Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Spring 2025

Homework1 – A Game with Matching Shapes

Due: February 19, 2025, 21:00 (**not** midnight)

No Late Submissions

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases (with the exceptions of assumptions given in the homework text); you are expected to take action accordingly!

You can NOT collaborate with your friends and discuss solutions. You must write down the code on your own. Plagiarism will not be tolerated!

Introduction

In your first homework of this semester, you are going to recall and practice your programming skills in implementing a single player game using C++. In the game, you will use a two-dimensional (2D) matrix that must be implemented as a `vector` of `vector` of `char` data type. The matrix will contain characters that represent different shapes of elements. The initial content of the matrix will be read from a text file. After that the game starts. At each round, the player swaps two adjacent elements of the matrix (this also requires some input as will be explained later) and after that if there exists a group of consecutive same-shape elements of three or more, they are removed from the matrix. Then, the remaining elements are dropped down according to gravity. This group removal and gravity application processes automatically continue until no more groups of minimum three same shape elements exist. The player may continue with more swaps

until it wants to stop the game. There are also some inputs checks to be done during matrix reading and game playing, which are explained in the rest of the document.

Inputs

This homework requires both file and keyboard input.

Input from file

Mainly the initial content of the game board will be read from a file. A valid text file contains a 2D matrix of characters structured as rows and columns. Each character must be one of the following uppercase letters: ‘O’, ‘S’, or ‘X’.

There are some input checks that you should perform for the file content.

- Your program should check for consistency in the number of characters in each row. In other words, all rows must be of the same length.
- Your program should check that the file only contains three uppercase letters, as mentioned above. Any other letter/symbol (even blank characters or blank lines) in a file will make that file invalid.

You should make these checks while reading the file. In case of an error, you have to display an error message and exit the program. Please check out sample runs for the content of the error message.

The file name is a keyboard input; the details of reading the name of the file is explained in the "Program Flow" section below.

Input from keyboard

After successfully reading the matrix content from a valid input file and storing it in a **`vector`** of **`vector`** of **`char`** structure, your program will read inputs for game playing. At each round, two integers for the cell position, and a character for the direction (‘r’ for right, ‘l’ for left, ‘u’ for up, and ‘d’ for down) will be read. The cell position will be read in the form of a *`row column`*, where *`row`* corresponds to the row (horizontal) index of a cell, and *`column`* corresponds to the column (vertical) index of a cell. The top-left cell of the matrix has the indices of (0,0), i.e., 0th *`row`* and 0th *`column`*.

You may assume that the user always enters integers for the indices. However, you should check whether the indices are nonnegative. If a negative index is entered, you should give an error message. In case of a wrong direction letter is entered, it is also an error yielding an error message. Please check out the sample runs for this error message content. After displaying such an error message, the program should read new set of inputs to continue to game.

Entering `0 0 q` indicates the end of the game. That means, when this particular triplet is entered, the game finishes.

Program Flow

Your program will start by asking the user to enter a file name for the input text file. Then, your program will open the file and check if it is opened successfully or not (you may use `fail()` member function of input file stream for this). If, for some reason, the program could not open the file, it should keep reading another file name until it is opened successfully. When the program opens the file and reads its content into the matrix, firstly it should check for the validity of the matrix by checking the rules mentioned in the "Input from file" section above. If the matrix is not valid, then the program should display a generic error message and terminate as shown in some sample runs. Otherwise, the program continues as explained below.

After a valid matrix is read, your program should print the matrix on the screen and then proceed with game playing. In each round of the game, the player will enter row and column indices of a cell, and a character indicating the direction of intended swap, as explained in the "Input from keyboard" section above.

After that, your program will try to swap two adjacent cells on the matrix: the cell of which the indices are entered, and its neighbor at the entered direction. A swap is considered to be a successful (valid) one if it results in a horizontal or vertical match of three or more cells of the same shape. If the swap is not valid (more on this later), the matrix will not be changed. If, on the other hand, the swap is a valid one, then there will be a sequence of actions that your program needs to take. Firstly, the matrix should be displayed after the swap is done, before clearing the matches. Then, the matches will be cleared by replacing the matched cells with dashes '-'. If there are both horizontal and vertical matches, you have to apply first the horizontal ones before the verticals. Here, please remark that after applying a horizontal clearing, a previously existed vertical match may disappear due to removal of a particular cell (you may check sample run 7 for this case). Once all the matches are cleared, you will display the current content of the matrix. Then, the matrix will be subjected to a gravity effect that makes the cells on upper rows fall down, if there are empty cells below them in the same column. You have to display the matrix after applying gravity as well. Of course, after applying the gravity, since the matrix will change, some new matches might be formed without any player intervention. Your program should also clear them and then apply gravity continuously (using the clearing and gravity rules explained above) until no matches are created.

So far we have explained valid player input yielding match(es). However, the player input might not yield a match or there could be some content related problems with the input. These cases are explained below. All of these issues should generate different error messages (please check out sample runs) and in case of having multiple error cases, you should display the upper one and not check the rest.

1) Indices out of boundary: if the entered row and column indices are out of matrix boundary, then you have to display an error message. Of course, you should not try to make a swap.
2) Neighbor is out of boundary: if the cell is on the matrix, but its neighbor on the direction entered is not, then this is also an error. An appropriate error message must be displayed. Of course, you should not try to make a swap in such a case as well.

3) Empty cell(s): If one or both of the cells to be swapped is/are empty, then you cannot swap them. You should also display an error message in such a case.
4) Swap not yielding a match: After a swap, if there becomes no matching of at least three consecutive same shape cells (horizontal or vertical), then this swap is considered as failure and needs to be undone. Moreover, you should also display an error message indicating this.

So far, one round of the game has been explained. The game continues as many rounds as the player wants. The player indicates finishing the game by entering `0 0 q` as the keyboard input for a round. When this is entered, a good bye message must be displayed and the program should terminate.

Last, but not the least, there is an important assumption about the initial content of the matrix that is read from the input file. We assume there will be no existing matches in the matrix present in any of the input files in this homework. That means, you do not need to check matches before getting the first round inputs from the player.

Some Important Programming Rules

Please do not use any non-ASCII characters (Turkish or other) in your code (not even as comments). And also do not use non-ASCII characters in your file and folder names. We really mean it; otherwise, you may encounter some errors.

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 homework, 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**. Of course, your program should work in *Debug* mode as well.

You are **not** allowed to use codes found somewhere online. These restrictions include code repositories, sites like geeksforgeeks, codes generated by GenAI tools, etc. Moreover, you are not allowed to use any statement, command, concept, topic that has not been covered in CS201 and CS204 (until now). Trying to find help online would generally cause such a problem. Thus, you always have to find help within the course material.

You cannot use `break` and `continue` statements. Global variables cannot be used either.

You are allowed to use sample codes shared with the class by the instructor and TAs. However, you cannot start with an existing .cpp or .h file directly and update it; you have to start with an empty file. Only the necessary parts of the shared code files can be used and these parts must be clearly marked in your homework by putting comments like the following. Even if you take a

piece of code and update it slightly, you have to put a similar marking (by adding "`and updated`" to the comments below.

```
/* Begin: code taken from lab1.cpp */

…

/* End: code taken from lab1.cpp */
```

Since CodeRunner configuration in this homework does not allow to use extra non-standard C++ library/function/class files, if you want to use such functions/classes covered in the classes (such as strutils), you will need to copy the necessary declarations and implementations to your main program by following the citation rules mentioned above.

Submission Rules and Some Grading Tips (PLEASE READ, IMPORTANT)

It'd be a good idea to write your name and last name in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). For example, if your full name is "Satılmış Özbugsızkodyazaroğlu", then you must type it as follows:

// Satilmis Ozbugsizkodyazaroglu

We use CodeRunner in SUCourse+ for submission. No other way of submission is possible. Since the functionality part of the grading process will be automatic, you have to strictly follow these guidelines; otherwise we cannot grade your homework.

The advantage CodeRunner it is that you will be able to test your code against sample test cases. However, the output should be exact, but the textual differences between the correct output and yours can be highlighted (by pressing "show differences" button) on the submission interface.

You should copy the full content of the main .cpp file and paste it into the specified “Answer” area in the relevant assignment submission page on SUCourse+. Then you can test your code via CodeRunner against the sample runs. Since you will not upload a file, your local cpp file name is not important.

Even any tiny change in the output format will result in your grade being zero (0) for that particular test case, so please test your programs yourself, and against the sample runs that are available at the relevant assignment submission page on SUCourse+ (CodeRunner).

In the CodeRunner, there are some visible and invisible (hidden) test cases. You can test your code via CodeRunner against the sample runs (by pressing the "Precheck" button). You can precheck as much as you can; there is no penalty for multiple prechecks. This Precheck is a bit different than "Check" that you might have used in other courses. After pressing "Precheck" you will be able to see the performance of your code **only** for the visible test cases; you will see a green check mark (for success) or a red cross (for failure) at the beginning of each visible test case. There will **not** be any feedback for the hidden test cases and there will **not** be any

cumulative feedback at the end. Moreover, there will **not** be "show differences" button during Precheck.

You will be able to get feedback about whether or not your code passed all test cases, including the hidden ones, only after you finish your attempt and complete the submission process (while previewing the submission). After the submission, "show differences" button will be there for you to highlight the textual differences between your output and the expected one.

If you see a problem after you finish your attempt and complete the submission process, you will be able to re-attempt and make another submission. There is no penalty for re-attempts. However, you have to finish attempt and complete the submission process every time; **there is no automatic submission on the deadline**.

This process ensures that you will know whether or not your code has successfully passed all the test cases (visible and hidden) before finalizing your ultimate submission. However, we keep our rights to add more test cases in the grading process after the submission. **Thus, please make sure that you have read this documentation carefully and covered/tested all possible cases, even some other cases you may not have seen on CodeRunner or the sample runs.** Due to these reasons, **your final grade may conflict with what you have seen on CodeRunner**. We will also **manually** check your code against some criteria, comments, indentations and so on; hence, please do not object to your grade based on the **CodeRunner** results, but rather, consider every detail on this documentation and in general homework rules.

We will consider your last submission in grading with no exceptions. Thus, please make sure that the last submission is your final solution version. Also, we still do not suggest that you develop your solution on CodeRunner, but rather on your IDE on your computer.

Last, even if you cannot completely finish your homework, you can still submit.

Please see the syllabus for general homework grading issues.

Plagiarism

Plagiarism is 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 share any part of your code to your friends by any means or you might be charged as well, although you have done your homework by yourself.

Homework are to be done personally and you have to submit your own work. **Cooperation will NOT be counted as an excuse.**

Our experience shows that code taken from online sources or generated by AI tools also show resemblance; thus if you try to get such help, you also may be charged by plagiarism with a person that you do not know.

In case of plagiarism, the rules written in the Syllabus apply.

Sample Runs

Sample runs are given below, but these are not comprehensive, therefore you must consider **all possible cases** to get full mark. User inputs are shown in **bold.**

The input files of the visible test cases are also provided in the homework package. In C-Lion, input text files must be placed in the folder that starts with cmake-build-. This is something different than Visual Studio.

We configured CodeRunner to test these sample runs for you (as visible test cases). However, there also are some hidden test cases that would affect your grade. We will not disclose the hidden test cases before the grading has been completed.

We do **not** recommend you to copy and paste the prompts and messages from this document since some hidden control characters and non-standard characters might cause problems in CodeRunner.

Sample Run 1

```
Please enter the file name:
Jan.tx
The file couldn't be opened.
Please enter a valid file name:
Jan.txt
The matrix either has invalid dimensions or contains invalid characters.
Exiting the game. Bye bye.
```

Sample Run 2

```
Please enter the file name:
in2.txt
The matrix either has invalid dimensions or contains invalid characters.
Exiting the game. Bye bye.
```

Sample Run 3

```
Please enter the file name:
in.txt
The file couldn't be opened.
Please enter a valid file name:
in1.tx
The file couldn't be opened.
Please enter a valid file name:
in1.txt
The content of the matrix is:
XOXOXOS
XSXXOOX
SOOXSXS
OOXOSOS

Enter row, col, and direction (r/l/u/d). Type '0 0 q' to exit.
Move:
0 0 r
Invalid move: No match found!
Move:
0 2 l
Invalid move: No match found!
Move:
0 0 l
Move out of bounds!
Move:
0 3 u
Move out of bounds!
Move:
2 0 l
Move out of bounds!
Move:
2 6 r
Move out of bounds!
Move:
3 3 d
Move out of bounds!
Move:
1 5 r
Invalid move: No match found!
Move:
3 6 r
Move out of bounds!
Move:
3 6 f
Invalid input. Try again.
Move:
0 6 r
Move out of bounds!
Move:
3 6 d
```

```
Move out of bounds!
Move:
1 6 r
Move out of bounds!
Move:
6 4 r
Invalid coordinates!
Move:
5 4 l
Invalid coordinates!
Move:
-2 3 g
Invalid input. Try again.
Move:
-2 3 r
Invalid input. Try again.
Move:
-2 -4 u
Invalid input. Try again.
Move:
0 -4 u
Invalid input. Try again.
Move:
0 4 u
Move out of bounds!
Move:
0 4 d
After swap:
XOXOOOS
XSXXXOX
SOOXSXS
OOXOSOS

Move successful. Clearing matches...
After clearing matches:
XOX---S
XS---OX
SOOXSXS
OOXOSOS

After applying gravity:
XO----S
XSX--OX
SOOXSXS
OOXOSOS

Move:
0 0 q
Exiting the game. Bye bye.
```

Sample Run 4

```
Please enter the file name:
Feb
The file couldn't be opened.
Please enter a valid file name:
Feb.text
The file couldn't be opened.
Please enter a valid file name:
Feb.txtt
The file couldn't be opened.
Please enter a valid file name:
Feb.txt
The content of the matrix is:
SXOX
OSXO
XOSX
OXSO
SXOX
OSXO

Enter row, col, and direction (r/l/u/d). Type '0 0 q' to exit.
Move:
5 1 l
Invalid move: No match found!
Move:
6 2 d
Invalid coordinates!
Move:
5 2 d
Move out of bounds!
Move:
1 4 r
Invalid coordinates!
Move:
1 4 l
Invalid coordinates!
Move:
5 5 l
Invalid coordinates!
Move:
5 3 r
Move out of bounds!
Move:
5 4 l
Invalid coordinates!
Move:
1 1 l
Invalid move: No match found!
Move:
1 1 u
Invalid move: No match found!
```

```
Move:
1 1 d
Invalid move: No match found!
Move:
1 1 r
After swap:
SXOX
OXSO
XOSX
OXSO
SXOX
OSXO

Move successful. Clearing matches...
After clearing matches:
SXOX
OX-O
XO-X
OX-O
SXOX
OSXO

After applying gravity:
SX-X
OX-O
XO-X
OXOO
SXOX
OSXO

Move:
2 2 r
Cannot swap with an empty cell!
Move:
2 1 r
Cannot swap with an empty cell!
Move:
2 1 l
After swap:
SX-X
OX-O
OX-X
OXOO
SXOX
OSXO

Move successful. Clearing matches...
After clearing matches:
S--X
---O
---X
--OO
S-OX
```

```
OSXO

After applying gravity:
---X
---O
---X
S-OO
S-OX
OSXO

Move:
5 3 r
Move out of bounds!
Move:
5 2 r
After swap:
---X
---O
---X
S-OO
S-OX
OSOX

Move successful. Clearing matches...
After clearing matches:
---X
---O
---X
S--O
S--X
OS-X

After applying gravity:
---X
---O
---X
S--O
S--X
OS-X

Move:
5 1 r
Cannot swap with an empty cell!
Move:
5 2 r
Cannot swap with an empty cell!
Move:
5 2 u
Cannot swap with an empty cell!
Move:
5 1 d
Move out of bounds!
Move:
```

5 1 u
Cannot swap with an empty cell!
Move:
5 1 l
After swap:
---X
---O
---X
S--O
S--X
SO-X

Move successful. Clearing matches...
After clearing matches:
---X
---O
---X
---O
---X
-O-X

After applying gravity:
---X
---O
---X
---O
---X
-O-X

Move:
3 3 r
Move out of bounds!
Move:
3 3 d
Invalid move: No match found!
Move:
3 3 l
Cannot swap with an empty cell!
Move:
3 3 u
After swap:
---X
---O
---O
---X
---X
-O-X

Move successful. Clearing matches...
After clearing matches:
---X
---O
---O

```
----
----
-O--

After applying gravity:
----
----
----
---X
---O
-O-O

Move:
1 2 u
Cannot swap with an empty cell!
Move:
3 3 h
Invalid input. Try again.
Move:
0 0 s
Invalid input. Try again.
Move:
0 0 q
Exiting the game. Bye bye.
```

Sample Run 5

```
Please enter the file name:
inloong.txt
The file couldn't be opened.
Please enter a valid file name:
in_loong.txt
The file couldn't be opened.
Please enter a valid file name:
in_long.tx
The file couldn't be opened.
Please enter a valid file name:
in_long.txt
The content of the matrix is:
XOXOXOS
XSXXOOX
SOOXSXS
XOXOSOS
XSXXOOX
OXOSOSS

Enter row, col, and direction (r/l/u/d). Type '0 0 q' to exit.
Move:
9 8 r
Invalid coordinates!
Move:
```

5 7 r
Invalid coordinates!
Move:
5 6 r
Move out of bounds!
Move:
5 6 d
Move out of bounds!
Move:
5 6 u
After swap:
XOXOXOS
XSXXOOX
SOOXSXS
XOXOSOS
XSXXOOS
OXOSOSX

Move successful. Clearing matches...
After clearing matches:
XOXOXOS
XSXXOOX
SOOXSX-
XOXOSO-
XSXXOO-
OXOSOSX

After applying gravity:
XOXOXO-
XSXXOO-
SOOXSX-
XOXOSOS
XSXXOOX
OXOSOSX

Move:
5 0 d
Move out of bounds!
Move:
5 0 l
Move out of bounds!
Move:
5 0 u
Invalid move: No match found!
Move:
5 0 r
After swap:
XOXOXO-
XSXXOO-
SOOXSX-
XOXOSOS
XSXXOOX
XOOSOSX

```
Move successful. Clearing matches...
After clearing matches:
XOXOXO-
XSXXOO-
SOOXSX-
-OXOSOS
-SXXOOX
-OOSOSX

After applying gravity:
-OXOXO-
-SXXOO-
-OOXSX-
XOXOSOS
XSXXOOX
SOOSOSX

Move:
2 1 l
Cannot swap with an empty cell!
Move:
2 1 r
Invalid move: No match found!
Move:
2 2 l
Invalid move: No match found!
Move:
2 2 r
After swap:
-OXOXO-
-SXXOO-
-OXOSX-
XOXOSOS
XSXXOOX
SOOSOSX

Move successful. Clearing matches...
After clearing matches:
-O-OXO-
-S-XOO-
-O-OSX-
XO-OSOS
XS-XOOX
SOOSOSX

After applying gravity:
-O-OXO-
-S-XOO-
-O-OSX-
XO-OSOS
XS-XOOX
SOOSOSX
```

Move:
4 2 l
Cannot swap with an empty cell!
Move:
4 2 r
Cannot swap with an empty cell!
Move:
4 1 r
Cannot swap with an empty cell!
Move:
4 1 u
Invalid move: No match found!
Move:
4 1 d
After swap:
-O-OXO-
-S-XOO-
-O-OSX-
XO-OSOS
XO-XOOX
SSOSOSX

Move successful. Clearing matches...
After clearing matches:
-O-OXO-
-S-XOO-
---OSX-
X--OSOS
X--XOOX
SSOSOSX

After applying gravity:
---OXO-
---XOO-
---OSX-
XO-OSOS
XS-XOOX
SSOSOSX

Move:
5 3 r
Invalid move: No match found!
Move:
5 3 l
After swap:
---OXO-
---XOO-
---OSX-
XO-OSOS
XS-XOOX
SSSOOSX

```
Move successful. Clearing matches...
After clearing matches:
---OXO-
---XOO-
---OSX-
XO-OSOS
XS-XOOX
---OOSX

After applying gravity:
---OXO-
---XOO-
---OSX-
---OSOS
XO-XOOX
XS-OOSX

Move:
0 3 l
Cannot swap with an empty cell!
Move:
0 3 r
Invalid move: No match found!
Move:
1 3 r
After swap:
---OXO-
---OXO-
---OSX-
---OSOS
XO-XOOX
XS-OOSX

Move successful. Clearing matches...
After clearing matches:
----XO-
----XO-
----SX-
----SOS
XO-XOOX
XS-OOSX

After applying gravity:
----XO-
----XO-
----SX-
----SOS
XO-XOOX
XS-OOSX

Move:
3 4 r
After swap:
```

```
----XO-
----XO-
----SX-
----OSS
XO-XOOX
XS-OOSX

Move successful. Clearing matches...
After clearing matches:
----XO-
----XO-
----SX-
-----SS
XO-X-OX
XS-O-SX

After applying gravity:
-----O-
-----O-
-----X-
----XSS
XO-XXOX
XS-OSSX

Move:
4 5 r
After swap:
-----O-
-----O-
-----X-
----XSS
XO-XXXO
XS-OSSX

Move successful. Clearing matches...
After clearing matches:
-----O-
-----O-
-----X-
----XSS
XO----O
XS-OSSX

After applying gravity:
-------
-----O-
-----O-
-----XS
XO--XSO
XS-OSSX

Move:
3 6 r
```

```
Move out of bounds!
Move:
3 6 l
After swap:
-------
-----O-
-----O-
-----SX
XO--XSO
XS-OSSX

Move successful. Clearing matches...
After clearing matches:
-------
-----O-
-----O-
------X
XO--X-O
XS-OS-X

After applying gravity:
-------
-------
-------
------X
XO--XOO
XS-OSOX

Move:
0 0 q
Exiting the game. Bye bye.
```

Sample Run 6

```
Please enter the file name:
sample.t
The file couldn't be opened.
Please enter a valid file name:
sample.txt
The content of the matrix is:
XXSO
XOOX
SXSS

Enter row, col, and direction (r/l/u/d). Type '0 0 q' to exit.
Move:
2 0 u
Invalid move: No match found!
Move:
2 0 r
```

```
After swap:
XXSO
XOOX
XSSS

Move successful. Clearing matches...
After clearing matches:
-XSO
-OOX
----

After applying gravity:
----
-XSO
-OOX

Move:
2 3 u
After swap:
----
-XSX
-OOO

Move successful. Clearing matches...
After clearing matches:
----
-XSX
----

After applying gravity:
----
----
-XSX

Move:
0 0 q
Exiting the game. Bye bye.
```

Sample Run 7

```
Please enter the file name:
sample2
The file couldn't be opened.
Please enter a valid file name:
sample2.txt
The content of the matrix is:
SSXO
XSOX
SOSS

Enter row, col, and direction (r/l/u/d). Type '0 0 q' to exit.
```

```
Move:
6 -9 f
Invalid input. Try again.
Move:
6 -9 r
Invalid input. Try again.
Move:
2 0 r
After swap:
SSXO
XSOX
OSSS

Move successful. Clearing matches...
After clearing matches:
SSXO
XSOX
O---

After applying gravity:
S---
XSXO
OSOX

Move:
0 1 r
Cannot swap with an empty cell!
Move:
0 1 u
Move out of bounds!
Move:
0 2 u
Move out of bounds!
Move:
0 2 r
Cannot swap with an empty cell!
Move:
0 0 r
Cannot swap with an empty cell!
Move:
0 0 q
Exiting the game. Bye bye.
```

Sample Run 8

```
Please enter the file name:
in1
The file couldn't be opened.
Please enter a valid file name:
in1.txt
The content of the matrix is:
```

```
XOXOXOS
XSXXOOX
SOOXSXS
OOXOSOS

Enter row, col, and direction (r/l/u/d). Type '0 0 q' to exit.
Move:
3 6 r
Move out of bounds!
Move:
3 3 r
Invalid move: No match found!
Move:
3 3 l
After swap:
XOXOXOS
XSXXOOX
SOOXSXS
OOOXSOS

Move successful. Clearing matches...
After clearing matches:
XOXOXOS
XSX-OOX
SOO-SXS
----SOS

After applying gravity:
----XOS
XOX-OOX
XSX-SXS
SOOOSOS

After clearing matches:
----XOS
XOX-OOX
XSX-SXS
S---SOS

After applying gravity:
----XOS
X---OOX
XOX-SXS
SSX-SOS

Move:
2 5 u
Invalid move: No match found!
Move:
2 5 d
After swap:
----XOS
X---OOX
```

```
XOX-SOS
SSX-SXS

Move successful. Clearing matches...
After clearing matches:
----X-S
X---O-X
XOX-S-S
SSX-SXS

After applying gravity:
----X-S
X---O-X
XOX-S-S
SSX-SXS

Move:
1 6 r
Move out of bounds!
Move:
1 6 u
After swap:
----X-X
X---O-S
XOX-S-S
SSX-SXS

Move successful. Clearing matches...
After clearing matches:
----X-X
X---O--
XOX-S--
SSX-SX-

After applying gravity:
----X--
X---O--
XOX-S--
SSX-SXX

Move:
0 0 q
Exiting the game. Bye bye.
```

Good Luck
Ahmed Salem and Albert Levi