1. Write a C program to find largest and smallest element of an array using pointers.

**Code:**

#include <stdio.h>

void findLargestAndSmallest(int \*arr, int size, int \*largest, int \*smallest) {

\*largest = \*smallest = \*arr;

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

if (\*(arr + i) > \*largest) {

\*largest = \*(arr + i);

}

if (\*(arr + i) < \*smallest) {

\*smallest = \*(arr + i);

}

}

}

int main() {

int n;

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

scanf("%d", &n);

int arr[n];

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

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

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

}

int largest, smallest;

findLargestAndSmallest(arr, n, &largest, &smallest);

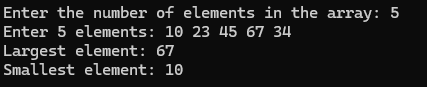
printf("Largest element: %d\n", largest);

printf("Smallest element: %d\n", smallest);

return 0;

}

**Output:**



1. Write a C program to find sum of elements in an array using pointers.

**Code:**

#include <stdio.h>

int main() {

int n, i, sum = 0;

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

scanf("%d", &n);

int arr[n];

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

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

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

}

int \*ptr = arr; // Pointer to the first element of the array

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

sum += \*(ptr + i);

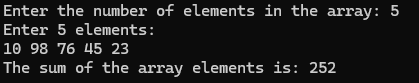
}

printf("The sum of the array elements is: %d\n", sum);

return 0;

}

**Output:**



1. Write a C program to search the target element in an array using pointers.

**Code:**

#include <stdio.h>

int main() {

int n, i, target;

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

scanf("%d", &n);

int arr[n];

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

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

scanf("%d", arr + i);

}

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

scanf("%d", &target);

int \*ptr = arr;

int found = 0;

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

if(\*(ptr + i) == target) {

printf("Element %d found at position %d.", target, i + 1);

found = 1;

break;

}

}

if(!found) {

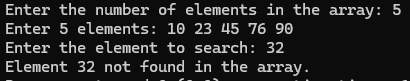
printf("Element %d not found in the array.", target);

}

return 0;

}

**Output:**



1. Write a C program to reverse an array of elements using pointers.

**Code:**

#include <stdio.h>

void reverseArray(int \*arr, int size) {

int \*start = arr;

int \*end = arr + size - 1;

int temp;

while (start < end) {

temp = \*start;

\*start = \*end;

\*end = temp;

start++;

end--;

}

}

int main() {

int n;

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

scanf("%d", &n);

int arr[n];

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

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

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

}

reverseArray(arr, n);

printf("Reversed array: ");

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

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

}

printf(" ");

return 0;

}

**Output:**



1. Write a C program to print transpose of a matrix using pointers.

**Code:**

#include <stdio.h>

int main() {

int rows, cols;

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

scanf("%d", &rows);

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

scanf("%d", &cols);

int matrix[rows][cols];

int transpose[cols][rows];

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

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

for (int j = 0; j < cols; j++) {

scanf("%d", (\*(matrix + i) + j));

}

}

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

for (int j = 0; j < cols; j++) {

\*(\*(transpose + j) + i) = \*(\*(matrix + i) + j);

}

}

printf("Transpose of the matrix:\n");

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

for (int j = 0; j < rows; j++) {

printf("%d ", \*(\*(transpose + i) + j));

}

printf("\n");

}

return 0;

}

**Output:**

