upper left corner and going in clockwise direction. For instance, the configuration shown above is given by the sequence $(1, 2, 3, 4, 5, 6, 7, 8)$. This configuration is the initial configuration.

   Three basic transformations, identified by the letters 'A', 'B' and 'C', can be applied to a sheet:

   - 'A': exchange the top and bottom row,
   - 'B': single right circular shifting of the rectangle,
   - 'C': single clockwise rotation of the middle four squares.

   Below is a demonstration of applying the transformations to the initial squares given above:

A:

8	7	6	5
1	2	3	4

B:

4	1	2	3
5	8	7	6

C:

1	7	2	4
8	6	3	5

All possible configurations are available using the three basic transformations.

Please write a program that computes a minimal sequence of basic transformations that transforms the initial configuration above to a specific target configuration.

The input consists of exactly 8 space-separated integers (a permutation of $\{1 \dots 8\}$) on a single line. This is the target configuration. As output, please print a single integer on the first line that is the length of the shortest transformation sequence. On the second line, please print the lexicographically earliest string of transformation as a string of characters. For example, if two solutions, say "ABC" and "ACB" reach a target configuration, you should print "ABC" as the answer as it is the smallest lexicographic transformation that takes the initial configuration to the target configuration. Sample input and output are shown below, and also contained in files `msquare.in1` and `msquare.out1`.

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

2 6 8 4 5 7 3 1

Sample Output (see msquare.out1):

7
BCABCCB
```

2. **Binomial Coefficients (10 points):** The binomial coefficient $B(n, k)$ arises in the expansion of the polynomial $(a + b)^n$. In fact, it is the coefficient of the $a^k b^{(n-k)}$ term. For example, if we expand $(a + b)^3$, we get $1a^0b^3 + 3a^1b^2 + 3a^2b^1 + 1a^3b^1$. The coefficient of the a^1b^2 term is 3, and therefore $B(3, 1) = 3$. It also represents the number of ways of choosing k items among n total items, and is usually expressed using the formula $B(n, k) = \dfrac{n!}{k!(n-k)!}$, where $n! = 1 \times 2 \times 3 \dots \times n$.

   Using the above formula however, is not the most accurate or efficient way to compute $N(n, k)$ due to multiplying and dividing large numbers. We can compute $B(n, k)$ using only additions with the following recursive definition.

   (a) $B(n, 0) = B(n, n) = 1$.

   (b) $B(n, k) = B(n - 1, k - 1) + B(n - 1, k)$.

   In this problem, we would like to compute $B(n, k)$ given n and k as input. Your program must have a **recursive function** that computes $B(n, k)$. Since the output can be very large, we will use Java's implementation of `BigInteger` to store arbitrarily large integers. Please import `Java.math.BigInteger`. Then we can create a `BigInteger` object using the command

   ```
   BigInteger x = new BigInteger("15");
   ```

   Note that the constructor requires a `String` representation of the number. You can also use the `BigInteger` constants `BigInteger.ZERO` and `BigInteger.ONE` for the numbers 0 and 1 respectively. Please refer to the Java docs `https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html` for more information.

The input to the program are two integers n and k, with $1 \leq n \leq 1000$ and $1 \leq k \leq n$. Please output the value of $B(n, k)$. Two sample input and output are shown below, and are also contained in the respective files.

```
Sample Input (binom.in1):

3 1

Sample Output (binom.out1):

B(3, 1) = 3

Sample Input (binom.in2):
1000 400

Sample Output (binom.out2):

B(1000, 400) = 4965272386254228861150735628896231326213413536598276046629321840126459057320964573821649641365755074171723390420897787519048878570924119105790774124085399482049741297783904373939542516768005246806534782666623643526192441809311540207011119823280007769803059555256495013699432020799967895391 50
```

You can check your output using this online calculator `https://www.hackmath.net/en/calculator/n-choose-k`. Please name your program `Binom.java`.

Submission: Please submit all the files as a single zip archive `final.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.