1. Min max

#include <stdio.h>

void findMinMax(int arr[], int low, int high, int \*min, int \*max) {

int mid, min1, max1, min2, max2;

if (low == high) {

\*min = \*max = arr[low];

return;

}

if (high == low + 1) {

if (arr[low] < arr[high]) {

\*min = arr[low];

\*max = arr[high];

} else {

\*min = arr[high];

\*max = arr[low];

}

return;

}

mid = (low + high) / 2;

findMinMax(arr, low, mid, &min1, &max1);

findMinMax(arr, mid + 1, high, &min2, &max2);

if (min1 < min2)

\*min = min1;

else

\*min = min2;

if (max1 > max2)

\*max = max1;

else

\*max = max2;

}

int main() {

int n, i, min, max;

printf("number of elements: ");

scanf("%d", &n);

int arr[n];

printf(" elements are : ");

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

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

}

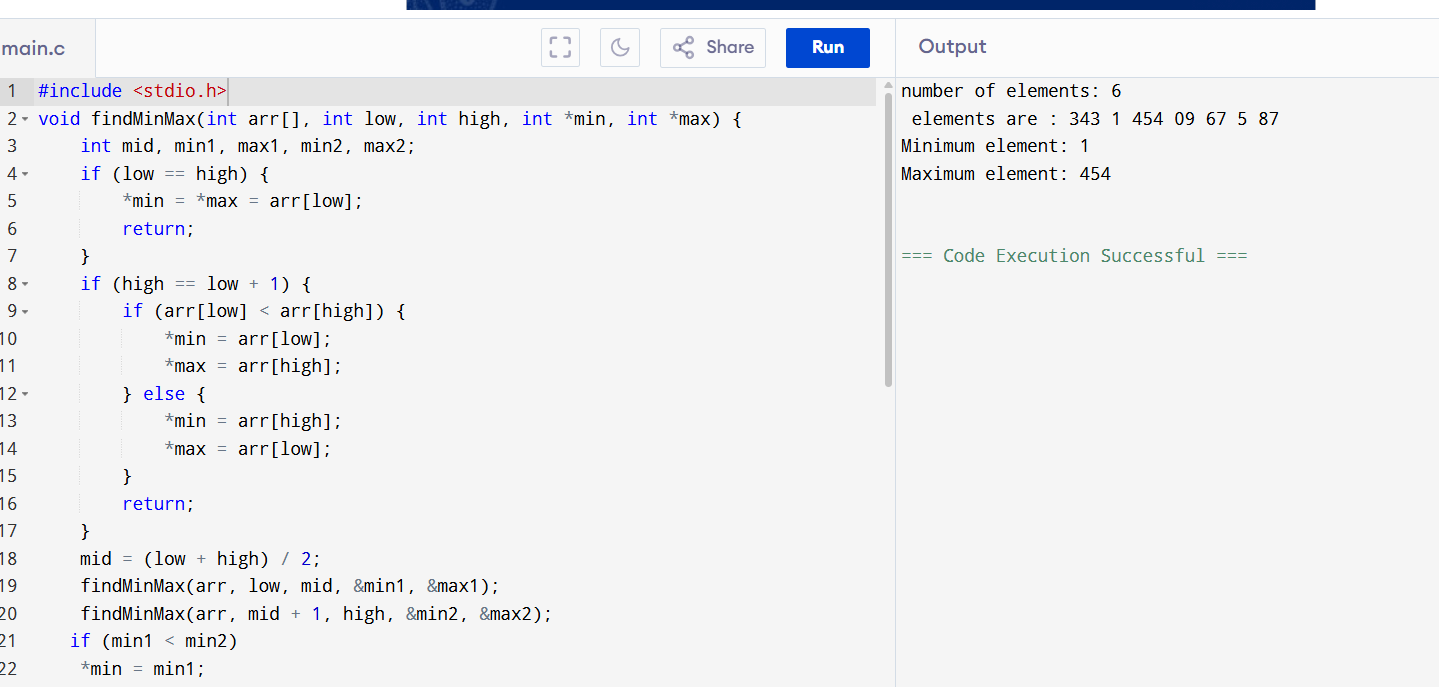
findMinMax(arr, 0, n - 1, &min, &max);

printf("Minimum element: %d\n", min);

printf("Maximum element: %d\n", max);

return 0;

}



1. Hash table code

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

struct Node {

int data;

struct Node\* next;

};

struct Node\* hashTable[SIZE];

int hash(int key) {

return key % SIZE;

}

void insert(int key) {

int index = hash(key);

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = key;

newNode->next = hashTable[index];

hashTable[index] = newNode;

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

}

void search(int key) {

int index = hash(key);

struct Node\* temp = hashTable[index];

while (temp != NULL) {

if (temp->data == key) {

printf("%d found in hash table!\n", key);

return;

}

temp = temp->next;

}

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

}

void delete(int key) {

int index = hash(key);

struct Node\* temp = hashTable[index];

struct Node\* prev = NULL;

while (temp != NULL) {

if (temp->data == key) {

if (prev == NULL) {

hashTable[index] = temp->next;

} else {

prev->next = temp->next;

}

free(temp);

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

return;

}

prev = temp;

temp = temp->next;

}

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

}

void display() {

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

printf("[%d] -> ", i);

struct Node\* temp = hashTable[i];

while (temp != NULL) {

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

temp = temp->next;

}

printf("NULL\n");

}

}

int main() {

int choice, key;

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

hashTable[i] = NULL;

}

printf("\n Hash Table Menu \n");

printf("1. Insert\n2. Search\n3. Delete\n4. Display\n5. Exit\n");

while (1) {

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf(" value to insert: ");

scanf("%d", &key);

insert(key);

break;

case 2:

printf(" value to search: ");

scanf("%d", &key);

search(key);

break;

case 3:

printf("value to delete: ");

scanf("%d", &key);

delete(key);

break;

case 4:

display();

break;

case 5:

printf("Exit\n");

exit(0);

default:

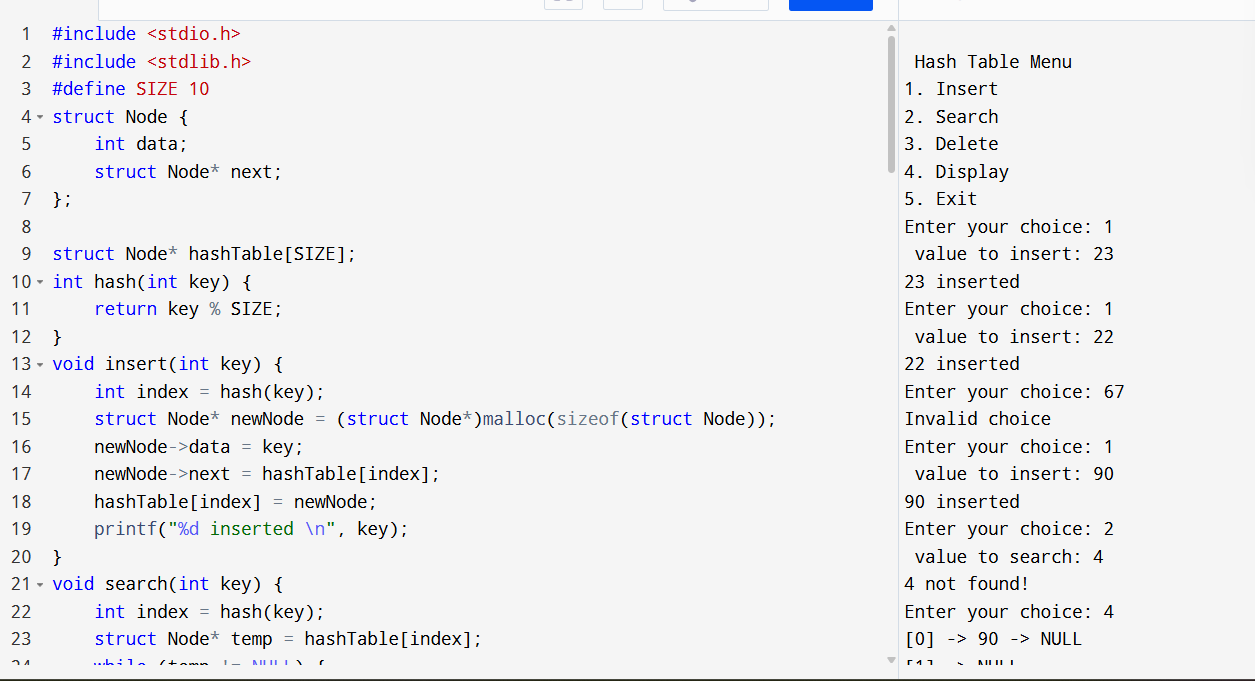
printf("Invalid choice\n");

}

}

return 0;

}



A screenshot of a computer program

AI-generated content may be incorrect.

3-subarray

#include <stdio.h>

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

if (a >= b && a >= c) return a;

else if (b >= a && b >= c) return b;

else return c;

}

int maxCrossingSubarray(int arr[], int low, int mid, int high, int \*cross\_low, int \*cross\_high) {

int sum = 0;

int left\_sum = -1000000;

int max\_left = mid;

for (int i = mid; i >= low; i--) {

sum += arr[i];

if (sum > left\_sum) {

left\_sum = sum;

max\_left = i;

}

}

sum = 0;

int right\_sum = -1000000;

int max\_right = mid + 1;

for (int i = mid + 1; i <= high; i++) {

sum += arr[i];

if (sum > right\_sum) {

right\_sum = sum;

max\_right = i;

}

}

\*cross\_low = max\_left;

\*cross\_high = max\_right;

return left\_sum + right\_sum;

}

int maxSubarray(int arr[], int low, int high, int \*start, int \*end) {

if (low == high) {

\*start = low;

\*end = high;

return arr[low];

}

int mid = (low + high) / 2;

int left\_start, left\_end, right\_start, right\_end, cross\_start, cross\_end;

int left\_sum = maxSubarray(arr, low, mid, &left\_start, &left\_end);

int right\_sum = maxSubarray(arr, mid + 1, high, &right\_start, &right\_end);

int cross\_sum = maxCrossingSubarray(arr, low, mid, high, &cross\_start, &cross\_end);

if (left\_sum >= right\_sum && left\_sum >= cross\_sum) {

\*start = left\_start;

\*end = left\_end;

return left\_sum;

}

else if (right\_sum >= left\_sum && right\_sum >= cross\_sum) {

\*start = right\_start;

\*end = right\_end;

return right\_sum;

}

else {

\*start = cross\_start;

\*end = cross\_end;

return cross\_sum;

}

}

int main() {

int n;

printf("Enter number of elements: ");

scanf("%d", &n);

int arr[n];

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

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

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

}

int start, end;

maxSubarray(arr, 0, n - 1, &start, &end);

printf("Maximum Subarray: ");

for (int i = start; i <= end; i++) {

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

}

printf("\n");

return 0;

}

