```
#include <iostream>
#include <unordered_set>
#include <string>
using namespace std;

class Solution
{ public:
    string longestNiceSubstring(string s)
        { if (s.length() < 2) return "";

        unordered_set<char> charSet(s.begin(), s.end());

        for (int i = 0; i < s.length(); i++) {
            if (charSet.count(tolower(s[i])) && charSet.count(toupper(s[i])))
                { continue;
            }

            // If s[i] is not part of a nice substring, split the string
            string left = longestNiceSubstring(s.substr(0, i));
            string right = longestNiceSubstring(s.substr(i + 1));

            return left.length() >= right.length() ? left : right;
        }

        return s;  // The whole string is nice
    }
};
```

```
#include <vector>
#include <algorithm>  // For std::max

class Solution {
public:
    int maxSubArray(std::vector<int>& nums) {
        int maxSum = nums[0], currentSum = nums[0];

        for (size_t i = 1; i < nums.size(); i++) {
            currentSum = std::max(nums[i], currentSum + nums[i]);
            maxSum = std::max(maxSum, currentSum);
        }

        return maxSum;
    }
};
```