**Functions**

**Q1-** Write a C++ function that takes a string as a parameter and reverses the order of words. The return type of function is void.

**Input**

The quick brown fox

**Output**

fox brown quick The

**Q2-** Write a C++ function that removes a specific word from a given string. Return the updated string.

**Test Data:**("Exercises Practice Solution", "Solution") -> "Exercises Practice"  
("Exercises Practice Solution", "Practice ") -> "Exercises Solution"  
("Exercises Practice Solution", " Solution") -> " Practice Solution"

**Q3-** Write a C++ program to reverse all words that have odd lengths in a string.

**Test Data:**("Exercises Practice Solution" ) -> "sesicrexE Practice Solution"  
("The quick brown fox jumps over the lazy dog") -> "ehT kciuq nworb xof spmuj over eht lazy dog."

**Q4-** Implement the tic tac toe game, make sure to make use of functions. Feel free to come up with your own logic. **But the main function will only call the playGame(matrix) function**. Here are some sample functions:

* Bool checkColumn(char), checks if any one column has three values of character parameter.
* Bool checkRow(char), checks if any one row has three values of character parameter.
* Similarly, write separate functions to check for rightDiagonal & Left diagonal. I.e checkLeftDiagonal(char) and checkRightDiagonal(char)
* playGame(matrix), to take input from the user alternatively in loop.

**Explanation**

**Tic-tac-toe** is played by two players A and B on a 3 x 3 grid. The rules of Tic-Tac-Toe are:

* Players take turns placing characters into empty squares ' '.
* The first player A always places 'X' characters, while the second player B always places 'O' characters.
* 'X' and 'O' characters are always placed into empty squares, never on filled ones.
* The game ends when there are **three** of the same (non-empty) character filling any row, column, or diagonal.
* The game also ends if all squares are non-empty.
* No more moves can be played if the game is over.

Given a 2D integer array moves where moves[i] = [rowi, coli] indicates that the ith move will be played on grid[rowi][coli]. return *the winner of the game if it exists* (A or B). In case the game ends in a draw return "Draw". If there are still movements to play return "Pending".

Make the check if the move is valid (i.e., it follows the rules of **Tic-Tac-Toe**), the grid is initially empty, and A will play first.

**Q5-** Given an m x n matrix of **distinct** numbers, return *all* ***lucky numbers*** *in the matrix in* ***any*** *order*. A **lucky number** is an element of the matrix such that it is the minimum element in its row and maximum in its column.

Write a C++ function that takes matrix as a parameter and return the array of integers.

**Example 1:**

**Input:** matrix = [[3,7,8],[9,11,13],[15,16,17]]

**Output:** [15]

**Explanation:** 15 is the only lucky number since it is the minimum in its row and the maximum in its column.

**Example 2:**

**Input:** matrix = [[1,10,4,2],[9,3,8,7],[15,16,17,12]]

**Output:** [12]

**Explanation:** 12 is the only lucky number since it is the minimum in its row and the maximum in its column.

**Example 3:**

**Input:** matrix = [[7,8],[1,2]]

**Output:** [7]

**Explanation:** 7 is the only lucky number since it is the minimum in its row and the maximum in its column.