Q.1 :

```
public class PeakElement {

    public static int findPeakElement(int[] arr) {
        int n = arr.length;

        // Binary search to find a peak element
        int left = 0, right = n - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            // Check if mid is a peak
            if ((mid == 0 || arr[mid - 1] <= arr[mid]) && (mid == n - 1 || arr[mid + 1] <= arr[mid])) {
                return 1;  // Found a peak element at index mid
            }

            // If the element to the left of mid is greater, move left
            else if (mid > 0 && arr[mid - 1] > arr[mid]) {
                right = mid - 1;
            }

            // If the element to the right of mid is greater, move right
            else {
                left = mid + 1;
            }
        }

        return 0;  // No peak element found
```

```
    }

    public static void main(String[] args) {
        int[] arr = {1, 3, 20, 4, 1, 0};
        int output = findPeakElement(arr);
        System.out.println(output);
    }
}
```

Q.2 :

```
public class CoinChangeWays {

    public static int coinChangeWays(int[] coins, int sum) {
        int[] dp = new int[sum + 1];
        dp[0] = 1; // There is one way to make sum 0 (by not selecting any coin)

        for (int coin : coins) {
            for (int i = coin; i <= sum; i++) {
                dp[i] += dp[i - coin];
            }
        }

        return dp[sum];
    }

    public static void main(String[] args) {
        int[] coins = {1, 2, 5};
        int sum = 5;
        int ways = coinChangeWays(coins, sum);
        System.out.println("Number of ways to make sum " + sum + ": " + ways);
```

```
    }
}
```