Name **Khuram Khalid**

Class **SP22 BCS 026**

Assignment **Data Structure**

Comsats University Islamabad, Vehari

Problem 01

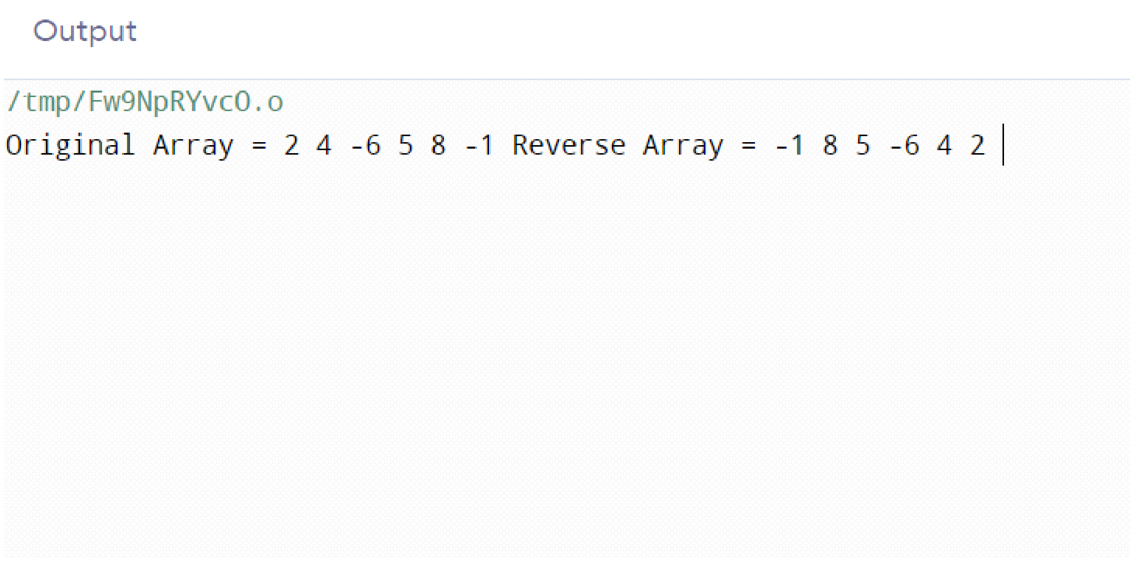
Program to reverse an array using pointers

Code

```
#include <iostream>
using namespace std;

// Function to swap two memory contents
void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Function to reverse the array through pointers
void reverse(int array[], int array_size)
{

    // pointer1 pointing at the beginning of the array
    int *pointer1 = array,

        // pointer2 pointing at end of the array
        *pointer2 = array + array_size - 1;
    while (pointer1 < pointer2) {
        swap(pointer1, pointer2);
        pointer1++;
        pointer2--;
    }
}
```

```
// Function to print the array
void print(int* array, int array_size)
{

    // Length pointing at end of the array
    int *length = array + array_size,

        // Position pointing to the beginning of the array
        *position = array;
    cout << "Array = ";
    for (position = array; position < length; position++)
        cout << *position << " ";
}

// Driver function
int main()
{

    // Array to hold the values
    int array[] = { 2, 4, -6, 5, 8, -1 };

    cout << "Original ";
    print(array, 6);

    cout << "Reverse ";
    reverse(array, 6);
    print(array, 6);
    return 0;
}
```

Output


Output

```
/tmp/Fw9NpRYvcO.o
Original Array = 2 4 -6 5 8 -1 Reverse Array = -1 8 5 -6 4 2
```

Problem

C program to update the values of variables in main using pointers

Code

```
#include <stdio.h>

void update(int *a, int *b) {
    // Update the values pointed to by a and b
    int temp_a = *a + *b;
    int temp_b = *a - *b;
    *a = temp_a;
    *b = temp_b;
}

int main() {
    int a, b;
    int *pa = &a, *pb = &b;

    scanf("%d %d", &a, &b);
    update(pa, pb);
    printf("%d\n%d", a, b);

    return 0;
}
```

Problem

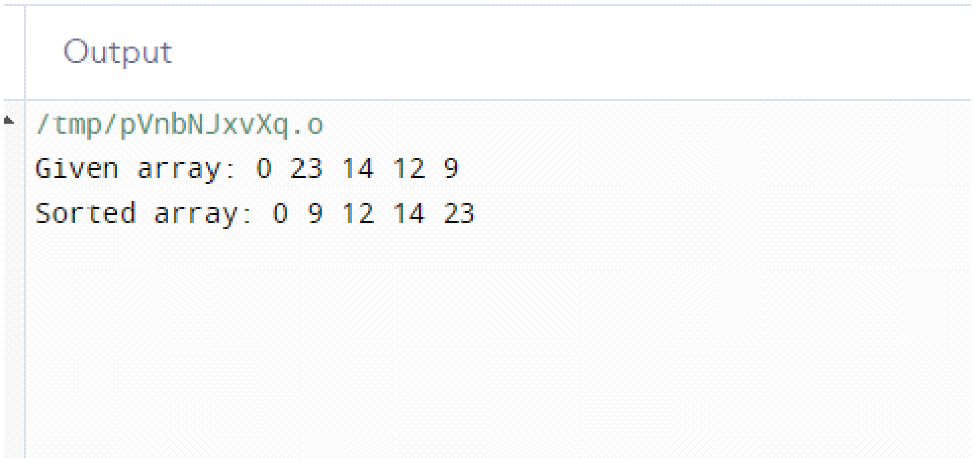
C program to sort an array using pointers

Code

```
#include <stdio.h>

// Function to sort the numbers using pointers
void sort(int n, int* ptr) {
    int i, j, t;

    // Sort the numbers using pointers
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (*(ptr + j) < *(ptr + i)) {
                t = *(ptr + i);
                *(ptr + i) = *(ptr + j);
                *(ptr + j) = t;
            }
        }
    }
}
```

```
int main() {
    int n = 5;
    printf("Given array: ");
    int arr[] = {0, 23, 14, 12, 9};

    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]); // Print each element of the given array
    }

    sort(n, arr); // Call the sort function to sort the array

    printf("\nSorted array: ");

    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]); // Print each element of the sorted array
    }

    return 0;
}
```

Output



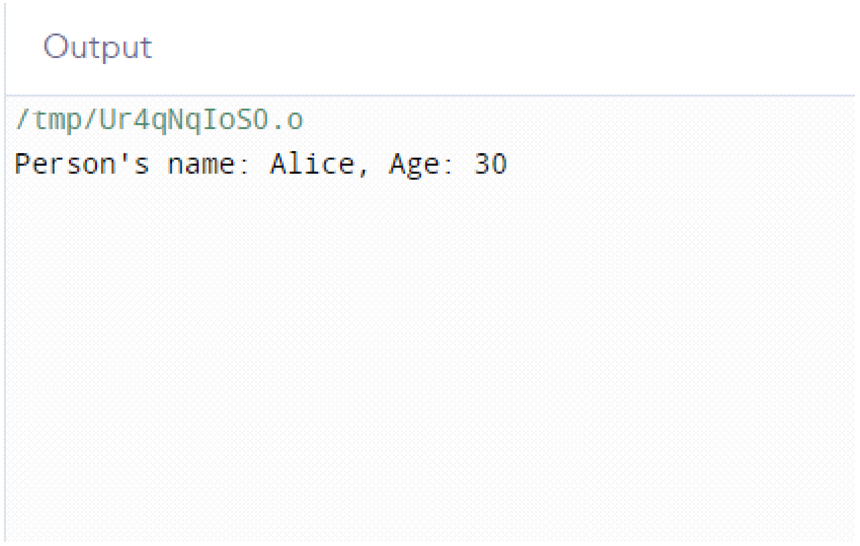
```
/tmp/pVnbNJxvXq.o
Given array: 0 23 14 12 9
Sorted array: 0 9 12 14 23
```

Problem

C++ Pointers to Object

```
#include <iostream>
using namespace std;

class Person {
public:
    string name;
    int age;
};

int main() {
    Person person;
    Person *personPtr = &person;
    personPtr->name = "Alice";
    personPtr->age = 30;
    cout << "Person's name: " << personPtr->name << ", Age: " << personPtr->age << endl;
    return 0;
}
```

Output



```
/tmp/Ur4qNqIoS0.o
Person's name: Alice, Age: 30
```

Problem

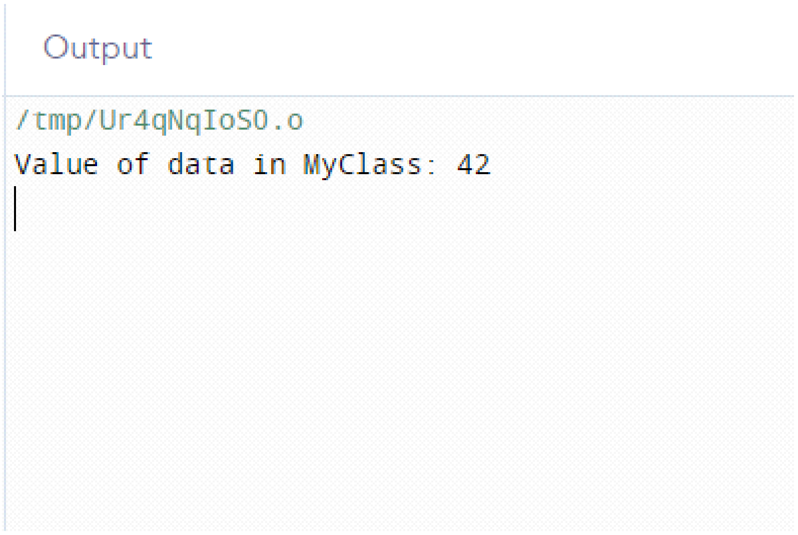
Pointer to Member of Class

Code

```
#include <iostream>
using namespace std;

class MyClass {
public:
    int data = 42;
};

int main() {
    MyClass obj;
    int MyClass::*memberPtr = &MyClass::data;
    cout << "Value of data in MyClass: " << obj.*memberPtr << endl;
    return 0;
```

}

Output



```
/tmp/Ur4qNqIoS0.o
Value of data in MyClass: 42
```

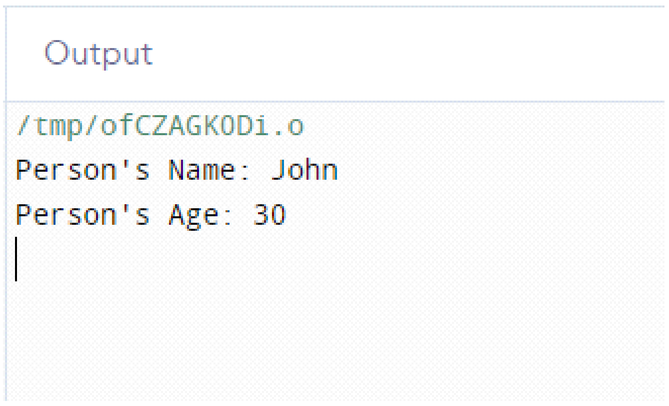
Program

Handling Structures with Pointers:

Code

```
#include <stdio.h>
#include <string.h>

// Define a structure for a person
struct Person {
    char name[50];
    int age;
};

int main() {
    // Declare a structure variable
    struct Person person;
    // Create a pointer to the structure
    struct Person *ptr = &person;

    // Initialize structure members using pointers
    strcpy(ptr->name, "John");
    ptr->age = 30;

    // Access and print structure members using pointers
    printf("Person's Name: %s\n", ptr->name);
    printf("Person's Age: %d\n", ptr->age);

    return 0;
}
```

Output



Output

```
/tmp/ofCZAGKODi.o
Person's Name: John
Person's Age: 30
```

Program

C program to multiply tw matrices using pointers

Code

```
/**
 * C program to multiply two matrix using pointers
 */
```

```
#include <stdio.h>

#define ROW 3
#define COL 3


/* Function declarations */
void matrixInput(int mat[][COL]);
void matrixPrint(int mat[][COL]);
void matrixMultiply(int mat1[][COL], int mat2[][COL], int res[][COL]);



int main()
{
    int mat1[ROW][COL];
    int mat2[ROW][COL];
    int product[ROW][COL];


    /*
     * Input elements in matrices.
     */
    printf("Enter elements in first matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat1);

    printf("Enter elements in second matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat2);
    // Call function to multiply both matrices
    matrixMultiply(mat1, mat2, product);



    // Print product of both matrix
    printf("Product of both matrices is : \n");
    matrixPrint(product);

    return 0;
}




/**
 * Function to input elements in matrix from user.
 *
 * @mat     Two-dimensional array to store user input.
 */
void matrixInput(int mat[][COL])
{
    int row, col;

    for (row = 0; row < ROW; row++)
    {
        for (col = 0; col < COL; col++)
        {
            scanf("%d", (*(mat + row) + col));
        }
    }
```

```
}




/**
 * Function to print elements in a two-dimensional array.
 *
 * @mat     Two-dimensional array to print.
 */
void matrixPrint(int mat[][COL])
{
    int row, col;

    for (row = 0; row < ROW; row++)
    {
        for (col = 0; col < COL; col++)
        {
            printf("%d ", *(*(mat + row) + col));
        }

        printf("\n");
    }
}




/**
 * Function to multiply two matrices.
 *
 * @mat1    First matrix
 * @mat2    Second matrix
 * @res     Resultant matrix to store product of both matrices.
 */
void matrixMultiply(int mat1[][COL], int mat2[][COL], int res[][COL])
{
    int row, col, i;
    int sum;



    for (row = 0; row < ROW; row++)
    {
        for (col = 0; col < COL; col++)
        {
            sum = 0;

            /*
             * Find sum of product of each elements of
             * rows of first matrix and columns of second
             * matrix.
             */
            for (i = 0; i < COL; i++)
            {
                sum += (*(*(mat1 + row) + i)) * (*(*(mat2 + i) + col));
            }



            /*
             * Store sum of product of row of first matrix
             * and column of second matrix to resultant matrix.
             */
            *(*(res + row) + col) = sum;
        }
    }
}
```

Problem

C++ Program to pass C++ Argument to Function

Call-By-Value

Call-By-Reference with a Pointer Argument

Call-By-Reference with a Reference Argument

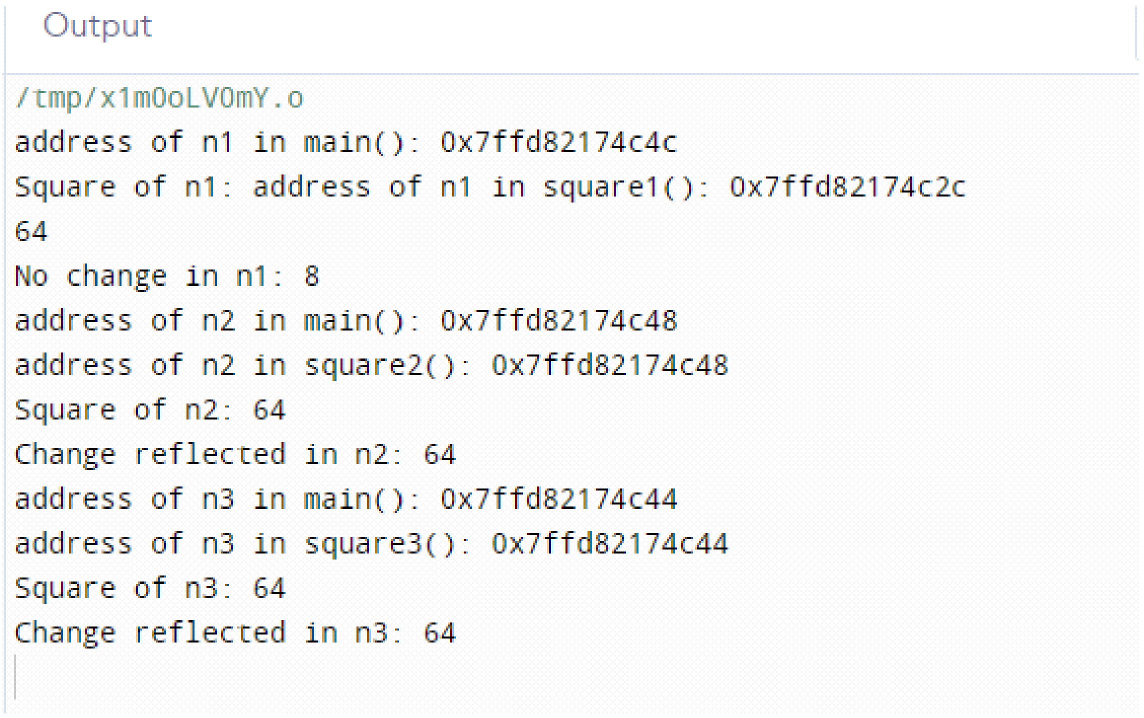
Code

```
// C++ program to illustrate call-by-methods

#include <bits/stdc++.h>
using namespace std;

// Pass-by-Value
int square1(int n)
{
            // Address of n in square1() is not the same as n1 in
            // main()
            cout << "address of n1 in square1(): " << &n << "\n";

            // clone modified inside the function
            n *= n;
            return n;
}
// Pass-by-Reference with Pointer Arguments
```

```
void square2(int* n)
{
	// Address of n in square2() is the same as n2 in main()
	cout << "address of n2 in square2(): " << n << "\n";

	// Explicit de-referencing to get the value pointed-to
	*n *= *n;
}
// Pass-by-Reference with Reference Arguments
void square3(int& n)
{
	// Address of n in square3() is the same as n3 in main()
	cout << "address of n3 in square3(): " << &n << "\n";

	// Implicit de-referencing (without '*')
	n *= n;
}
void geeks()
{
	// Call-by-Value
	int n1 = 8;
	cout << "address of n1 in main(): " << &n1 << "\n";
	cout << "Square of n1: " << square1(n1) << "\n";
	cout << "No change in n1: " << n1 << "\n";

	// Call-by-Reference with Pointer Arguments
	int n2 = 8;
	cout << "address of n2 in main(): " << &n2 << "\n";
	square2(&n2);
	cout << "Square of n2: " << n2 << "\n";
```

```
        cout << "Change reflected in n2: " << n2 << "\n";

        // Call-by-Reference with Reference Arguments
        int n3 = 8;
        cout << "address of n3 in main(): " << &n3 << "\n";
        square3(n3);
        cout << "Square of n3: " << n3 << "\n";
        cout << "Change reflected in n3: " << n3 << "\n";
}
// Driver program
int main() { geeks(); }
```

Output



Output

```
/tmp/x1m0oLVOmY.o
address of n1 in main(): 0x7ffd82174c4c
Square of n1: address of n1 in square1(): 0x7ffd82174c2c
64
No change in n1: 8
address of n2 in main(): 0x7ffd82174c48
address of n2 in square2(): 0x7ffd82174c48
Square of n2: 64
Change reflected in n2: 64
address of n3 in main(): 0x7ffd82174c44
address of n3 in square3(): 0x7ffd82174c44
Square of n3: 64
Change reflected in n3: 64
```

Problem

Program to Reverse a String using Pointers

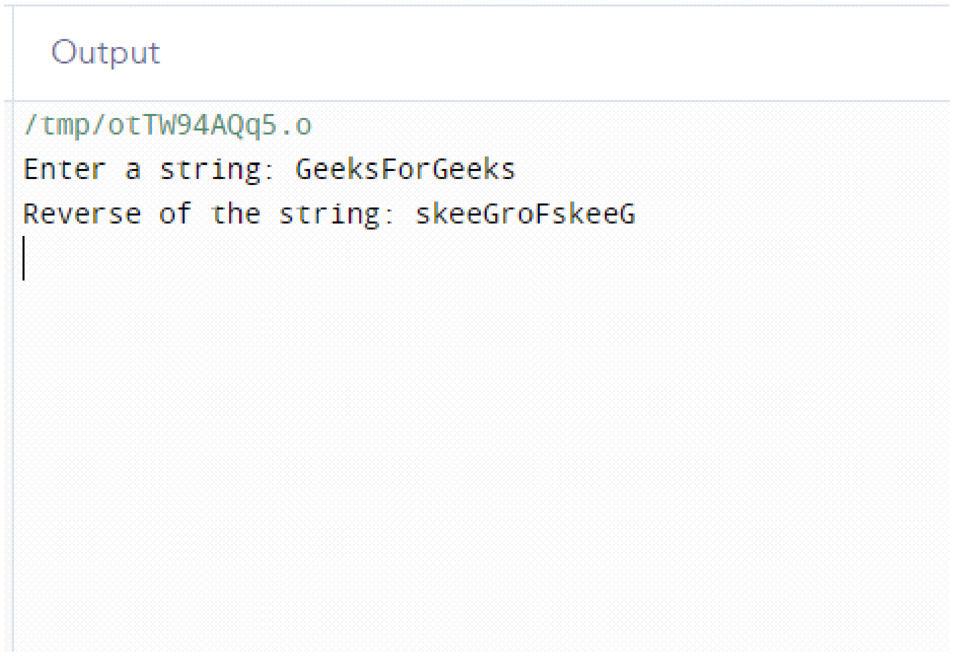
Code

```
#include <bits/stdc++.h>
```

```
using namespace std;

// Function to reverse the string
// using pointers
void reverseString(char* str)
{
int l, i;
char *begin_ptr, *end_ptr, ch;

// Get the length of the string
l = strlen(str);

// Set the begin_ptr
// initially to start of string
begin_ptr = str;

//Setting end_ptr initially to
//the end of the string
end_ptr = str + l - 1;

// Swap the char from start and end
// index using begin_ptr and end_ptr
for (i = 0; i < (l - 1) / 2; i++) {

        // swap character
        ch = *end_ptr;
        *end_ptr = *begin_ptr;
        *begin_ptr = ch;

        // update pointers positions
```

```
            begin_ptr++;
            end_ptr--;
    }
    }

    // Driver code
    int main()
    {

    // Get the string
    char str[100] = "GeeksForGeeks";
    cout<<"Enter a string: "<<str<<endl;

    // Reverse the string
    reverseString(str);

    // Print the result
    printf("Reverse of the string: %s\n", str);

    return 0;
    }
```

Output



```
/tmp/otTW94AQq5.o
Enter a string: GeeksForGeeks
Reverse of the string: skeeGroFskeeG
```

Program

Program to copy one array to another using pointers

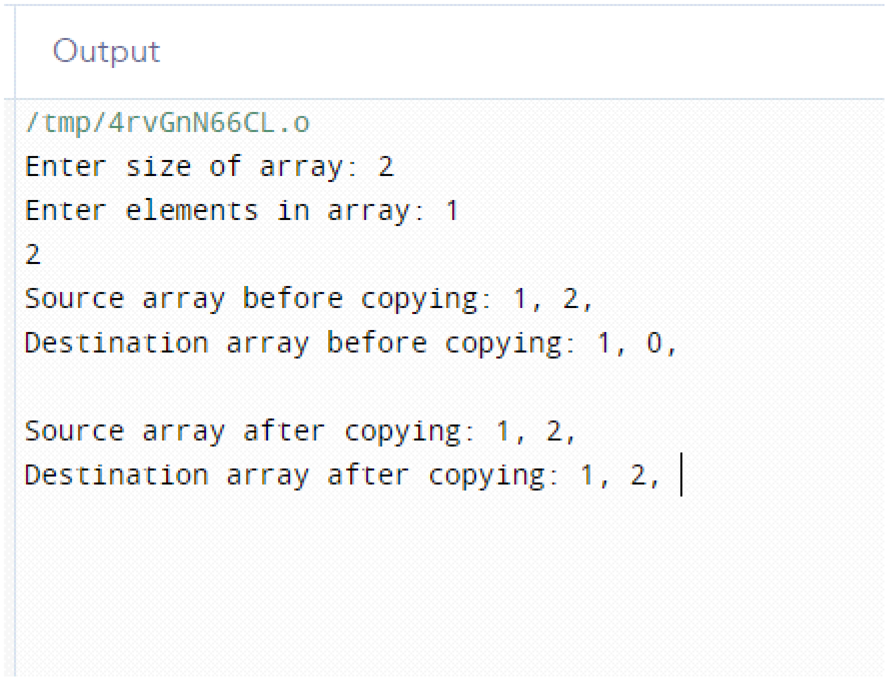
Code

```
/**
 * C program to copy an array to another array using pointers
 */

#include <stdio.h>

#define MAX_SIZE 100 // Maximum array size


/* Function declaration to print array */
void printArray(int arr[], int size);


int main()
```

```
{
    int source_arr[MAX_SIZE], dest_arr[MAX_SIZE];
    int size, i;

    int *source_ptr = source_arr;   // Pointer to source_arr
    int *dest_ptr   = dest_arr;     // Pointer to dest_arr

    int *end_ptr;


    /*
     * Input size and elements in source array
     */
    printf("Enter size of array: ");
    scanf("%d", &size);
    printf("Enter elements in array: ");
    for (i = 0; i < size; i++)
    {
        scanf("%d", (source_ptr + i));
    }


    // Pointer to last element of source_arr
    end_ptr = &source_arr[size - 1];


    /* Print source and destination array before copying */
    printf("\nSource array before copying: ");
    printArray(source_arr, size);
    printf("\nDestination array before copying: ");
    printArray(dest_arr, size);



    /*
     * Run loop till source_ptr exists in source_arr
     * memory range.
     */
    while(source_ptr <= end_ptr)
    {
        *dest_ptr = *source_ptr;

        // Increment source_ptr and dest_ptr
        source_ptr++;
        dest_ptr++;
    }


    /* Print source and destination array after copying */
    printf("\n\nSource array after copying: ");
    printArray(source_arr, size);

    printf("\nDestination array after copying: ");
    printArray(dest_arr, size);



    return 0;
}
```

```
/**
 * Function to print array elements.
 *
 * @arr     Integer array to print.
 * @size    Size of array.
 */
void printArray(int *arr, int size)
{
    int i;

    for (i = 0; i < size; i++)
    {
        printf("%d, ", *(arr + i));
    }
}
```

Output



Output

```
/tmp/4rvGnN66CL.o
Enter size of array: 2
Enter elements in array: 1
2
Source array before copying: 1, 2,
Destination array before copying: 1, 0,

Source array after copying: 1, 2,
Destination array after copying: 1, 2,
```

Probem

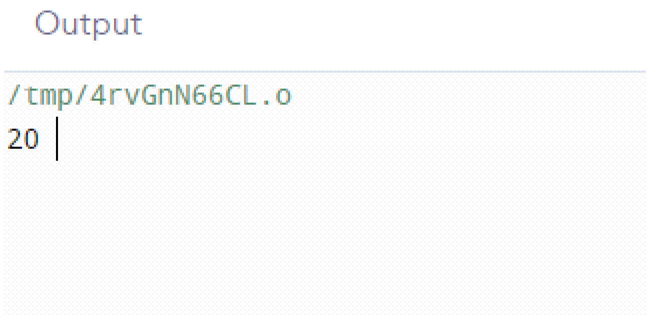
Program to Find the Largest Element in an Array using the Pointer

Code

```
// C program for the above approach
#include <stdio.h>
#include <stdlib.h>

// Function to find the largest element
// using dynamic memory allocation
void findLargest(int* arr, int N)
{
	int i;

	// Traverse the array arr[]
	for (i = 1; i < N; i++) {
		// Update the largest element
		if (*arr < *(arr + i)) {
			*arr = *(arr + i);
		}
	}

	// Print the largest number
	printf("%d ", *arr);
}

// Driver Code
int main()
{
	int i, N = 4;
```

```
    int* arr;

    // Memory allocation to arr
    arr = (int*)calloc(N, sizeof(int));

    // Condition for no memory
    // allocation
    if (arr == NULL) {
        printf("No memory allocated");
        exit(0);
    }

    // Store the elements
    *(arr + 0) = 14;
    *(arr + 1) = 12;
    *(arr + 2) = 19;
    *(arr + 3) = 20;

    // Function Call
    findLargest(arr, N);
    return 0;
}
```

Output


Output

```
/tmp/4rvGnN66CL.o
20
```