Day 4: DATA STRUCTURES PROGRAMS

1.write a c program to implement binary search tree.

Program:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int data) {

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

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

struct Node\* insert(struct Node\* root, int data) {

if (root == NULL) {

return createNode(data);

}

if (data < root->data) {

root->left = insert(root->left, data);

} else if (data > root->data) {

root->right = insert(root->right, data);

}

return root;

}

void inOrderTraversal(struct Node\* root) {

if (root != NULL) {

inOrderTraversal(root->left);

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

inOrderTraversal(root->right);

}

}

int main() {

struct Node\* root = NULL;

int n, value;

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

scanf("%d", &n);

printf("Enter the elements:\n");

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

scanf("%d", &value);

root = insert(root, value);

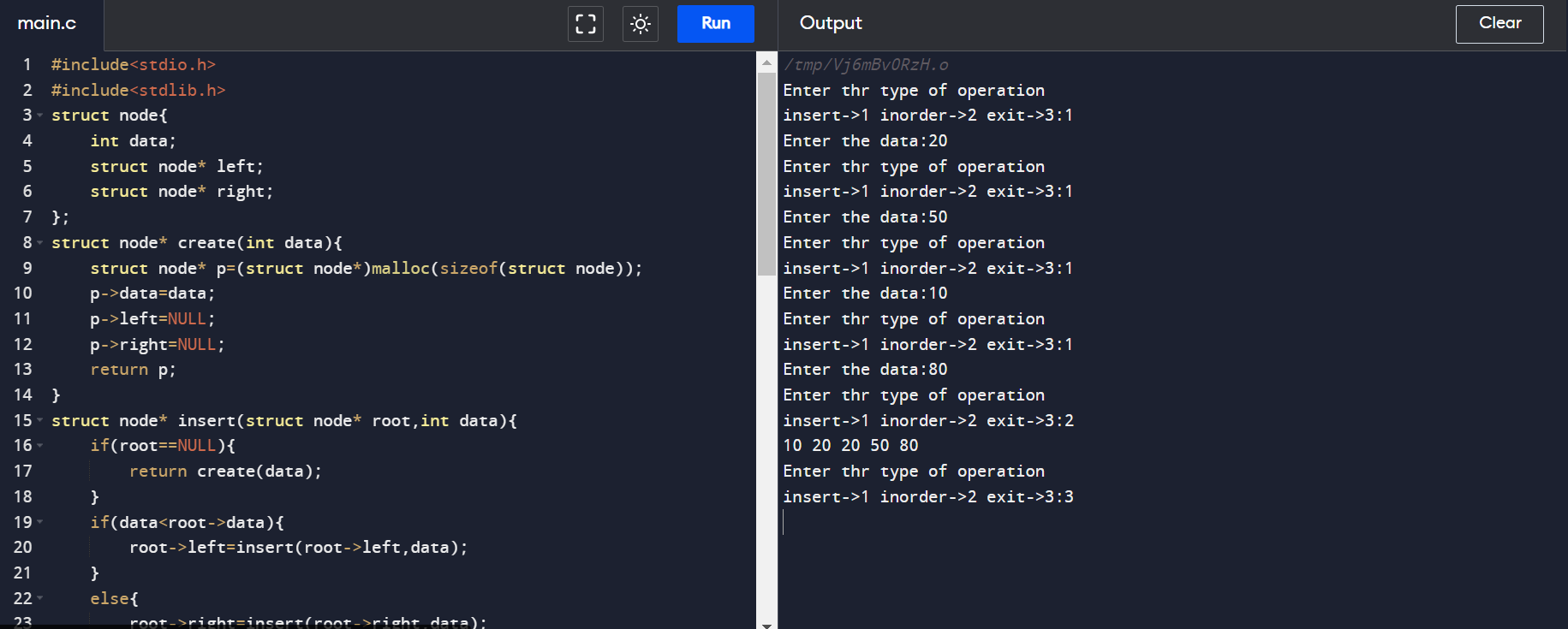
}

printf("In-order traversal of the binary search tree: ");

inOrderTraversal(root);

printf("\n");

return 0;

}2.write a c program to implement AVL tree.

Program:

4.write a c program to implement bubble sort.

Program:

#include<stdio.h>

int main(){

int n;

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

scanf("%d", &n);

int a[n];

printf("Enter the elements: ");

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

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

}

int c=1;

while(c<n){

for(int j=0;j<n-c;j++){

if(a[j]>a[j+1]){

int temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

c++;

}

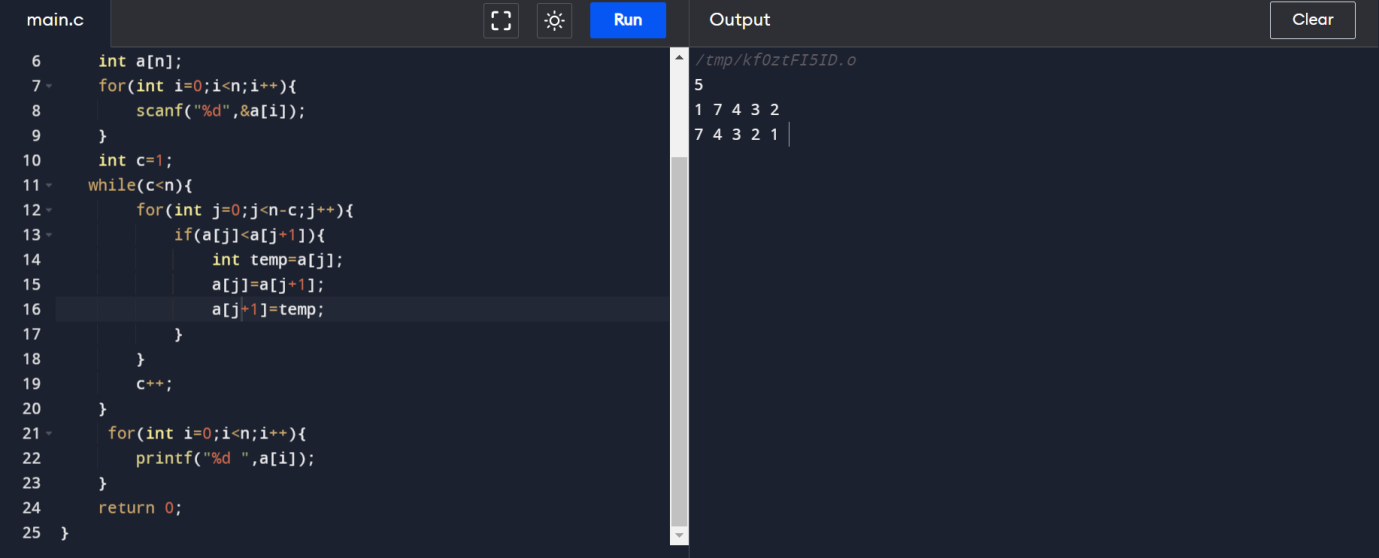
for(int i=0;i<n;i++){

printf("%d\t", a[i]);

}

return 0;

}



Sample input:

Enter the size of array: 7

Enter the elements: 23 45 78 22 11 7 96

Output:

7 11 22 23 45 78 96

5.write a c program to implement insertion sort.

Program:

#include <stdio.h>

int main(void)

{

int n, i, j, temp;

int arr[64];

printf("Enter number of elements\n");

scanf("%d", &n);

printf("Enter %d integers\n", n);

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

{

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

}

for (i = 1; i< n; i++)

{

j = i;

while (j > 0 &&arr[j - 1] >arr[j])

{

temp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = temp;

j--;

}

}

printf("Sorted list in ascending order:\n");

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

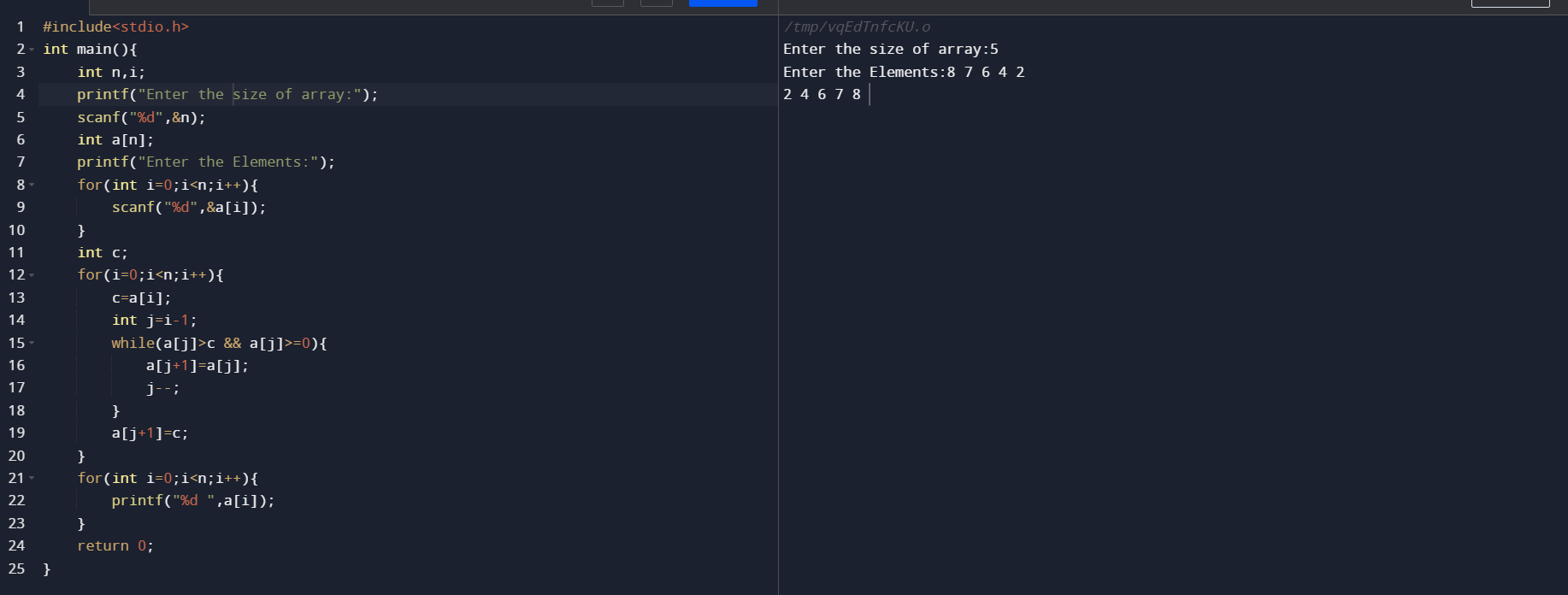
{

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

}

return 0;

}



6.write a c program to implement selection sort.

Program:

#include<stdio.h>

int main(){

int i, j, count, temp, number[25];

printf("How many numbers u are going to enter?: ");

scanf("%d",&count);

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

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

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

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

for(j=i+1;j<count;j++){

if(number[i]>number[j]){

temp=number[i];

number[i]=number[j];

number[j]=temp;

}

}

}

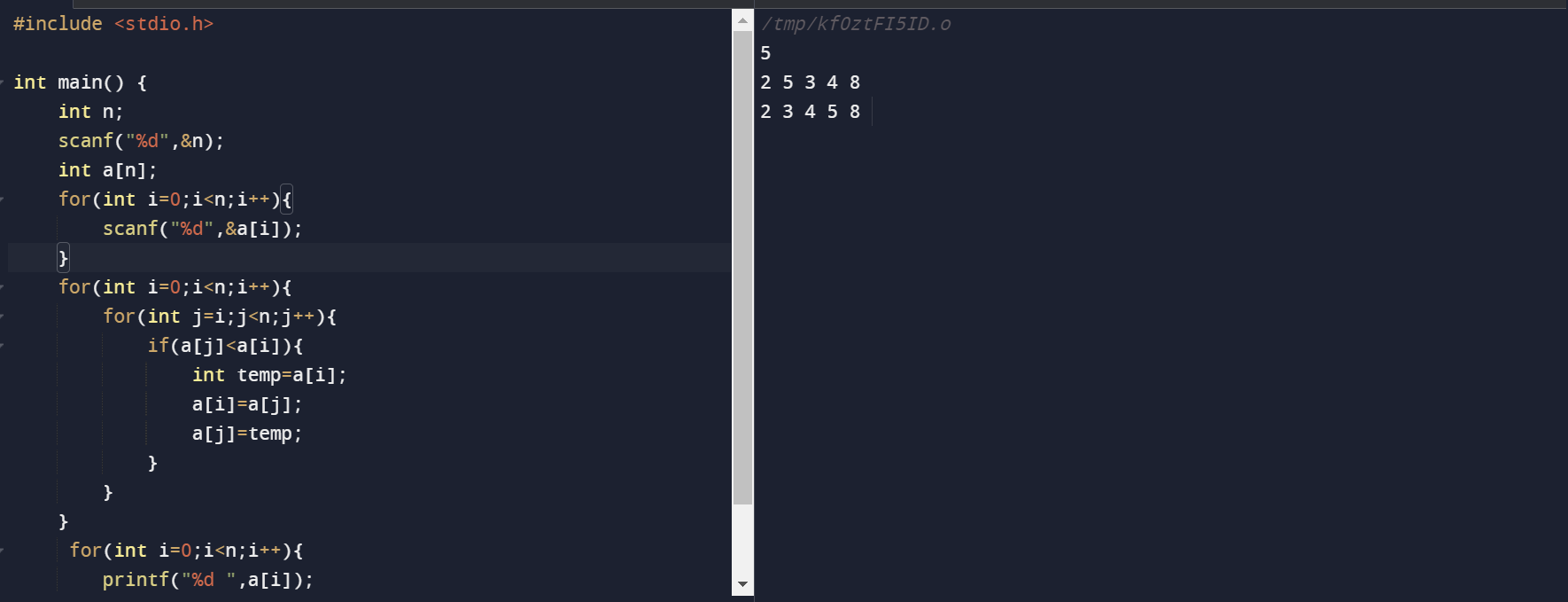
printf("Sorted elements: ");

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

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

   return 0;

}



7.write a c program to implement quick sort.

Program:

#include<stdio.h>

void quicksort(int number[25],int first,int last){

int i, j, pivot, temp;

if(first<last){

pivot=first;

i=first;

j=last;

while(i<j){

while(number[i]<=number[pivot]&&i<last)

i++;

while(number[j]>number[pivot])

j--;

if(i<j){

temp=number[i];

number[i]=number[j];

number[j]=temp;

}

}

temp=number[pivot];

number[pivot]=number[j];

number[j]=temp;

quicksort(number,first,j-1);

quicksort(number,j+1,last);

}

}

int main(){

int i, count, number[25];

printf("How many elements are u going to enter?: ");

scanf("%d",&count);

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

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

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

quicksort(number,0,count-1);

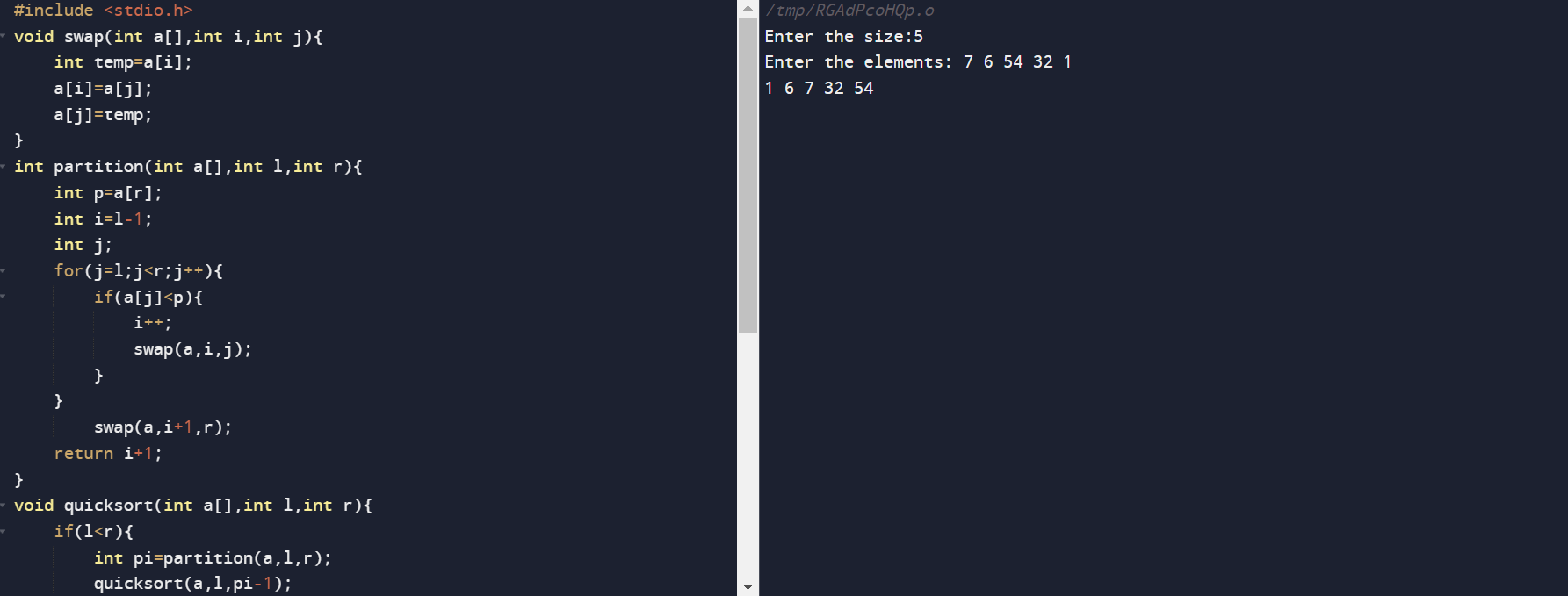
printf("Order of Sorted elements: ");

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

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

   return 0;

}



8.write a c program to implement merge sort.

Program:

#include <stdio.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 L[n1], R[n2];

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

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

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

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

i = 0;

j = 0;

k = left;

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

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

while (i< n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[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;

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

scanf("%d", &n);

int arr[n];

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

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

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

}

mergeSort(arr, 0, n - 1);

printf("Sorted array:\n");

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

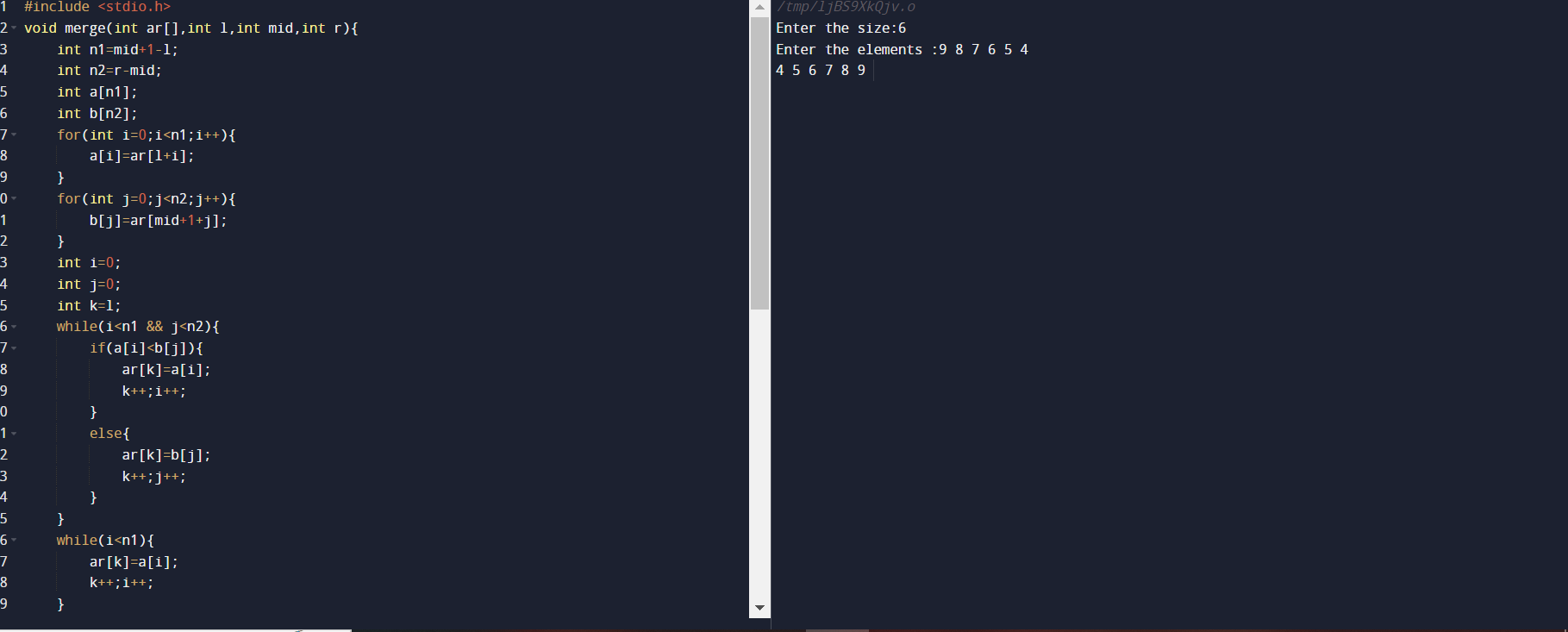
printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

.