Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2020-2021

Homework #1

Due: 16/10/2020 - 23:55

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!!

1 Introduction

In this assignment, you will validate and complete a Futoshiki puzzle. Futoshiki is a board game which played on fixed size square matrices. Purpose of the game is assigning values to empty cells, values must not be repeated in their row and column while satisfying the unequality constraints. However in this homework, you will be expected to validate a given file is in correct format and find the only possible value of empty cells. More information could be found at futoshiki.org.

2 Program Flow

You will be reading puzzles from provided files. A puzzle consists of two separate files; *board.txt* and *const.txt* whose names will be given by the user when requested by the program. A sample query-response sequence when the program is executed is given below:

```
Enter the board file:  board1.txt
Enter the constraint file:  const1.txt
```

The structure of a board file is as follows:

- A board file begins with an integer n which indicates the size of board.
- After this line, each line of file corresponds to a row of matrix.
- In a row, the integer values from $[1, n]$, separated with a space, represent the value of a corresponding cell of the board.
- The rows can also contain the letter `X` which represents an unknown value in the board.
- There are n such rows in the file, hence, $n + 1$ lines in total.

The structure of a constraint file is as follows:

- A constraint file begins with an integer m which indicates the number of constraints in the puzzle.
- After this line, each line of file is a constraint, hence there are m constraints and $m + 1$ lines in the file.
- Each constraint line is in the form of 4 coordinates, x_1 y_1 x_2 y_2 where the entries are separated with a space. The constraint implies `board[`x_1`][`y_1`]` $>$ `board[`x_2`][`y_2`]`.

The given files may not be in the correct format, i.e., may not have the structure above. You should check the file to satisfy the following properties:

- The size of the board must be equal to n, which is stated in the first line.
- Each value of board, except the unknown values, must be between 1 and n.
- Each empty cell must have only one possible value.
- The number of constraints must be equal to m, which is stated in the first line.
- Values of the coordinates must be between 1 and n.
- The constraints must hold with the values given in the puzzle.

After reading a puzzle from the given files you should check for the properties and report if the puzzle is valid or not. If a file is not valid you should stop execution immediately report the error (see the Sample Runs below) and return.

If puzzle is valid, you should continue and find the values of `X`'s in the matrix. For the puzzles that will be used for testing, there always exists at least one `X` that can only have one possible value. You just need to find this `X` and its corresponding value by checking the row entries, column entries and the corresponding constraints. After finding the value of this `X`, put it on the board and continue with the process to find the values of other `X`s. If you think that all `X`s have at least two possible values at some point (which can be the case) report "`Xs can have more than one value`" and immediately return.

After all unknowns are found, i.e., if the puzzle is solved without a problem, the board must be written to the standard output.

Hint: You can use vector of vectors to store matrices.
Hint: You can use vector of vectors or vector of static arrays to store constraints.
Hint: You can use pairs to store a specific location in matrix.

3 Sample Runs

Below, outputs of the sample puzzles are provided. While you don't have to indicate board name and the reason of invalidity, you must report if puzzle is valid and if it is print the matrix.

```
Enter the board file:  board1.txt
Enter the constraint file:  const1.txt
Files are in correct format
1 2 3 4
2 3 4 1
4 1 2 3
3 4 1 2
```

```
Enter the board file:  board2.txt
Enter the constraint file:  const2.txt
Files are in correct format
3 1 2 4
2 4 1 3
1 3 4 2
4 2 3 1
```

```
Enter the board file:  board3.txt
Enter the constraint file:  const3.txt
Files are in correct format
2 5 4 1 3 6
5 2 3 6 1 4
1 4 2 5 6 3
4 6 5 3 2 1
3 1 6 4 5 2
6 3 1 2 4 5
```

```
Enter the board file:  board4.txt
Enter the constraint file:  const4.txt
Files are in correct format
5 4 2 1 3 6
4 1 3 2 6 5
6 3 5 4 2 1
2 5 4 6 1 3
1 2 6 3 5 4
3 6 1 5 4 2
```

```
Enter the board file:  board5.txt
Enter the constraint file:  const5.txt
Files are in correct format
9 5 1 6 2 8 7 3 4
5 3 7 4 6 2 9 8 1
4 7 3 2 1 5 8 9 6
2 4 8 5 9 7 1 6 3
3 1 2 8 4 9 6 5 7
1 9 5 3 7 6 4 2 8
7 2 6 1 8 3 5 4 9
8 6 9 7 5 4 3 1 2
6 8 4 9 3 1 2 7 5

Enter the board file:  board6.txt
Enter the constraint file:  const6.txt
There is a value repeating in same column

Enter the board file:  board7.txt
Enter the constraint file:  const7.txt
Constraints do not match board

Enter the board file:  board8.txt
Enter the constraint file:  const8.txt
Value 9 is out of matrix size

Enter the board file:  board9.txt
Enter the constraint file:  const9.txt
File contains more lines than size

Enter the board file:  board10.txt
Enter the constraint file:  const10.txt
There are more constraints then previously
stated

Enter the board file:  board11.txt
Enter the constraint file:  const11.txt
Files are in correct format
5 4 2 1 3 6
4 1 3 2 6 5
6 3 5 4 2 1
2 5 4 6 1 3
1 2 6 3 5 4
3 6 1 5 4 2

Enter the board file:  board12.txt
Enter the constraint file:  const12.txt
Files are in correct format
5 4 2 1 3 6
4 1 3 2 6 5
6 3 5 4 2 1
2 5 4 6 1 3
1 2 6 3 5 4
3 6 1 5 4 2

Enter the board file:  board13.txt
Enter the constraint file:  const13.txt
Files are in correct format
1 4 3 2
2 3 4 1
3 2 1 4
4 1 2 3

Enter the board file:  board14.txt
Enter the constraint file:  const14.txt
X's can have more than 1 value
```

4 Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases.**

What and where to submit (PLEASE READ, IMPORTANT): You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the above mentioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course). Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.
Name your cpp file that contains your program as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.cpp

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the folder name must be:

cago_Caglayan_Ozbugsizkodyazaroglu_hw1.cpp

Do not add any other character or phrase to the folder name. Make sure that it contains the last version of your homework program. Compress this folder using WINZIP or WINRAR program. Please use "zip" compression. **"rar" or another compression mechanism is NOT allowed.**. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.
You will receive no credits if your compressed folder does not expand or it does not contain the correct files. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names. **Submit via SUCourse ONLY!** You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

CS204 Team (Fatih Taşyaran, Kamer Kaya)