Write a C/C++ program to perform the following operations on arrays:

a) Traverse an array.

b) Find the number of element (Length) of an array

c) Search an element from the array.

d) Find the largest element from the array.

e) Find the smallest element from the array

f) Insert an element at a given position in the array.

g) Delete an element from a given position in the array.

h) Search for an element using linear search.

#include <stdio.h>

#define MAX\_SIZE 100

// Function prototypes

void traverseArray(int arr[], int n);

int arrayLength(int n);

int searchElement(int arr[], int n, int key);

int findLargest(int arr[], int n);

int findSmallest(int arr[], int n);

int insertElement(int arr[], int \*n, int pos, int element);

int deleteElement(int arr[], int \*n, int pos);

int linearSearch(int arr[], int n, int key);

int main() {

int arr[MAX\_SIZE];

int n, choice, pos, element, key, index;

printf("Enter number of elements (max %d): ", MAX\_SIZE);

scanf("%d", &n);

if(n > MAX\_SIZE) {

printf("Size exceeds max limit!\n");

return 1;

}

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

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

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

do {

printf("\nMenu:\n");

printf("1. Traverse Array\n");

printf("2. Find Length of Array\n");

printf("3. Search an Element\n");

printf("4. Find Largest Element\n");

printf("5. Find Smallest Element\n");

printf("6. Insert Element at Position\n");

printf("7. Delete Element from Position\n");

printf("8. Linear Search for Element\n");

printf("9. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1:

printf("Array elements: ");

traverseArray(arr, n);

break;

case 2:

printf("Length of array: %d\n", arrayLength(n));

break;

case 3:

printf("Enter element to search: ");

scanf("%d", &key);

index = searchElement(arr, n, key);

if(index != -1)

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

else

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

break;

case 4:

printf("Largest element is %d\n", findLargest(arr, n));

break;

case 5:

printf("Smallest element is %d\n", findSmallest(arr, n));

break;

case 6:

printf("Enter position to insert (0 to %d): ", n);

scanf("%d", &pos);

if(pos < 0 || pos > n) {

printf("Invalid position!\n");

break;

}

printf("Enter element to insert: ");

scanf("%d", &element);

if(insertElement(arr, &n, pos, element))

printf("Element inserted.\n");

else

printf("Insertion failed.\n");

break;

case 7:

printf("Enter position to delete (0 to %d): ", n-1);

scanf("%d", &pos);

if(pos < 0 || pos >= n) {

printf("Invalid position!\n");

break;

}

if(deleteElement(arr, &n, pos))

printf("Element deleted.\n");

else

printf("Deletion failed.\n");

break;

case 8:

printf("Enter element to linear search: ");

scanf("%d", &key);

index = linearSearch(arr, n, key);

if(index != -1)

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

else

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

break;

case 9:

printf("Exiting...\n");

break;

default:

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

}

} while(choice != 9);

return 0;

}

// a) Traverse an array

void traverseArray(int arr[], int n) {

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

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

printf("\n");

}

// b) Find the number of elements (Length)

int arrayLength(int n) {

return n;

}

// c) Search an element in array (returns index or -1)

int searchElement(int arr[], int n, int key) {

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

if(arr[i] == key)

return i;

}

return -1;

}

// d) Find largest element

int findLargest(int arr[], int n) {

int max = arr[0];

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

if(arr[i] > max)

max = arr[i];

}

return max;

}

// e) Find smallest element

int findSmallest(int arr[], int n) {

int min = arr[0];

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

if(arr[i] < min)

min = arr[i];

}

return min;

}

// f) Insert element at given position

int insertElement(int arr[], int \*n, int pos, int element) {

if(\*n >= MAX\_SIZE) // Check for overflow

return 0;

for(int i = \*n; i > pos; i--) {

arr[i] = arr[i-1];

}

arr[pos] = element;

(\*n)++;

return 1;

}

// g) Delete element from given position

int deleteElement(int arr[], int \*n, int pos) {

if(\*n <= 0)

return 0;

for(int i = pos; i < \*n - 1; i++) {

arr[i] = arr[i+1];

}

(\*n)--;

return 1;

}

// h) Linear search (returns index or -1)

int linearSearch(int arr[], int n, int key) {

return searchElement(arr, n, key);

}

### Part D: Application-Based Problems

### 5. Find the “Leader Elements” in an array:

### An element is a leader if it is greater than all the elements to its right side. The rightmost

### element is always a leader.

### 

#include <stdio.h>

void findLeaders(int arr[], int n) {

int maxFromRight = arr[n - 1];

int leaders[n]; // To store leaders

int leaderCount = 0;

// The rightmost element is always a leader

leaders[leaderCount++] = maxFromRight;

// Traverse from second last element to the beginning

for (int i = n - 2; i >= 0; i--) {

if (arr[i] > maxFromRight) {

maxFromRight = arr[i];

leaders[leaderCount++] = maxFromRight;

}

}

// Leaders were stored from right to left, print in reverse order

printf("Leader elements are: ");

for (int i = leaderCount - 1; i >= 0; i--) {

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

}

printf("\n");

}

int main() {

int n;

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

scanf("%d", &n);

int arr[n];

printf("Enter array elements: ");

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

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

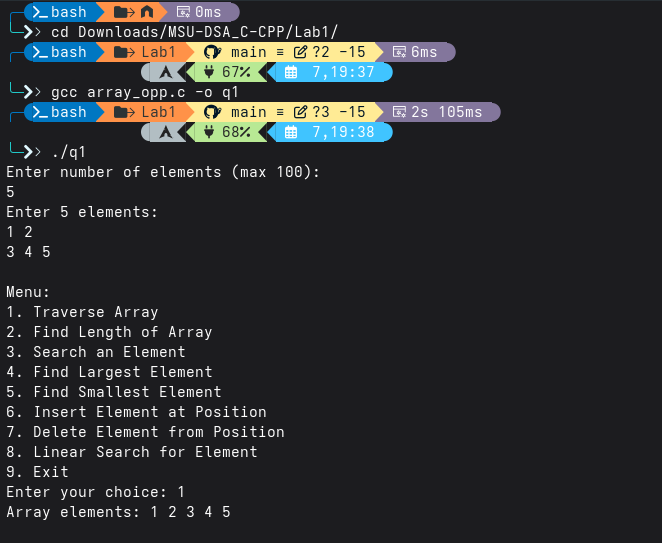
findLeaders(arr, n);

return 0;

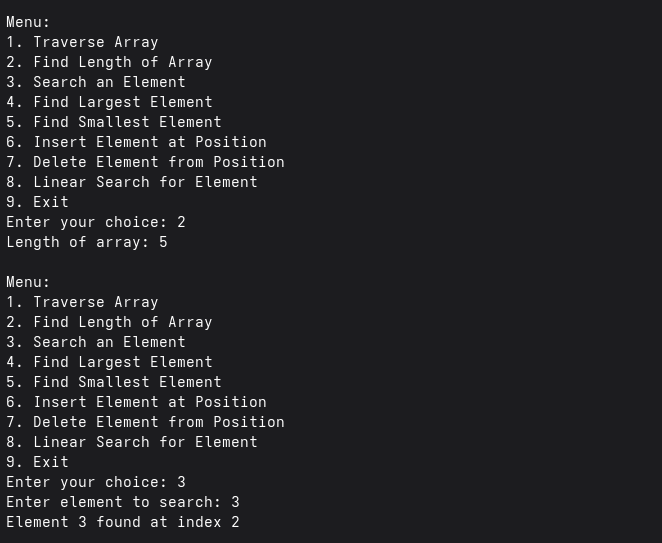
}

Write a C/C++ program to perform the following operations on arrays:

a) Traverse an array.

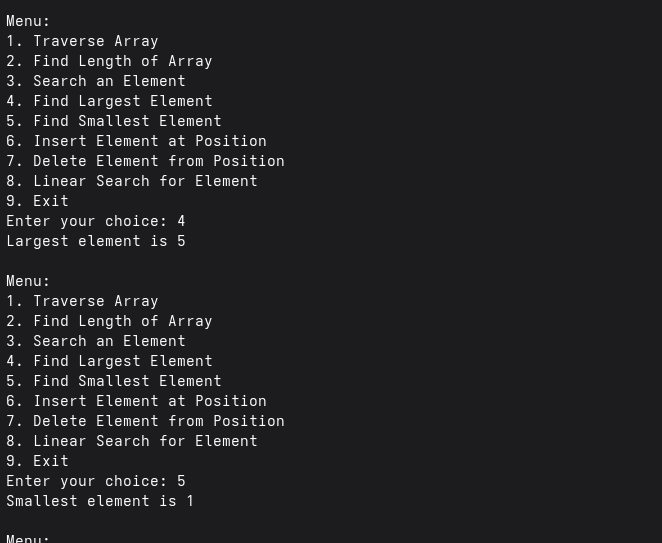
b) Find the number of element (Length) of an array

c) Search an element from the array.

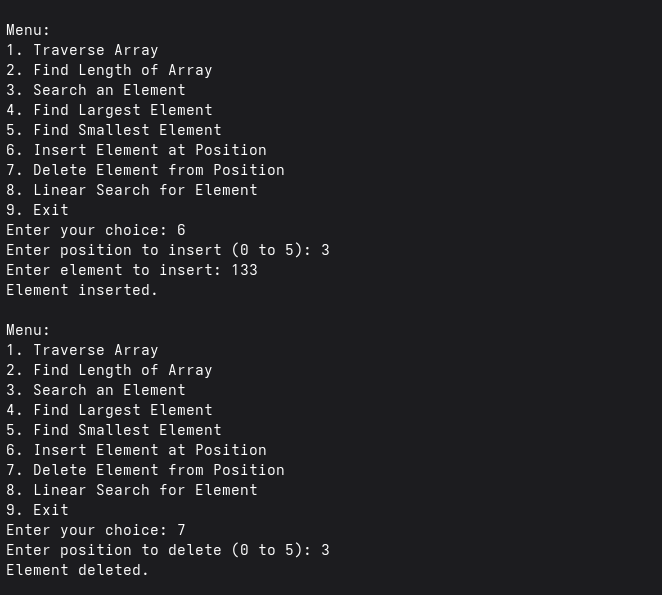


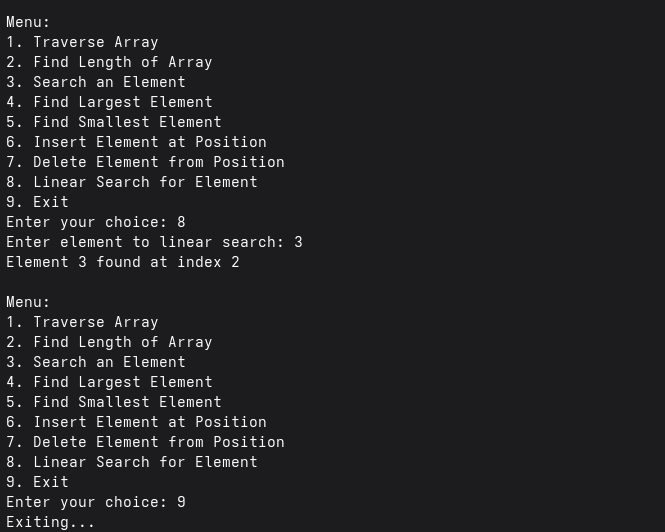
d) Find the largest element from the array.

e) Find the smallest element from the array

f) Insert an element at a given position in the array.

g) Delete an element from a given position in the array.

h) Search for an element using linear search.



5. Find the “Leader Elements” in an array:

An element is a leader if it is greater than all the elements to its right side. The rightmost

element is always a leader.

