***Exercise 1***: Implementing a Stack using Arrays

1. Define a structure for the stack that includes an array to hold the stack elements and an integer to keep track of the top of the stack.
2. Implement functions for the following operations:
   * Push: to add an element to the top of the stack
   * Pop: to remove and return the element from the top of the stack
   * Peek: to return the element at the top of the stack without removing it
   * isEmpty: to check if the stack is empty
   * isFull: to check if the stack is full
3. Test your stack implementation with various operations.

***Exercise 2***: Implementing a Stack using Linked Lists

1. Define a structure for the stack node that includes a data field to hold the element and a pointer to the next node.
2. Implement functions for the following operations:
   * Push: to add an element to the top of the stack
   * Pop: to remove and return the element from the top of the stack
   * Peek: to return the element at the top of the stack without removing it
   * isEmpty: to check if the stack is empty
3. Test your stack implementation with various operations.

***Exercise 3***: Write a function that takes a word as input and returns the reverse of that word using a stack.

***Exercise 4***: Using a stack, write a function that takes a word as input and checks if the word is a palindrome or not. Recall that a word is a palindrome if it reads the same from left to right and right to left, such as "moon", "dad", "radar", "ici", etc.

***Exercise 5***: Use stacks to test if one word is a suffix of another word. A word is a suffix of another word if the second word ends with the first. For example, the word "mangeable" is a suffix of the word "immangeable".

***Exercise 6***: Write a program that takes an arithmetic expression as input and checks if this expression is valid or not. Here, we only focus on whether the number and order of the following characters are correct: '(', '{', '[', ']', '}', ')'.

***Exercise 7***: Write a program that, based on a stack, evaluates an arithmetic expression taken as input. Expressions consist only of the operations "+, -, \*, /" and we do not consider the priorities of the operations.

***Exercise 8***: Write a program that uses two stacks to sort an input array. Here is an example:



