1. Implement Stack data structure in JavaScript. It should have seven methods (read about stack data structure from Grokking Algorithms).
    - push() - add an element to the stack.
    - pop() - delete an element from the stack.
    - peek() - get the top element of the stack.
    - length() - return the length of the stack.
    - search() - search for the element in the stack
    - isEmpty() - check if the stack is empty.
    - print() - print the elements of the stack.

2. The ABACABA pattern is a recursive fractal pattern that shows up in many places in the real world (such as in geometry, art, music, poetry, number systems, literature and higher dimensions).
   Create a function that takes a number `n` as an argument and returns a `string` that represents the full pattern.

   ```
   abacabaPattern(1) → "A"

   abacabaPattern(2) → "ABA"

   abacabaPattern(3) → "ABACABA"
   ```

3. Write a recursive function that will get all values from the tree.

```
const tree = {
  value: 12,
  next: {
    value: 20,
    next: {
      value: 30,
      next: {
        value: -10,
        next: null
      }
    }
  }
};

const fn = (tree) => {

}
```

```
fn(tree) // [12, 20, 30, -10]
```

4. Implement Insertion sort.

5. Given an unsorted array, find a pair with the given sum in it.
   **Input:**

   nums = [8, 7, 2, 5, 3, 1]
   target = 10

   **Output:**

   Pair found (8, 2)
   or
   Pair found (7, 3)

6*. (Additional task): Chess Knight Problem | Find the shortest path from source to destination

Given a chessboard, find the shortest distance (minimum number of steps) taken by a knight to reach a given destination from a given source.

For example,

**Input:**

N = 8 (8 × 8 board)

Source = (7, 0)

Destination = (0, 7)

**Output:** Minimum number of steps required is 6

The knight's movements are illustrated in the following figure: