NAME: BHARATH V

DEPT: CSBS

1.Buy and Sell stocks

```
public class StockProfit {
    public static int maxProfit(int[] prices) {
        if (prices.length < 2) {
            return 0;
        }
        int minPrice = prices[0];
        int maxProfit = 0;
        for (int i = 1; i < prices.length; i++) {
            int profit = prices[i] - minPrice;
            maxProfit = Math.max(maxProfit, profit);
            minPrice = Math.min(minPrice, prices[i]);
        }
        return maxProfit;
}
    public static void main(String[] args) {
        int[] prices = {1, 2, 3, 4, 5};
        System.out.println("Maximum profit: " + maxProfit(prices));
    }
}
```

OUTPUT:

```
865
```

Complexity:

Time: O(m+n)

Space:O(m+n)

2.Coin Change

```
public class CoinChange {


   public static int coinChange(int[] coins, int amount) {
      int[] dp = new int[amount + 1];
      dp[0] = 0;


      for (int i = 1; i <= amount; i++) {
         dp[i] = Integer.MAX_VALUE;
      }
      for (int coin : coins) {
         for (int i = coin; i <= amount; i++) {
            if (dp[i - coin] != Integer.MAX_VALUE) {
               dp[i] = Math.min(dp[i], dp[i - coin] + 1);
            }
         }
      }
      return dp[amount] == Integer.MAX_VALUE ? -1 : dp[amount];
   }
   public static void main(String[] args) {
      int[] coins = {1, 2, 5};
      int amount = 11;
      System.out.println(coinChange(coins, amount));
   }
}
```

OUTPUT

```
3
```

Complexity:

Time:O(n*m)

Space:O(n)

3.First and Last occurrence of element

```
public class FirstAndLastOccurrence {

    public static void findOccurrences(String text, String word) {
        String[] words = text.split(" ");
        int first = -1, last = -1;

        for (int i = 0; i < words.length; i++) {
            if (words[i].equals(word)) {
                if (first == -1) {
                    first = i;
                }
                last = i;
            }
        }
        if (first == -1) {
            System.out.println("Word not found");
        } else {
            System.out.println("First occurrence: " + first);
            System.out.println("Last occurrence: " + last);
        }
    }
    public static void main(String[] args) {
        String text = "the quick brown fox jumps over the lazy dog the fox is quick";
        String word = "the";

        findOccurrences(text, word);
    }
}
```

OUTPUT:

```
[0,9]
```

Complexity;

Time:O(n)

Space:O(n)

4.First transition Point

```
public class FirstTransitionPoint {


    public static int findTransitionPoint(int[] arr) {
        int low = 0, high = arr.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == 1 && (mid == 0 || arr[mid - 1] == 0)) {
                return mid;
            } else if (arr[mid] == 0) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] arr = {0, 0, 0, 1, 1, 1, 1};
        int transitionPoint = findTransitionPoint(arr);


        if (transitionPoint == -1) {
            System.out.println("No transition point found");
        } else {
            System.out.println("First transition point: " + transitionPoint);
        }
    }
}
```

OUTPUT

```
First transition point: 3
```

Complexity:

Time: O(nlogn)

Space: O(nlogn)

5.Wave Array

```
import java.util.Arrays;

public class WaveArray {

    public static void convertToWave(int[] arr) {

        Arrays.sort(arr);

        for (int i = 0; i < arr.length - 1; i += 2) {

            int temp = arr[i];

            arr[i] = arr[i + 1];

            arr[i + 1] = temp;

        }

    }

    public static void main(String[] args) {

        int[] arr = {3, 6, 5, 10, 7, 20};

        convertToWave(arr);

        System.out.println(Arrays.toString(arr));

    }

}
```

OUTPUT:

```
[6, 3, 10, 5, 20, 7]
```

Complexity:

Time: O(nlogn)

Space: O(1)

6.Remove duplicate from sorted Array

```
public class RemoveDuplicates {
    public static int removeDuplicates(int[] arr) {
        if (arr.length == 0) return 0;
        int index = 1;
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] != arr[i - 1]) {
                arr[index++] = arr[i];
            }
        }
        return index;
    }
    public static void main(String[] args) {
        int[] arr = {1, 1, 2, 2, 3, 4, 4, 5};
        int newLength = removeDuplicates(arr);

        for (int i = 0; i < newLength; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

OUTPUT:

```
1 2 3 4 5
```

Complexity:

Time: O(n)

Space: O(1)