DATE:23.12.23 QUIZ-1

1.You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's. Increment the large integer by one and return the resulting array of digits.

CODE:

```
#include <stdio.h>
#include <stdlib.h>

int* increment_large_integer(int* digits, int n) {
   int carry = 1;

   for (int i = n - 1; i >= 0; i--) {
      int total = digits[i] + carry;
      digits[i] = total % 10;
      carry = total / 10;
   }

   if (carry) {

      digits = realloc(digits, (n + 1) * sizeof(int));
      for (int i = n; i > 0; i--) {
         digits[i] = digits[i - 1];
      }
      digits[0] = carry;
   }

   return digits;
}

int main() {
   int input_digits[] = {1, 2, 3};
   int n = sizeof(input_digits) / sizeof(input_digits[0]);


   int* output_digits = increment_large_integer(input_digits, n);
   for (int i = 0; i < n + 1; i++) {
      printf("%d ", output_digits[i]);
   }
   free(output_digits);

   return 0;
}
```

2. You are given an integer array nums. You are initially positioned at the array's first index, and each element in the array represents your maximum jump length at that position. Return true if you can reach the last index, or false otherwise.

PROGRAM CODE:

```
#include <stdio.h>
#include <stdbool.h>

bool canJump(int* nums, int numsSize) {
   int maxReach = 0;

   for (int i = 0; i < numsSize; i++) {
      if (i > maxReach) {

         return false;
      }


      maxReach = (i + nums[i] > maxReach) ? i + nums[i] : maxReach;
      if (maxReach >= numsSize - 1) {
         return true;
      }
   }

   return false;
}

int main() {
   int nums[] = {2, 3, 1, 1, 4};
   int numsSize = sizeof(nums) / sizeof(nums[0]);
   bool result = canJump(nums, numsSize);
   printf(result ? "true\n" : "false\n");

   return 0;
}
```

3. Given an integer array nums, find the subarray with the largest sum, and return its sum. Example 1:
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6

PROGRAM CODE:

```
 #include <stdio.h>

int maxSubArray(int* nums, int numsSize) {
    int maxSum = nums[0];
    int currentSum = nums[0];

    for (int i = 1; i < numsSize; i++) {

        currentSum = (currentSum + nums[i] > nums[i]) ? currentSum + nums[i] : nums[i];
        maxSum = (currentSum > maxSum) ? currentSum : maxSum;
    }

    return maxSum;
}

int main() {
    int nums[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4};
    int numsSize = sizeof(nums) / sizeof(nums[0]);

    int result = maxSubArray(nums, numsSize);
    printf("%d\n", result);

    return 0;
}
```