1. Implementation of finding the maximum and minimum element using divide and conquer strategy. Analyze and describe how the divide and conquer strategy is better when compared to traditional approach.

#include <stdio.h>

#include <stdlib.h>

void merge(int arr[], int left, int mid, int right) {

int i, j, k;

int n1 = mid - left + 1;

int n2 = right - mid;

int leftArr[n1], rightArr[n2];

for (i = 0; i < n1; i++)

leftArr[i] = arr[left + i];

for (j = 0; j < n2; j++)

rightArr[j] = arr[mid + 1 + j];

i = 0;

j = 0;

k = left;

while (i < n1 && j < n2) {

if (leftArr[i] <= rightArr[j]) {

arr[k] = leftArr[i];

i++;

}

else {

arr[k] = rightArr[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = leftArr[i];

i++;

k++;

}

while (j < n2) {

arr[k] = rightArr[j];

j++;

k++;

}

}

void mergeSort(int arr[], int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

int main() {

int n, i;

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

scanf("%d", &n);

int arr[n];

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

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

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

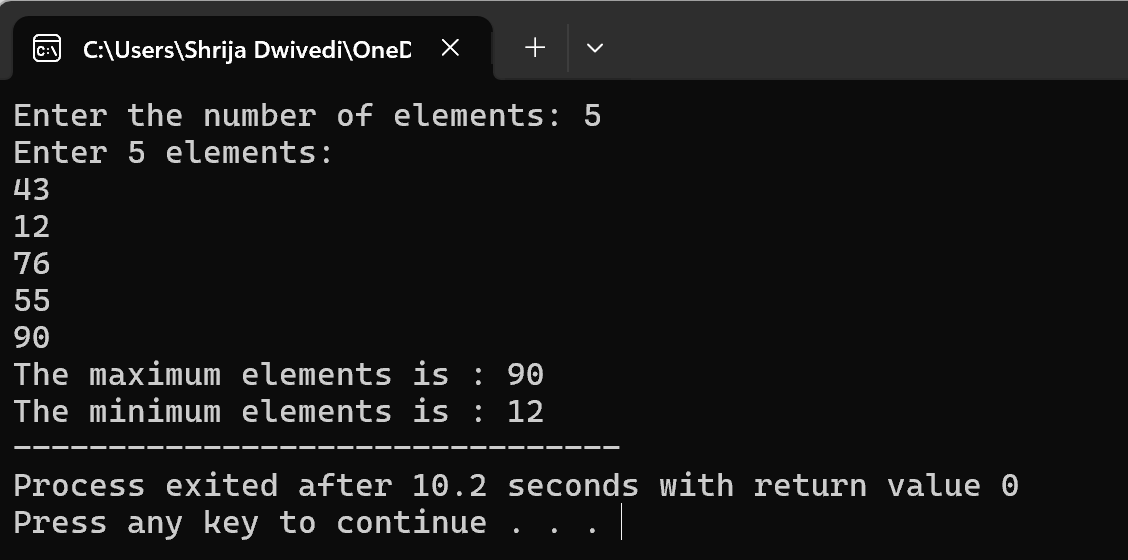
mergeSort(arr, 0, n - 1);

printf("The maximum elements is : %d \n",arr[n-1]);

printf("The minimum elements is : %d",arr[0]);

return 0;

}



1. Divide and conquer: Implementation of maximum-subarray problem.

#include <stdio.h>

#include <limits.h>

int max(int a, int b) {

return (a > b) ? a : b;

}

int max3(int a, int b, int c) {

return max(max(a, b), c);

}

int maxCrossingSum(int arr[], int l, int m, int r) {

int i,sum = 0;

int left\_sum = INT\_MIN;

for (i = m; i >= l; i--) {

sum += arr[i];

if (sum > left\_sum)

left\_sum = sum;

}

sum = 0;

int right\_sum = INT\_MIN;

for (i = m + 1; i <= r; i++) {

sum += arr[i];

if (sum > right\_sum)

right\_sum = sum;

}

return left\_sum + right\_sum;

}

int maxSubArraySum(int arr[], int l, int r) {

if (l == r)

return arr[l];

int m = (l + r) / 2;

return max3(maxSubArraySum(arr, l, m),maxSubArraySum(arr, m + 1, r),maxCrossingSum(arr, l, m, r));

}

int main() {

int n;

printf("Enter number of elements: ");

scanf("%d", &n);

int i,arr[n];

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

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

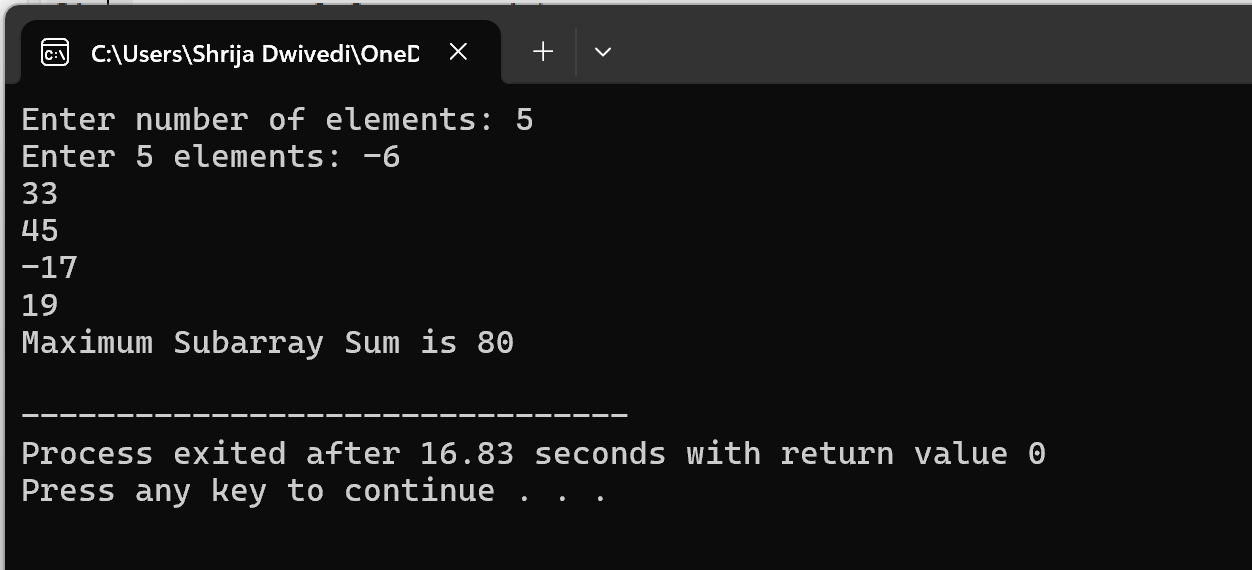
scanf("%d", &arr[i]);

int max\_sum = maxSubArraySum(arr, 0, n - 1);

printf("Maximum Subarray Sum is %d\n", max\_sum);

return 0;

}



1. Write a program to implement Hash Table with Chaining Method. Perform the Insert, Search and Delete operation on Hash Table by taking user choices as:

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

struct node {

int data;

struct node\* next;

};

struct node\* hashTable[SIZE];

int hashFunction(int key) {

return key % SIZE;

}

void insert(int key) {

int index = hashFunction(key);

struct node\* newnode = (struct node\*)malloc(sizeof(struct node));

newnode->data = key;

newnode->next = hashTable[index];

hashTable[index] = newnode;

printf("%d is inserted \n", key);

}

void search(int key) {

int index = hashFunction(key);

struct node\* t = hashTable[index];

while (t != NULL) {

if (t->data == key) {

printf("%d is found at index %d \n", key, index);

return;

}

t = t->next;

}

printf("%d is not found \n", key);

}

// Delete operation

void deletenode(int key) {

int index = hashFunction(key);

struct node\* t = hashTable[index];

struct node\* prev = NULL;

while (t != NULL) {

if (t->data == key) {

if (prev == NULL) {

hashTable[index] = t->next;

} else {

prev->next = t->next;

}

free(t);

printf("%d is deleted \n", key);

return;

}

prev = t;

t = t->next;

}

printf("%d is not found, therefore deletion canoot be done \n", key);

}

void display() {

int i;

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

printf("Index %d: ", i);

struct node\* t = hashTable[i];

while (t != NULL) {

printf("%d -> ", t->data);

t= t->next;

}

printf("NULL\n");

}

}

int main() {

int ch, key;

do {

printf("Hash Table Operations \n");

printf("1. Insert\n");

printf("2. Search\n");

printf("3. Delete\n");

printf("4. Display\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &ch);

switch (ch) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &key);

insert(key);

break;

case 2:

printf("Enter value to search: ");

scanf("%d", &key);

search(key);

break;

case 3:

printf("Enter value to delete: ");

scanf("%d", &key);

deletenode(key);

break;

case 4:

display();

break;

case 5:

printf("Exited Successfully");

break;

default:

printf("Invalid choice. Try again.\n");

}

}while(ch!=5);

return 0;

}

