Program:12a

```
public class StockWithFee {
    public static int maxProfit(int[] prices, int fee) {
        int n = prices.length;
        if (n == 0) return 0;
        int cash = 0;
        int hold = -prices[0];
        for (int i = 1; i < n; i++) {
            cash = Math.max(cash, hold + prices[i] - fee);
            hold = Math.max(hold, cash - prices[i]);
        }
        return cash;
    }
}
```

Input:

prices = [1, 3, 2, 8, 4, 9]

fee = 2

Output:

Maximum Profit (with fee): 8

Program:12b

```
public class MinTaps {
    public static int minTaps(int n, int[] ranges) {
        int[] maxReach = new int[n + 1];
        for (int i = 0; i <= n; i++) {
            int left = Math.max(0, i - ranges[i]);
            int right = Math.min(n, i + ranges[i]);
            maxReach[left] = Math.max(maxReach[left], right);
        }
        int taps = 0;
        int currEnd = 0;
        int nextEnd = 0;
        for (int i = 0; i <= n; i++) {
            if (i > nextEnd) return -1; // can't reach position i
            nextEnd = Math.max(nextEnd, maxReach[i]);
            if (i == currEnd) {
                if (currEnd != nextEnd) {
                    taps++;
                    currEnd = nextEnd;
                }
                if (currEnd >= n) break;}
            }
        return currEnd >= n ? taps : -1;
    }
}
```

Input:

n = 5

ranges = [3, 4, 1, 1, 0, 0]

Output:

Minimum Taps Required: 1

Program:12c

```
public class WaterBottles {
    public static int numWaterBottles(int full, int exchange) {
        int total = 0;
        int empty = 0;
        while (full > 0) {
            total += full;
            empty += full;
            full = 0;
            full = empty / exchange;
            empty = empty % exchange;
        }
        return total;
    }
}
```

Input:

numBottles = 9

exchange = 3

Output:

Total Bottles Drunk: 13