Q:

Suppose you are given an array A [1...n] of numbers, which may be positive, negative, or zero, and which are not necessarily integers.
a) Describe and analyze an algorithm that finds the largest sum of elements in a contiguous subarray A [i.. j]
b) Describe and analyze an algorithm that finds the largest product of elements in a contiguous subarray A [i.. j].
Given the one-element array [-374] as input, your first algorithm should return 0, and your second algorithm should return1
a) Describe and analyze an algorithm that finds a contiguous subarray of A of length at most X that has the largest sum.
b) Describe and analyze an algorithm that finds a contiguous subarray of A with the largest sum less than or equal to X.
Note: Perform and provide a dry run of your algorithm for some input value to verify its correctness. Provide the algorithm by using a dynamic programming technique.

A:

a)

The algorithm is given as:

```
int maxSubArraySum(int A[]) {
        int size = A.length;
        int max_so_far = Integer.MIN_VALUE, max_ending_here = 0;

        for (int value : A) {
            max_ending_here = max_ending_here + value;
            if (max_so_far < max_ending_here)
                max_so_far = max_ending_here;
            if (max_ending_here < 0)
                max_ending_here = 0;
        }
        return max_so_far;
    }
```

Time Complexity: O(n)

Space Complexity: O(1)

b)

```
int maxSubarrayProduct(int A[]) {
        int n = A.length;
        int max_ending_here = 1;
        int min_ending_here = 1;
        int max_so_far = 0;
        int flag = 0;
        for (int i = 0; i < n; i++) {
            if (A[i] > 0) {
                max_ending_here = max_ending_here * A[i];
                min_ending_here = min(min_ending_here * A[i], 1);
                flag = 1;
            } else if (A[i] == 0) {
                max_ending_here = 1;
                min_ending_here = 1;
            } else {
                int temp = max_ending_here;
                max_ending_here = max(min_ending_here * A[i], 1);
                min_ending_here = temp * A[i];
            }

            if (max_so_far < max_ending_here)
                max_so_far = max_ending_here;
        }

        if (flag == 0 && max_so_far == 0)
            return 0;
        return max_so_far;
    }
```

Time Complexity: O(n)

Space Complexity: O(1)

NOTE: As per Chegg policy I am allowed to answer specific number of questions (including sub-parts) on a single post. Kindly post the remaining questior separately and I will try to answer them. Sorry for the inconvenience caused.