CMPE 224
Fall 2021
Programming Homework 2

This assignment is due by 23:59 on Friday, 12 November 2021.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:

- CMPE224-HW2-OfficeHour1: 1 Nov, 06:00-07:00 PM, Zoom ID: 919 9964 4636
- CMPE224-HW2-OfficeHour2: 5 Nov, 06:00-07:00 PM, Zoom ID: 917 4974 8644

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK(s)

Task 1

Steve Nash is a greatest mathematician and also interested in encryption systems. He is kidnapped by FBI agents to decrypt a mysterious safe. They believe that there could be a great treasure behind so that they have some possible target passwords. To unlock that safe, it is required to reach the target password from the initial combination very quickly. Nash calls for your help to get rid of this pesky situation as soon as possible.

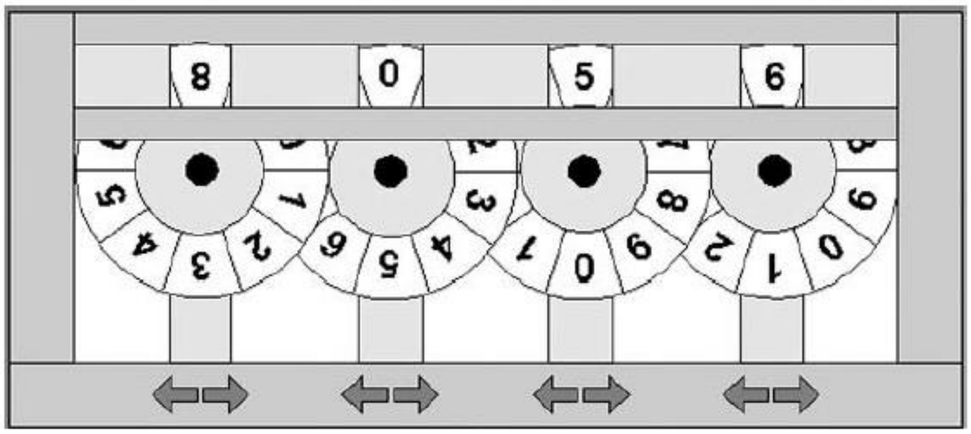

Figure 1. An example configuration for the safe

Figure 1 illustrates an example configuration for the safe. Here are its specifications:

- Digits ranging from 0 to 9 are printed consecutively (clockwise) on the periphery of each wheel.
- The topmost digits of the wheels form a four-digit integer. For example, in Figure 1 the wheels form the integer 8,056.
- Each wheel has two buttons associated with it. Pressing the button marked with a *left arrow* rotates the wheel one digit in the clockwise direction and pressing the one marked with the *right arrow* rotates it by one digit in the opposite direction.

Besides starting with an initial configuration of the wheels, with the topmost digits forming the four-digit integer, you will also be given a set of *n* forbidden password configurations and a target configuration. Your task is to write a program to calculate the minimum number of button presses required to transform the initial configuration to the target configuration without passing through a forbidden one. For this task, you must use your own implementation of undirected graphs by taking inspiration from your textbook. You are not allowed to use any external library or .jar file.

Here is the `sampleinput1.txt` file content:

```
8 0 5 6
6 5 0 8
5
8 0 5 7
8 0 4 7
5 5 0 8
7 5 0 8
6 4 0 8
```

First line of the input file shows the initial configuration of the wheels, specified by four digits. Two consecutive digits are separated by a space. The next line contains the target configuration. The third line contains an integer *n* giving the number of forbidden configurations. Each of the following *n* lines contains a forbidden configuration.

For the test case in the input, you should print a line containing the minimum number of button presses required. If the target configuration is not reachable, you should only print "`-1`". As a hint, you can think this problem as a searching application on undirected graphs.

The output for the `sampleinput1.txt` file is given as follows. Please check your program with this input file as well as the others that you will create. Please note that we may use other input files when grading your assignments.

```
% java Task1 sampleinput1.txt
14
```

Task 2

By filling a rectangle with slashes (/) and backslashes (\), you can generate little mazes. Figure 2 illustrates an example:

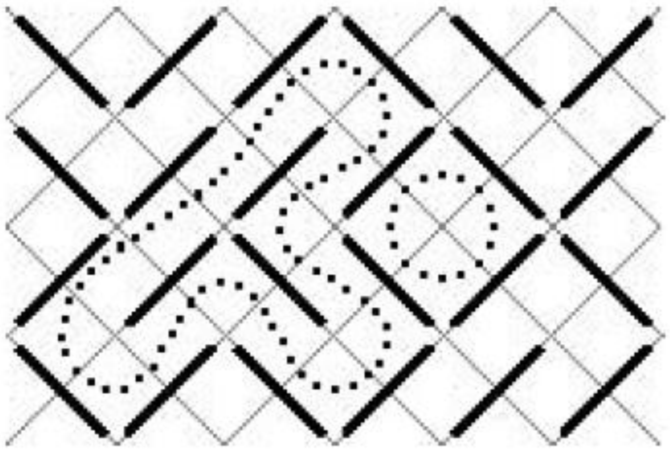

Figure 2. A maze example

As you can see from the Figure 2, the whole maze includes only cyclic paths and paths entering somewhere and leaving somewhere else. You are only interested in *cycles* and there are exactly two of them in that example. Your task is to write a program that counts the cycles and finds the *length* of the longest one. For this task, you must use your own implementation of undirected graphs by taking inspiration from your textbook. You are not allowed to use any external library or .jar file. The length is defined as the number of small squares the cycle consists of (the ones bordered by gray lines in the Figure 2). In Figure 2, the long cycle has length `16` and the short one length `4`.

Here is the `sampleinput2.txt` file content:

```
6 4
\//\\/
\///\/
//\\/\
\/\///
```

First line of the input file contains two integers *a* and *b* ($1 \leq a, b \leq 75$), representing the width and the height of the maze. The next *b* lines describe the maze itself and contain *a* characters each; all of these characters will be either "/" or "\".

For the test case (maze) in the input, you should print a line "`k Cycles; the longest has length l.`", where *k* is the number of cycles in the maze and *l* the length of the longest of the cycles. If the maze is acyclic, you should only print "`There are no cycles.`". As a hint, you can think this problem as a cycle detection application on undirected graphs.

The output for the `sampleinput2.txt` file is given as follows. Please check your program with this input file as well as the others that you will create. Please note that we may use other input files when grading your assignments.

```
% java Task2 sampleinput2.txt
2 Cycles; the longest has length 16.
```

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You can test your Java source files on (if) available Moodle VPL environment to ensure your solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input.
- A **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section).

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%30): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation and Functionality (%40): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%15): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%10): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Friday, November 12th.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-----------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-----------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//--------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//--------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TAs, Bedrettin Çetinkaya, Deniz Merve Gündüz and İbrahim İleri. Thus, you may ask them your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course instructors Tolga Kurtuluş Çapın and Ulaş Güleç for help.