Sabanci University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Summer 2020-2021

Homework 1 – Minesweeper
Due: 16 July 2021 11.55pm (SHARP)

DISCLAIMER:

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

Only checking the sample run cases might not be sufficient as your solution will be checked against a variety of samples different from the provided samples; however checking these cases is highly encouraged and recommended.

You must NOT collaborate with your friends and discuss your solutions with each other. You have to write down the code on your own. Plagiarism will not be tolerated AND cooperation is not an excuse!

Introduction

The aim of this assignment is to recall CS201 material and practice on matrices (i.e., two dimensional vectors), file streams and string operations. In this assignment, you are going to implement the game Minesweeper according to the rules explained. Players of your game will have 3 opportunities to get help to find where the mines are.

Inputs to Your Program

The first input to your program is the name of a file, in which there will be a matrix, structured as described below. Therefore, your program should first prompt for a filename and read it from the standard input (cin). As mentioned, this file will have the elements of a matrix. If the input file cannot be opened successfully, then your program should ask for another filename, and try opening it repeatedly until the file is successfully opened.

While reading the file, **you must store the matrix in a vector of string or char vectors**. Reading the file more than once would clearly be a bad solution and it will cause a grade deduction.

Other inputs to your program will also be given by the user. All inputs explained below are case sensitive. There are 3 options for users to select while playing the game:

- Open a cell.
  - When the user enters coordinates in the following format, your program should perform the open cell action.
    - *-o xcoord ycoord*
- Mark as a bomb or remove the mark of a bomb.
  - If the user enters the coordinates in the following format, then your program should mark the cell with the uppercase letter 'B', if the cell was not already marked. In the case that the cell is already marked, then this action should remove that mark.
    - *-b xcoord ycoord*
- *Get help*
  - The following input format indicates that the user wants to use one of his/her 3 chances to get help to find where one of the remaining bombs is.
    - *-h*

Input File

File given will start with a line indicating the number of rows and columns. These two pieces of information may be separated by space(s), tab(s), or a mixture of both. For the matrix file, these two numbers exactly tell how many rows and columns the matrix has. You can trust that the content of the file will obey this rule at all times.

After the initial line, in the matrix file, there will be as many more lines as dictated by the row count, and each of these lines will have as many elements as dictated by the column count. Thus, each line represents a row of the matrix, and the elements on the lines make up the columns. Here, the elements on a line may be separated by space(s), tab(s), or a mixture of both. You may assume that the matrix file will consist only of characters and spaces/tabs/newlines except row and column counts given in the first line.

The file given will only contain the dash ('-') character and the lowercase letter 'x' except the first line. Here, "x"s indicate that there is a bomb in the coordinate they stand on and "-"s are the safe zones.

You may assume that no format issue will be present for the given file. Hence, you do not need to check the content or format of the file before parsing. Coordinates of the first cell (top left corner) in the matrix is (0,0) and the user will give coordinates to open and mark the cells accordingly.

For any further details, you can see the sample runs and sample files.

Program Flow

Your program should start by asking the filename for the field matrix. While doing so, your program should make sure that the files with the given names actually exist. If the input file cannot be opened successfully, then your program should ask for another filename, and try opening it repeatedly until the file is successfully opened. In each try, the error message **"Problem occurred while reading the file!!!"** should be printed.

Then, your program should create and print out a board matrix that hides where the bombs are and of course this board matrix has to be in the same size with the field matrix given in the file. Your program should use the dot ('.') character to show the closed cells. Your program should use setw(4) while printing out the board matrix.

After that, your program will wait for the player inputs. The board matrix should be updated and printed each time a change is made by the user. The game will continue until a bomb is opened or there are no more closed cells left (that is, all the cells in the board game are either opened, shown as 'x', or tagged as 'B'). The number of bombs included in the field matrix are random, and as explained later in the document, information shown in the board matrix includes the number of neighbouring bombs as the cells are opened.

The characters that can occur on the board that is shown to the user are given in the table below.

Character	**DESCRIPTION**
.	A closed cell which may or may not be a bomb
Num(0,1..)	Number of bombs neighbour to a cell (up to at most 8)
B	A cell that has been tagged as bomb by the user
x	Bombs shown to the user when help chance is used

Rules of the Minesweeper Game

In this game, there will be a matrix that contains an unknown number of bombs in it. Initially, all the cells shown to the user will be closed (indicated with the dot character ('.')). As explained before, players will have 3 options; to open a cell, to mark/unmark a cell and to get help.

- When player selects a cell and opens it;
    - If the cell contains a bomb, then the game will end with a message saying **"You opened a mine! Game over:("** will be printed.
    - If the cell is a neighbour to at least one bomb, then only the cell itself will open and show the count of bombs it has as neighbours by checking its 4 diagonals, up, down, right and left cells. Please note that a cell can have 8 neighbours at most, and a cell can contain at most 1 bomb.
    - If the cell has no bombs in its neighbours, then the cell itself and *only* its neighbours will open (at most 9 cells, including the cell itself), and show how many bombs they have around if there are any. The cell selected by the user should be replaced with 0 in the output, since it does not include any neighboring cells with a bomb. On the other hand, the neighbouring cells should be replaced with the count of their neighboring cells having bombs if there are any; otherwise, they should also be replaced by 0.
        - Please note that this particular rule is simplified as compared to the original game. In the original game, not only the neighbours but also neighbours' neighbours, and so, are opened and they each show how many bombs are there around themselves. But in our game, only the neighbours will be checked, opened and at the end show how many bombs they have in their own neighbours. Please note that, if the neighbours of the opened cell's neighbours do not include any bombs, then your program should replace the respective cells with 0 as well.
    - If the cell selected was already opened and tagged as bomb, then your program should give an error message saying **"It seems like this cell is a bomb."** and continue.
    - If the cell selected was already opened but not tagged as a bomb, then your program should give an error message saying **"Already opened."**.
    - If the cell selected was outside of the given matrix, then your program should give an error message saying "**Please enter a valid coordinate! ".**

- When the player chooses to mark a cell as a bomb;
    - If the cell is closed, then it will be marked with a 'B' character.
    - If the cell is already marked with a 'B' character, then it will change back to a closed cell and the mark will turn back to a dot '.' character.
    - If the cell selected was already opened or shown as a bomb ('x'), then your program should give an error message saying **"Can't mark that cell as a bomb."** and continue.

    - If the cell selected was outside of the given matrix, then your program should give an error message saying "**Please enter a valid coordinate! ".**

- When the player chooses to get help;
    - The user will lose their chance to learn where a bomb is, if
        - All the cells are closed, or
        - None of the open cells has numbers higher than 0, or
        - There are cells with numbers higher than 0, but their neighbours are opened or marked as bombs.

      In such cases, an error message saying **"I can't help you."** will be printed.
    - If none of the conditions listed above are satisfied, i.e. when the user has the chance to get the help, your program should find the bomb that is
        - Closest to the upper left corner of the board matrix, and
        - Having an opened neighboring cell with a bomb count value, and
        - Which has not been flagged as a bomb.

      In other words, on the board matrix, if there are closed cells ('.') next to the cells with numbers on them, your program should open a cell with a bomb as a hint. Your program should start its search to decide on the hint from the top left corner, moving in the right direction until the columns are over, and keep searching from the row below.
    - Opening bombs with the help option will change the cell's character to 'x'.
    - The user will have only 3 help chances; and if they use all their chances, the player will not get help anymore. In that case, your program should give the error message **"Your help chances are over. :("**.

- If the user enters an invalid input (any input other than the ones described above), then your program should give the error message **"Invalid choice!!"**.

The purpose of the game is to open all the cells and detect (mark) bombs' places without opening them. When all the cells are opened and all the bombs (but only the bombs) are tagged as 'B' or 'x', the game will end and the player will win and message **"Congrats! You won!"** will be displayed. If any of the safe cells are tagged as bomb 'B' at the end the player will lose and message **"You put bombs in the wrong places!"** will be displayed. If any of the bombs is opened while playing, then the player will lose and message "You opened a mine! Game over:(" will be displayed.

Hint: You may want to create a third matrix that has the solution; meaning that cells representing the number of bomb neighbours and of course the bombs' places. This solution matrix can be helpful while updating the board matrix shown to the user while opening the cells.

Sample Runs

Below, we provide some sample runs for the program that you will develop. The *italic* and **bold** phrases are the standard input (cin) taken from the user (i.e., like ***this***). You have to display the required information in the same order and with the same words/spaces as here; in other words, there must be an exact match!

Please make sure that **setw(4)** has been used to print the board matrix in every step, otherwise you might encounter a problem with the GradeChecker.

We will be automatically grading your THE submission using GradeChecker, so it is very important to satisfy the exact same output given in the sample runs. You can utilize GradeChecker (https://learnt.sabanciuniv.edu/GradeChecker/) to check whether your code is working in the expected way. To be able to use GradeChecker, you should upload all of your files used in the homework **without zipping them**. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

Sample Run 1

Enter the input file name: ***file.txt***
Problem occurred while reading the file!!!
Enter the input file name: ***field.txt***
Welcome to the Minesweeper Game!
You may choose a cell to open (-o), get help (-h) or mark/unmark bomb (-b)!!

```
   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .
```

Please enter your choice: ***-o 0 4***
You opened a mine! Game over:(

Sample Run 2

```
Enter the input file name: field.txt
Welcome to the Minesweeper Game!
You may choose a cell to open (-o), get help (-h) or mark/unmark bomb (-b)!!
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .


Please enter your choice: -o 1 1
 0 0 0 . . . . . .
 0 0 0 . . . . . .
 0 0 0 . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .


Please enter your choice: -o 0 3
 0 0 0 1 . . . . .
 0 0 0 . . . . . .
 0 0 0 . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .
 . . . . . . . . .


Please enter your choice: -o 1 3
 0 0 0 1 . . . . .
 0 0 0 1 . . . . .
 0 0 0 . . . . . .
```

```
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
```

Please enter your choice: ***-o 1 4***

```
  0  0  0  1  . . . . .
  0  0  0  1  1  . . . .
  0  0  0  . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
```

Please enter your choice: ***-b 0 4***

```
  0  0  0  1  B  . . . .
  0  0  0  1  1  . . . .
  0  0  0  . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
```

Please enter your choice: ***-o 0 4***
It seems like this cell is a bomb.
Please enter your choice: ***-b 0 4***

```
  0  0  0  1  . . . . .
  0  0  0  1  1  . . . .
  0  0  0  . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
  . . . . . . . . .
```

```
Please enter your choice: -o 0 4
You opened a mine! Game over:(
```

Sample Run 3

```
Enter the input file name: field_2.txt
Welcome to the Minesweeper Game!
You may choose a cell to open (-o), get help (-h) or mark/unmark bomb (-b)!!
  .  .  .  .  .
  .  .  .  .  .
  .  .  .  .  .
  .  .  .  .  .


Please enter your choice: -o 3 0
  .  .  .  .  .
  .  .  .  .  .
  1  3  .  .  .
  0  1  .  .  .


Please enter your choice: -h
  .  .  .  .  .
  x  .  .  .  .
  1  3  .  .  .
  0  1  .  .  .


Please enter your choice: -o 1 1
  .  .  .  .  .
  x  3  .  .  .
  1  3  .  .  .
  0  1  .  .  .


Please enter your choice: -h
  .  .  .  .  .
  x  3  x  .  .
  1  3  .  .  .
  0  1  .  .  .
```

Please enter your choice: ***-b 2 2***

```
  .  .  .  .  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  .  .  .
```

Please enter your choice: ***-o 3 3***

```
  .  .  .  .  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  .  2  .
```

Please enter your choice: ***-o 3 2***

```
  .  .  .  .  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  1  2  .
```

Please enter your choice: ***-o 0 0***

```
  1  .  .  .  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  1  2  .
```

Please enter your choice: ***-o 0 1***

```
  1  2  .  .  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  1  2  .
```

Please enter your choice: ***-o 0 2***

```
  1  2  1  .  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  1  2  .
```

```
Please enter your choice: -b 3 4
  1  2  1  .  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  1  2  B
```

```
Please enter your choice: -h
I can't help you.
Please enter your choice: -h
Your help chances are over. :(
Please enter your choice: -o 0 3
  1  2  1  2  .
  x  3  x  .  .
  1  3  B  .  .
  0  1  1  2  B
```

```
Please enter your choice: -o 1 3
  1  2  1  2  .
  x  3  x  3  .
  1  3  B  .  .
  0  1  1  2  B
```

```
Please enter your choice: -o 2 3
  1  2  1  2  .
  x  3  x  3  .
  1  3  B  3  .
  0  1  1  2  B
```

```
Please enter your choice: -b 0 4
  1  2  1  2  B
  x  3  x  3  .
  1  3  B  3  .
  0  1  1  2  B
```

```
Please enter your choice: -o 1 4
  1  2  1  2  B
  x  3  x  3  1
  1  3  B  3  .
  0  1  1  2  B
```

```
Please enter your choice: -o 2 4
  1  2  1  2  B
  x  3  x  3  1
  1  3  B  3  1
  0  1  1  2  B
```

```
Congrats! You won!
```

Sample Run 4

```
Enter the input file name: field_3.txt
Welcome to the Minesweeper Game!
You may choose a cell to open (-o), get help (-h) or mark/unmark bomb (-b)!!
    .    .    .
    .    .    .
    .    .    .


Please enter your choice: -o 2 0
    .    .    .
    0    1    .
    0    0    .


Please enter your choice: -b 1 0
Can't mark that cell as a bomb.
Please enter your choice: -b 0 0
```

```
     B      .      .
     0      1      .
     0      0      .


Please enter your choice: -o 3 4
Please enter a valid coordinate!
Please enter your choice: -b 0 2
     B      .      B
     0      1      .
     0      0      .


Please enter your choice: -o 0 1
     B      1      B
     0      1      .
     0      0      .


Please enter your choice: -o 2 2
     B      1      B
     0      1      1
     0      0      0


You put bombs in the wrong places! Game over:(
```

Some Important Rules

Although some of the information is given below, please also read THE submission and grading policies from the lecture notes of the first week. In order to get a full credit, your program must be efficient, modular (with the use of functions), well commented and indented. Besides, you also have to use understandable identifier names. Presence of any redundant computation, bad indentation, meaningless identifiers or missing/irrelevant comments may decrease your grade in case that we detect them.

When we grade your THEs, we pay attention to these issues. Moreover, in order to observe the real performance of your code, we are going to run your programs in Release mode and **we may test your programs with very large test cases**. Hence, take into consideration the efficiency of your algorithms other than correctness.

How to get help?

You may ask your questions to TAs or to the instructor. Information regarding the office hours of the TAs are available at SUCourse+. For the instructor, you may get an appointment via email.

YOU SHOULD USE GRADE CHECKER FOR THIS THE!

You should use Grade Checker (https://learnt.sabanciuniv.edu/GradeChecker/) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

Make sure you upload the .txt files, too.

Grade Checker and the automated grading system use a different compiler than MS Visual Studio does. Hence, you should check the "***Common Errors***" page to see some extra situations to consider while doing your homework. If you do not consider these situations, you may get a lower score (even zero) even if your program works correctly with MS Visual Studio.

Common Errors Page: https://learnt.sabanciuniv.edu/GradeChecker/commonerrors.jsp

Grade Checker can be pretty busy and unresponsive during the last day of the submission. Due to this fact, leaving the THE for the last day generally is not a good idea. You may wait for hours to test your THE or make an untested submission, sorrily..

Grade Checker and Sample Runs together give a good estimate of how correct your implementation is, however we may test your programs with different test cases and **your final grade may conflict with what you have seen on Grade Checker.** We will also **manually** check your code (comments, indentations and so on), hence do not object to your grade based on the Grade Checker results; but rather, consider every detail on this documentation. **So please make sure that you have read this documentation carefully and covered all possible cases, even some other cases you may not have seen on Grade Checker or Sample Runs**. The cases that you *do not need* to consider are also given throughout this documentation.

Submit via SUCourse+ ONLY! **Grade Checker is not considered as a submission**. Paper, e-mail or any other methods are not acceptable, either.

The internal clock of SUCourse+ might be a couple of minutes skewed, so make sure you do not leave the submission to the last minute. In the case of failing to submit your homework on time:

"No successful submission on SUCourse+ on time = A grade of 0 directly."

What and where to submit (PLEASE READ, IMPORTANT)

You should test your program using Grade Checker. We will use the same UNIX based C++ compiler that Grade Checker uses for grading your THE.

It'd be a good idea to write your name and lastname in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). If your full name is "Duygu Karaoğlan Altop", and if you want to write it as comment; then you must type it as follows:

// Duygu Karaoglan Altop

Submission guidelines are below. Since the grading process will be automatic, you are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be *zero*. The lack of even one space character in the output will result in your grade being zero, so please test your programs yourself and with the Grade Checker tool explained above.

- Name your cpp file that contains your program as follows:

  "SUCourseUserName_the1.cpp"

  Your SUCourse user name is actually your SUNet username which 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 SU e-mail address is **atam@sabanciuniv.edu**, then the file name must be: **"atam_the1.cpp"**

- Please make sure that this file is the latest version of your THE program.
- You should upload all the .txt files to SUCourse+ as well.
- Do not zip any of the documents but upload them as separate files only.
- Submit your work **through SUCourse+ only**! You can use the Grade Checker only to see if your program can produce the correct outputs both in the correct order and in the correct format. It will not be considered as the official submission. You must submit your work to SUCourse+.

You may visit the office hours if you have any questions regarding submissions.

Plagiarism

Plagiarism will be checked by automated tools and we are very capable of detecting such cases. Be careful with that...

Exchange of abstract ideas are totally okay but once you start sharing the code with each other, it is very probable to get caught by plagiarism. So, do NOT send any part of your code to your friends by any means or you might be charged as well, although you have done your THE by yourself. THEs are to be done personally and you have to submit your own work. **Cooperation will NOT be counted as an excuse.**

In case of plagiarism, the rules on the Syllabus apply.

Good Luck!
Elif Pınar Ön