**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.**

**CODE :**

#include <stdio.h>

#include <stdlib.h>

int main() {

int maxSize = 100;

int input[maxSize];

int inputSize;

printf("Enter the number of digits: ");

scanf("%d", &inputSize);

if (inputSize > maxSize || inputSize <= 0) {

printf("Invalid input size.\n");

return 1;

}

printf("Enter the digits (from most significant to least significant): ");

for (int i = 0; i < inputSize; i++) {

scanf("%d", &input[i]);

}

input[inputSize - 1] += 1;

int carry = 0;

for (int i = inputSize - 1; i >= 0; i--) {

input[i] += carry;

carry = input[i] / 10;

input[i] %= 10;

}

if (carry > 0) {

int\* result = (int\*)malloc((inputSize + 1) \* sizeof(int));

result[0] = carry;

for (int i = 0; i < inputSize; i++) {

result[i + 1] = input[i];

}

printf("Output: [");

for (int i = 0; i < inputSize + 1; i++) {

printf("%d", result[i]);

if (i != inputSize) {

printf(", ");

}

}

printf("]\n");

free(result);

} else {

printf("Output: [");

for (int i = 0; i < inputSize; i++) {

printf("%d", input[i]);

if (i != inputSize - 1) {

printf(", ");

}

}

printf("]\n");

}

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.**

**CODE :**

#include <stdio.h>

#include <stdbool.h>

int main() {

int nums1[] = {2, 3, 1, 1, 4};

int nums2[] = {3, 2, 1, 0, 4};

int size1 = sizeof(nums1) / sizeof(nums1[0]);

int size2 = sizeof(nums2) / sizeof(nums2[0]);

int maxReach = 0;

for (int i = 0; i < size1; i++) {

if (i > maxReach) {

printf("Example 1: false\n");

return 0;

}

maxReach = (i + nums1[i] > maxReach) ? i + nums1[i] : maxReach;

if (maxReach >= size1 - 1) {

printf("Example 1: true\n");

break;

}

}

maxReach = 0;

for (int i = 0; i < size2; i++) {

if (i > maxReach) {

printf("Example 2: false\n");

return 0;

}

maxReach = (i + nums2[i] > maxReach) ? i + nums2[i] : maxReach;

if (maxReach >= size2 - 1) {

printf("Example 2: true\n");

break;

}

}

return 0;

}

**3.  Given an integer array nums, find the subarray with the largest sum, and return its sum.**

**CODE :**

#include <stdio.h>

int main() {

int nums[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4};

int size = sizeof(nums) / sizeof(nums[0]);

int maxSum = nums[0]; // Initialize with the first element

int currentSum = nums[0];

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

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

maxSum = (currentSum > maxSum) ? currentSum : maxSum;

}

printf("Output: %d\n", maxSum);

return 0;

}