PRACTICAL NO. 4

NAME: DIVYA SONTAKKE

SECTION: A4\_B1 ROLL NO.: 11

**Aim:** Implement maximum sum of subarray for the given scenario of resource allocation using

the divide and conquer approach.

**Problem Statement:**

A project requires allocating resources to various tasks over a period of time. Each task requires

a certain amount of resources, and you want to maximize the overall efficiency of resource

usage. You're given an array of resources where resources[i] represents the amount of resources required for the ith task. Your goal is to find the contiguous subarray of tasks that maximizes the total resources utilized without exceeding a given resource constraint.

Handle cases where the total resources exceed the constraint by adjusting the subarray window

accordingly. Your implementation should handle various cases, including scenarios where there's no feasible subarray given the constraint and scenarios where multiple subarrays yield

the same maximum resource utilization.

**CODE**

#include <stdio.h>

#include <stdlib.h>

#define LARGE\_N 100000

int arr10[LARGE\_N];

int maxSubArraySum(int arr[], int n, int constraint) {

if (n == 0 || constraint == 0) return 0;

int left = 0, right = 0;

int sum = 0;

int max\_sum = 0;

while (right < n) {

sum += arr[right];

while (sum > constraint && left <= right) {

sum -= arr[left];

left++;

}

if (sum <= constraint && sum > max\_sum) {

max\_sum = sum;

}

right++;

}

return max\_sum;

}

int main() {

for (int i = 0; i < LARGE\_N; i++) {

arr10[i] = i + 1;

}

int arr1[] = {2, 1, 3, 4};

int constraint1 = 5;

int n1 = sizeof(arr1) / sizeof(arr1[0]);

printf("Test 1: Max sum = %d\n", maxSubArraySum(arr1, n1, constraint1));

int arr2[] = {2, 2, 2, 2};

int constraint2 = 4;

int n2 = sizeof(arr2) / sizeof(arr2[0]);

printf("Test 2: Max sum = %d\n", maxSubArraySum(arr2, n2, constraint2));

int arr3[] = {1, 5, 2, 3};

int constraint3 = 5;

int n3 = sizeof(arr3) / sizeof(arr3[0]);

printf("Test 3: Max sum = %d\n", maxSubArraySum(arr3, n3, constraint3));

int arr4[] = {6, 7, 8};

int constraint4 = 5;

int n4 = sizeof(arr4) / sizeof(arr4[0]);

printf("Test 4: Max sum = %d\n", maxSubArraySum(arr4, n4, constraint4));

int arr5[] = {1, 1, 1};

int constraint5 = 5;

int n5 = sizeof(arr5) / sizeof(arr5[0]);

printf("Test 5: Max sum = %d\n", maxSubArraySum(arr5, n5, constraint5));

int arr6[] = {1, 1, 1, 1, 1};

int constraint6 = 4;

int n6 = sizeof(arr6) / sizeof(arr6[0]);

printf("Test 6: Max sum = %d\n", maxSubArraySum(arr6, n6, constraint6));

int arr7[] = {4, 2, 3, 1};

int constraint7 = 5;

int n7 = sizeof(arr7) / sizeof(arr7[0]);

printf("Test 7: Max sum = %d\n", maxSubArraySum(arr7, n7, constraint7));

int \*arr8 = NULL;

int constraint8 = 10;

int n8 = 0;

printf("Test 8: Max sum = %d\n", maxSubArraySum(arr8, n8, constraint8));

int arr9[] = {1, 2, 3};

int constraint9 = 0;

int n9 = sizeof(arr9) / sizeof(arr9[0]);

printf("Test 9: Max sum = %d\n", maxSubArraySum(arr9, n9, constraint9));

int constraint10 = 1000000000;

printf("Test 10: Max sum = %d\n", maxSubArraySum(arr10, LARGE\_N, constraint10));

return 0;

}

**OUTPUT**

