1.

#include <stdbool.h>

#include <stdio.h>

#include <conio.h>

bool canJump(int\* nums, int numsSize) {

int maxReach = 0;

int i;

for ( i = 0; i < numsSize; i++) {

if (i > maxReach) {

return false; // If the current index is not reachable

}

maxReach = fmax(maxReach, i + nums[i]);

if (maxReach >= numsSize - 1) {

return true; // If the last index is reachable

}

}

return false;

}

int main() {

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

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

bool result1 = canJump(nums1, size1);

printf("Example 1: %s\n", result1 ? "true" : "false");

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

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

bool result2 = canJump(nums2, size2);

printf("Example 2: %s\n", result2 ? "true" : "false");

return 0;

}



2.

#include <stdio.h>

#include <limits.h>

int maxSubArray(int\* nums, int numsSize) {

int maxSum = INT\_MIN;

int currentSum = 0;

int i;

for ( i = 0; i < numsSize; i++) {

currentSum = fmax(nums[i], currentSum + nums[i]);

maxSum = fmax(maxSum, currentSum);

}

return maxSum;

}

int main() {

int nums1[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4};

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

int result1 = maxSubArray(nums1, size1);

printf("Example 1: %d\n", result1);

int nums2[] = {1};

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

int result2 = maxSubArray(nums2, size2);

printf("Example 2: %d\n", result2);

int nums3[] = {5, 4, -1, 7, 8};

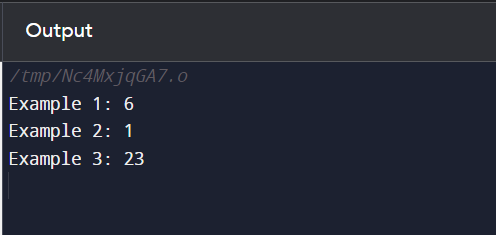
int size3 = sizeof(nums3) / sizeof(nums3[0]);

int result3 = maxSubArray(nums3, size3);

printf("Example 3: %d\n", result3);

return 0;

}



3.

#include <stdio.h>

#include <stdlib.h>

// Function to increment the large integer represented as an array of digits

int\* increment(int\* digits, int size, int\* resultSize) {

// Increment the least significant digit

digits[size - 1] += 1;

int i,j;

// Check for carry and propagate if necessary

for (i = size - 1; i > 0 && digits[i] == 10; i--) {

digits[i] = 0;

digits[i - 1] += 1;

}

// If the most significant digit got a carry, resize the array

if (digits[0] == 10) {

digits[0] = 0;

(\*resultSize) = size + 1;

int\* result = (int\*)malloc((\*resultSize) \* sizeof(int));

result[0] = 1;

for ( j = 1; j < (\*resultSize); j++) {

result[j] = digits[j - 1];

}

return result;

}

(\*resultSize) = size;

return digits;

}

// Function to print an array of integers

void printArray(int\* array, int size) {

printf("[");

int i;

for ( i = 0; i < size; i++) {

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

if (i < size - 1) {

printf(",");

}

}

printf("]\n");

}

int main() {

int digits1[] = {1, 2, 3};

int size1 = sizeof(digits1) / sizeof(digits1[0]);

int resultSize1;

int\* result1 = increment(digits1, size1, &resultSize1);

printf("Example 1:\nInput: ");

printArray(digits1, size1);

printf("Output: ");

printArray(result1, resultSize1);

free(result1);

int digits2[] = {9};

int size2 = sizeof(digits2) / sizeof(digits2[0]);

int resultSize2;

int\* result2 = increment(digits2, size2, &resultSize2);

printf("Example 2:\nInput: ");

printArray(digits2, size2);

printf("Output: ");

printArray(result2, resultSize2);

free(result2);

return 0;

}

