Practice Arena

Practice problems aimed to improve your coding skills.

Trivial Tic-Tac-Toe

LABEXAM-PRAC-02_ENDSEM

Trivial Tic-Tac-Toe [40 marks]

Problem Statement
Mr C was playing tic-tac-toe with his clone. Mr C was playing X and the clone was playing O. However, their game got interrupted since a student asked Mr C to compile a student's program. We will give you the incomplete tic-tac-toe board as input. In your output, you have to generate all possible ways Mr C, i.e. the X player, can win this game if the two of them resume playing.

Recall that the tic-tac-toe board is a 3 x 3 board with 9 cells

```
   |   |
-----------
   |   |
-----------
   |   |
```

We will be playing a slightly simplified version of the tic-tac-toe game in this question for your convenience. In our version, X starts playing in the first round and places an X in some blank cell. Then O places an O in some blank cell. This continues till all the cells are filled in. Given the filled board, if any row is completely filled with X, then X is declared a winner.

Beware that this is not the standard tic-tac-toe game. In the standard game, X wins even if any column is completely filled with X or any diagonal is completely filled with X. A standard game also does not wait for the board to be completely filled in. In a standard game, the moment X is winning, the game ends even if there are blank cells left. However, in this question we are going to use a much simpler version of the game where only row fills allow Mr C i.e. X, to win.

Your output should print every winning board for Mr C on a separate line. A board is printed by first printing the first row then the second row then the third row. Each row is printed from left to right. There are no spaces while printing each board. For example, the board (which is not winning for Mr C)

```
 X | O | X
-----------
 X | X | O
-----------
 O | X | X
```

will be printed as "XOXXXOOXX" (without quotes). Print the winning combinations in lexicographic order i.e. in the order in which these strings would appear in a dictionary (note that O comes before X in the alphabet).

Your input will be given to you as a sequence of 9 characters on the same line (no spaces), indicating the board position at the beginning. An X and O will indicate that those letters have been placed. A B will indicate a blank cell. So
XOBBBBBBB will indicate the starting position.

```
 X | O |
-----------
   |   |
-----------
   |   |
```

Compulsory function usage in your code
This is a question that benefits from use of recursion. In your code, you should write a function in the following format. Be warned that not using such a function to write your code will cause you to lose a small number of manual grading marks.

void genWinX(char *board, int *blanks, int b, int done)

1. board: the partially filled in board (may still have some blanks left in)
2. blanks: an array with the locations of the cells that were originally blank
3. b how many cells are blank
4. done: blank cells have been filled

Function invocation: invoke the function as genWinX(board, blanks, b, 0); from the main function
Base case: the base case happens when we have done = b. In that case we can simply print the board
Recursive case: Think carefully about how you will ensure lexicographic order. Remember, you can either put an O or an X in the next blank position.

Problem-specific Words of Caution:

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. Notice that when the board is full, there will always be 5 X and 4 O on the board. This is because X always starts the game. Also remember that Mr C is playing X.
3. You cannot modify the positions of the board that are already filled in the input. You can only fill in positions that are marked as blank using a B in the input.
4. We assure you that there will be at least one valid winning combination for X given our starting point. However, there may be several. You have to output all of them in lexicographic order.
5. We assure you that the starting point of the board will itself be a valid position of the game. For instance, we will never give you the board as XXXBBBBBBBBB since this would mean that the first 3 rounds were taken by X which is not allowed. X and O alternate in the rounds and X starts off.
6. We will not penalize you for extra newlines at the end of your output. However, do not have stray spaces anywhere in your output.

General Grading Policy

1. **TOTAL MARKS OF THE EXAM** 20 + 40 + 40 + 70 = 170
2. **TOTAL DURATION OF THE EXAM** 3 hours 30 minutes
3. See below for question-specific details of how partial marking would be done by the autograder in this question
4. Your submissions will be inspected by the autograder as well as a human grader
5. Human graders will (among other things) allot marks for the following

    1. Neatly structured code that uses at least one function other than the main function to process the input. The questions will usually suggest how to use functions to process the input. Submissions that ignore these suggestions and use only the main function to solve the entire problem, will lose a small fraction of marks.

    2. Proper and meaningful variable names

3. Nice looking and consistent indentation
4. At least a couple of comments explaining to the human grader what are you doing, especially when the steps are not obvious
5. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.

6. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
7. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
8. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released
9. You are allowed to use the libraries stdio.h, math.h, string.h, stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use any programming tools that we have discussed in lectures/tutorials or in lab questions such as arrays (1D, 2D, 3D, arrays of arrays etc), strings, loops, structures, functions, recursion, pointers, linked lists, stacks, queues, graphs, enumerations, flags, conditionals, global, static and shadowed variables.

EXAMPLE:
INPUT
```
XXXOOBBBB
```

OUTPUT:
```
XXXOOOOXX
XXXOOOXOX
XXXOOOXXO
XXXOOXOOX
XXXOOXOXO
XXXOOXXOO
```

Grading Scheme:
Total marks: **[40 Points]**

There will be partial grading in this question. There are several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. There are 4 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6250)