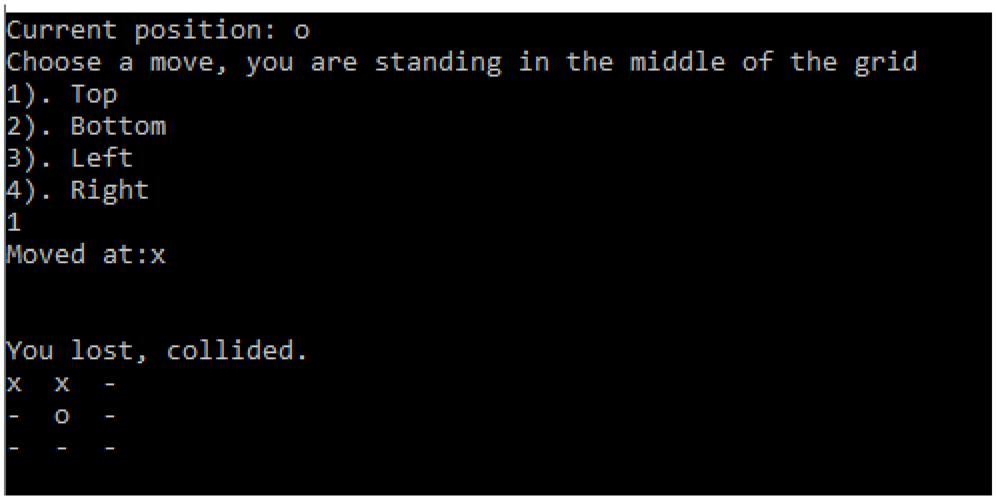
Lab Task 02

Task 1: Wandering in the grid

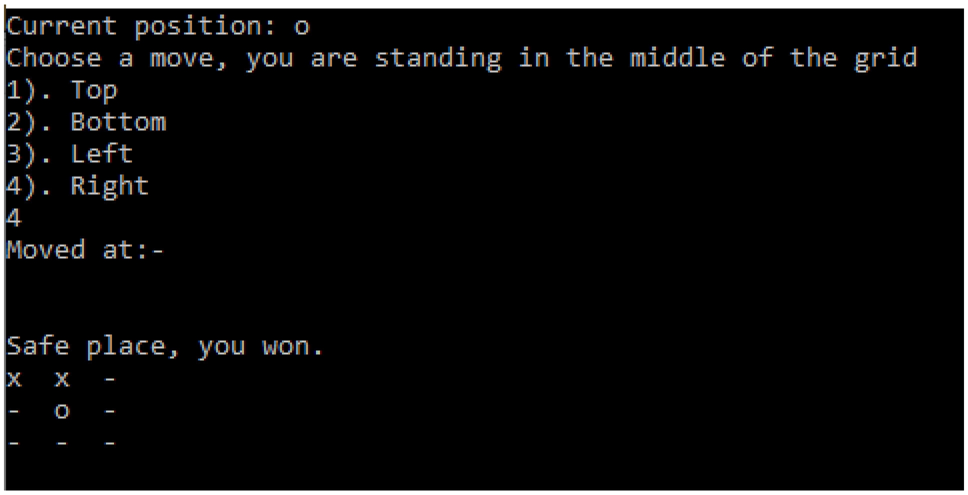
1. Write a program that simulates a simple collision detection game. Create a 2D grid of 3x3 that represents the game world. Each cell can contain either a reward or a collision object. Randomly populate the array with either **–** or **x,** where – represents a safe place, and **x** represents a collision object. However, at the start of the game, the player must be located in the middle of the grid. So, the character at the middle the grid should be **o**. If the player moves to a cell containing a reward, he won, so display the message "Yayy, you won, it was a reward". But if that cell contained a collision object, the player will die, so display the message "You lost, it was a collision object". **Implement movement of a player within the grid using pointers.** The valid moves for a player are either top, bottom, left, and right.

   <u>Sample Output:</u>



```
Current position: o
Choose a move, you are standing in the middle of the grid
1). Top
2). Bottom
3). Left
4). Right
1
Moved at:x


You lost, collided.
x  x  -
-  o  -
-  -  -
```



```
Current position: o
Choose a move, you are standing in the middle of the grid
1). Top
2). Bottom
3). Left
4). Right
4
Moved at:-


Safe place, you won.
x  x  -
-  o  -
-  -  -
```

Task 2: Character Puzzle Quest

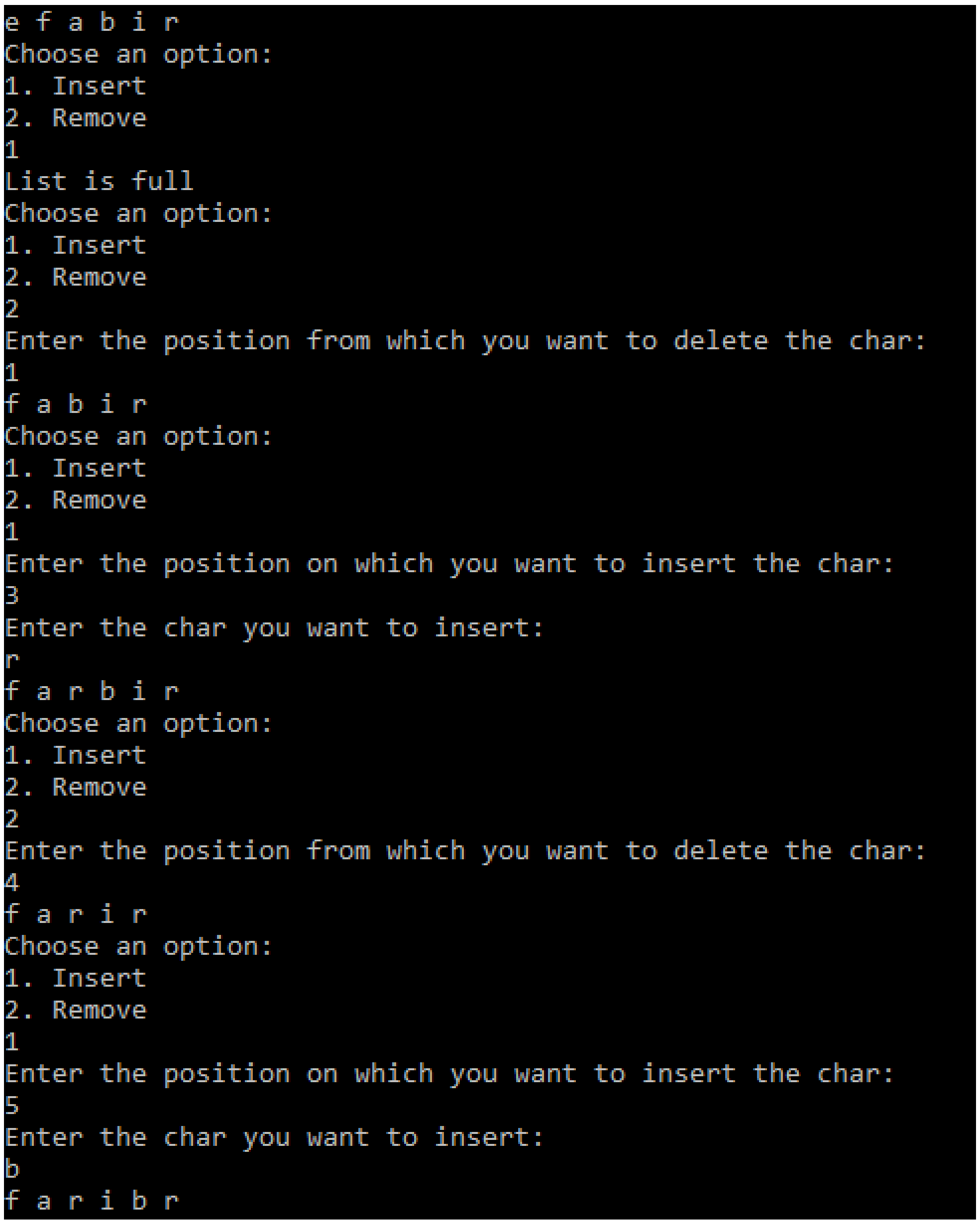
2. You're given an initial sequence of characters in a dynamic array (shuffled string e.g **efabir**). Your goal is to arrange these characters to perfectly match a given target string (**fariba**). You have access to a set of operations—insert and remove—that allow you to manipulate the arrangement of characters within the array. With each operation, you'll transform the array, getting closer to the final arrangement.

   Instructions:

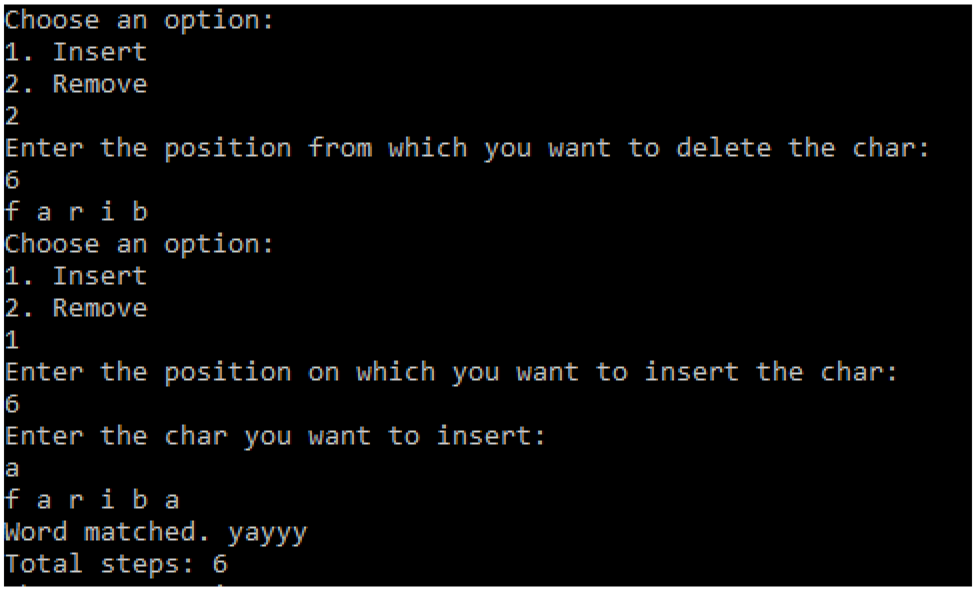
   - You have a sequence of characters already present in the dynamic array (insert the 6 characters hard coded by calling the insert method).
   - Implement the ArrayList class methods, insert, delete, search, printList, get. However, we will use make use of only three methods, rest are for practice only.
   - After each operation, you'll observe the array changing, and you should keep track of the steps taken.
   - Your arrangement should eventually match the provided target string.
   - Let's see who arranges my name with the minimum number of steps. I have done myself using 6 steps 😅

   Note that the task must be performed using dynamic array, and all the operations must be performed using pointers. Plus, you can use any built-in function/ easy method to match the strings.

Sample Output:



```
e f a b i r
Choose an option:
1. Insert
2. Remove
1
List is full
Choose an option:
1. Insert
2. Remove
2
Enter the position from which you want to delete the char:
1
f a b i r
Choose an option:
1. Insert
2. Remove
1
Enter the position on which you want to insert the char:
3
Enter the char you want to insert:
r
f a r b i r
Choose an option:
1. Insert
2. Remove
2
Enter the position from which you want to delete the char:
4
f a r i r
Choose an option:
1. Insert
2. Remove
1
Enter the position on which you want to insert the char:
5
Enter the char you want to insert:
b
f a r i b r
```



```
Choose an option:
1. Insert
2. Remove
2
Enter the position from which you want to delete the char:
6
f a r i b
Choose an option:
1. Insert
2. Remove
1
Enter the position on which you want to insert the char:
6
Enter the char you want to insert:
a
f a r i b a
Word matched. yayyy
Total steps: 6
```