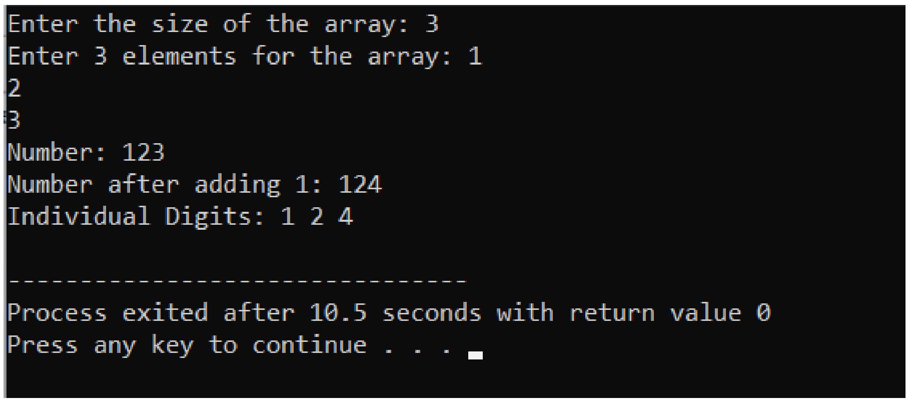
1 . You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return the resulting array of digits.

CODE

```
#include <stdio.h>
void storeDigits(int arr[], int n) {
   int i;
   int num = 0;
   for (i = 0; i < n; i++)
   {
      num = num * 10 + arr[i];
   }
   printf("Number: %d\n", num);
   num = num + 1;
   printf("Number after adding 1: %d\n", num);
   int temp = num;
   int count = 0;
   while (temp > 0)
   {
      temp = temp / 10;
      count++;
   }
   int digits[count];
   temp = num;
   for (i = count - 1; i >= 0; i--)
   {
      digits[i] = temp % 10;
      temp = temp / 10;
   }
   printf("Individual Digits: ");
```

```
  for (i = 0; i < count; i++)
  {
      printf("%d ", digits[i]);
  }
printf("\n");
}
int main()
{
    int n,i;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements for the array: ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    storeDigits(arr, n);
    return 0;
}
```



```
Enter the size of the array: 3
Enter 3 elements for the array: 1
2
3
Number: 123
Number after adding 1: 124
Individual Digits: 1 2 4

--------------------------------
Process exited after 10.5 seconds with return value 0
Press any key to continue . . .
```

2. You are given an integer array nums. You are initially positioned at the array&#39;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.

Input: nums = [2,3,1,1,4]

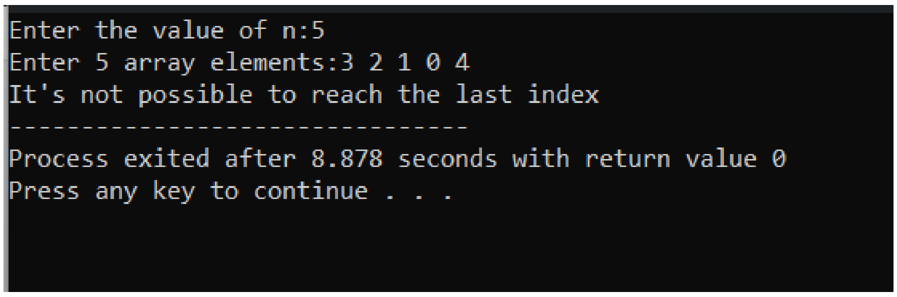
Output: true

Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.

CODE

```
#include <stdio.h>

#include <stdbool.h>

bool Jump(int nums[], int numsSize) {

   int maxReach = 0,i;

   for ( 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 n,i;

   printf("Enter the value of n:");

   scanf("%d",&n);

   int nums[n];

   printf("Enter %d array elements:",n);

   for(i=0;i<n;i++)

   {
```

```
            scanf("%d",&nums[i]);
    }
    int numsSize = sizeof(nums) / sizeof(nums[0]);
     bool result = Jump(nums, numsSize);
if (result)
{
  printf("It's possible to reach the last index");
}
else
{
 printf("It's not possible to reach the last index");
 }
return 0;
}
```



```
Enter the value of n:5
Enter 5 array elements:3 2 1 0 4
It's not possible to reach the last index
--------------------------------
Process exited after 8.878 seconds with return value 0
Press any key to continue . . .
```

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

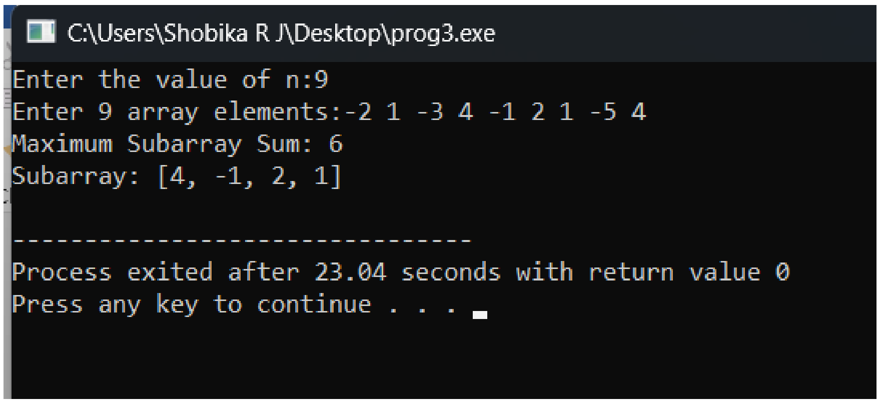
Explanation: The subarray [4,-1,2,1] has the largest sum 6.

CODE

```
#include <stdio.h>
void printSubarray(int nums[], int start, int end)
{
    int i;
    printf("Subarray: [");
    for (i = start; i <= end; i++) {
        printf("%d", nums[i]);
```

```c
        if (i < end) {
            printf(", ");
        }
    }
    printf("]\n");
}

void findMaxSubArray(int nums[], int numsSize) {
    int i;
    int maxSum = nums[0];
    int currentSum = nums[0];
    int start = 0;
    int end = 0;
    int tempStart = 0;

    for (i = 1; i < numsSize; i++) {
        if (nums[i] > currentSum + nums[i]) {
            currentSum = nums[i];
            tempStart = i;
        } else {
            currentSum = currentSum + nums[i];
        }

        if (currentSum > maxSum) {
            maxSum = currentSum;
            start = tempStart;
            end = i;
        }
    }
```

```
   printf("Maximum Subarray Sum: %d\n", maxSum);
    printSubarray(nums, start, end);
}


int main() {
    int n, i;
    printf("Enter the value of n:");
    scanf("%d", &n);
    int nums[n];
    printf("Enter %d array elements:", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }

    findMaxSubArray(nums, n);

    return 0;
}
```



C:\Users\Shobika R J\Desktop\prog3.exe

```
Enter the value of n:9
Enter 9 array elements:-2 1 -3 4 -1 2 1 -5 4
Maximum Subarray Sum: 6
Subarray: [4, -1, 2, 1]

--------------------------------
Process exited after 23.04 seconds with return value 0
Press any key to continue . . .
```