Practical-3

Aim : Write a program for finding NGE NEXT GREATER ELEMENT from an array.

Algorithm:-

1. **Initialization**:
    - **Stack**: We create an empty stack to keep track of elements for which we are trying to find the NGE.
    - **Result Array**: We create a result array of the same size as the input array. This array will store the NGE for each element.
2. **Iterate Through the Input Array in Reverse**:
    - We loop through the input array from the last element to the first element. This reverse iteration is crucial because it allows us to find the NGE for each element by looking at the elements that come after it (to the right).
3. **Processing Each Element**:
    - For each element **arr[i]**:
        - **Pop Elements from the Stack**: We pop elements from the stack until either the stack is empty or the top element of the stack is greater than **arr[i]**. This step helps us discard elements that cannot be the NGE for **arr[i]** because they are less than or equal to **arr[i]**.
        - **Check the Stack**:
            - If the stack is empty after popping, it means there is no greater element to the right of **arr[i]**, so we set **result[i]** to -1.
            - If the stack is not empty, the top element of the stack is the NGE for **arr[i]**, so we set **result[i]** to **stack.peek()**.
        - **Push Current Element**: Finally, we push **arr[i]** onto the stack. This step is important because **arr[i]** might be the NGE for the elements to the left of it in the next iterations.
4. **Return the Result Array**:
    - After processing all elements, the result array will contain the NGE for each element in the input array.

Example:-

- **Input Array**: **{9, 3, 8, 1, 2}**
- **Result Array**: Initially, **result = [0, 0, 0, 0, 0]**

Iteration Details

1. **i = 4 (arr[4] = 2)**:
    - Stack is empty, so **result[4] = -1**.
    - Push **2** onto the stack: **stack = [2]**.
2. **i = 3 (arr[3] = 1)**:
    - Stack top is **2** (greater than **1**), so **result[3] = 2**.
    - Push **1** onto the stack: **stack = [2, 1]**.
3. **i = 2 (arr[2] = 8)**:
    - Stack top is **1** (not greater), pop it: **stack = [2]**.
    - Stack top is **2** (not greater), pop it: **stack = []**.
    - Stack is empty, so **result[2] = -1**.
    - Push **8** onto the stack: **stack = [8]**.
4. **i = 1 (arr[1] = 3)**:
    - Stack top is **8** (greater than **3**), so **result[1] = 8**.
    - Push **3** onto the stack: **stack = [8, 3]**.
5. **i = 0 (arr[0] = 9)**:
    - Stack top is **3** (not greater), pop it: **stack = [8]**.
    - Stack top is **8** (not greater), pop it: **stack = []**.
    - Stack is empty, so **result[0] = -1**.
    - Push **9** onto the stack: **stack = [9]**.

Final Result Array

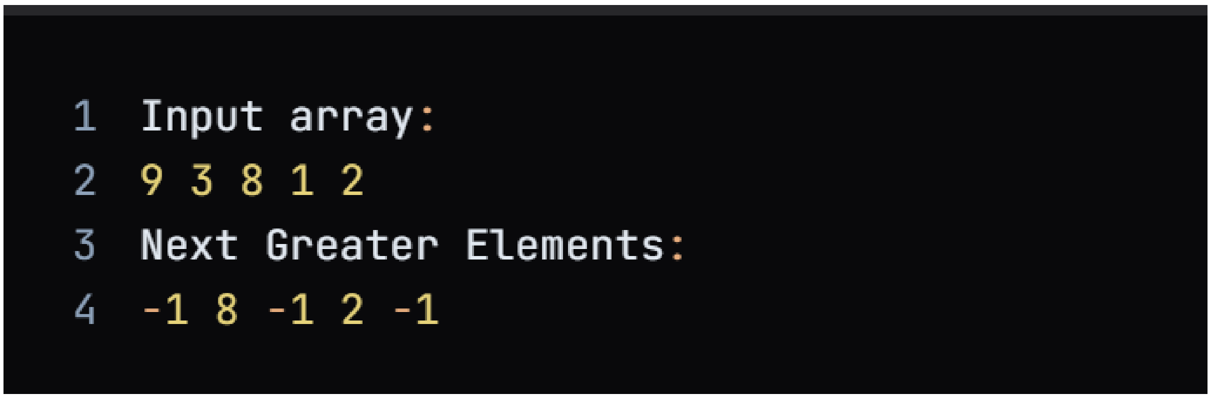
After processing all elements, the **result** array will be:

- **result = [-1, 8, -1, 2, -1]**

Program:-

```
import java.util.Stack;
public class NextGreaterElement {
    public static int[] nextGreaterElement(int[] arr) {
        // Initialize a stack and a result array
        Stack<Integer> stack = new Stack<>();
        int[] result = new int[arr.length];

        // Fill the result array with -1 as default
        for (int i = 0; i < result.length; i++) {
            result[i] = -1;
        }

        // Iterate through the array from right to left
        for (int i = arr.length - 1; i >= 0; i--) {
            // While stack is not empty and the top of the stack is less than or equal to arr[i]
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop(); // Pop elements that are not greater
            }

            // If stack is not empty, the top element is the next greater element
            if (!stack.isEmpty()) {
                result[i] = stack.peek();
            }

            // Push the current element onto the stack
            stack.push(arr[i]);
        }
        return result;
    }

    public static void main(String[] args) {
        int[] arr = {9, 3, 8, 1, 2};
        int[] result = nextGreaterElement(arr);

        // Print the results
        System.out.println("Input array: ");
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println("\nNext Greater Elements: ");
        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}
```

Output:-



```
Input array:
9 3 8 1 2
Next Greater Elements:
-1 8 -1 2 -1
```