

Assignment 01

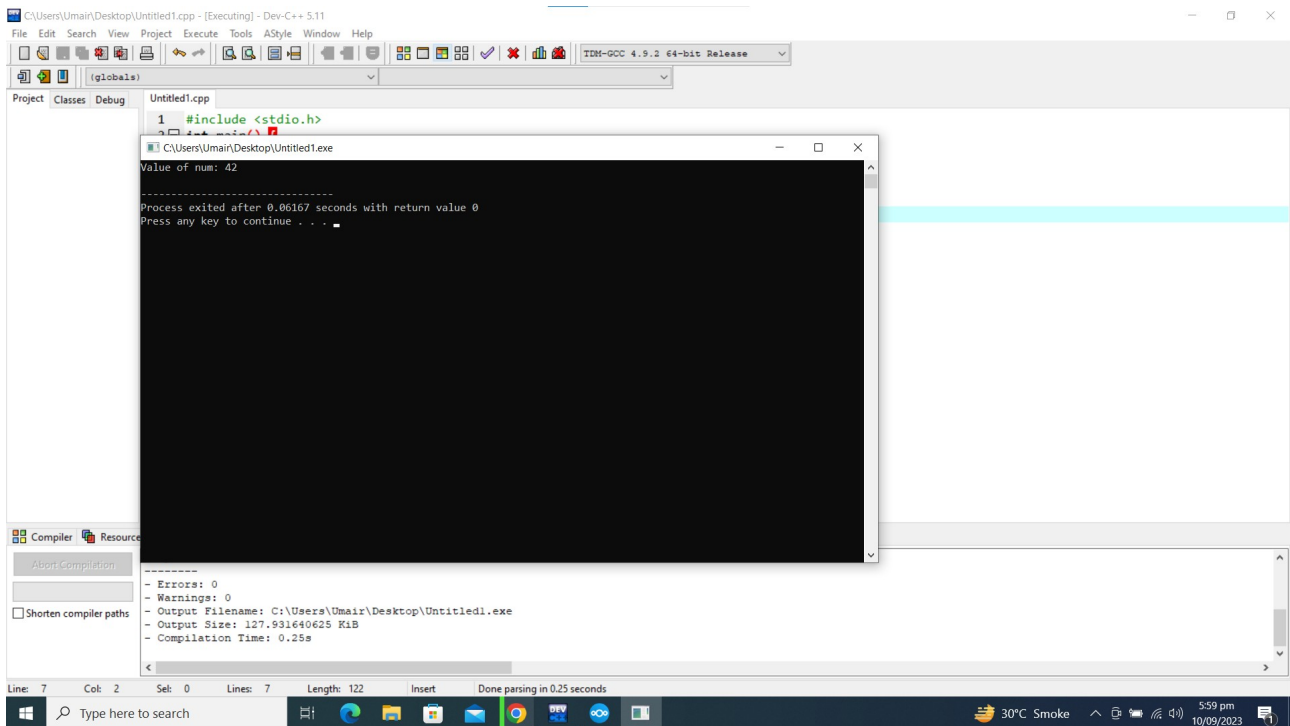
Submitted by:	Umair Abbas
Reg. No:	SP22-BCS-079
Section:	B
Subject:	Data Structure
Title:	Pointers
Submitted to:	Maam Yasmeen

Comsats University Islamabad, Vehari Campus

Program no 01:

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

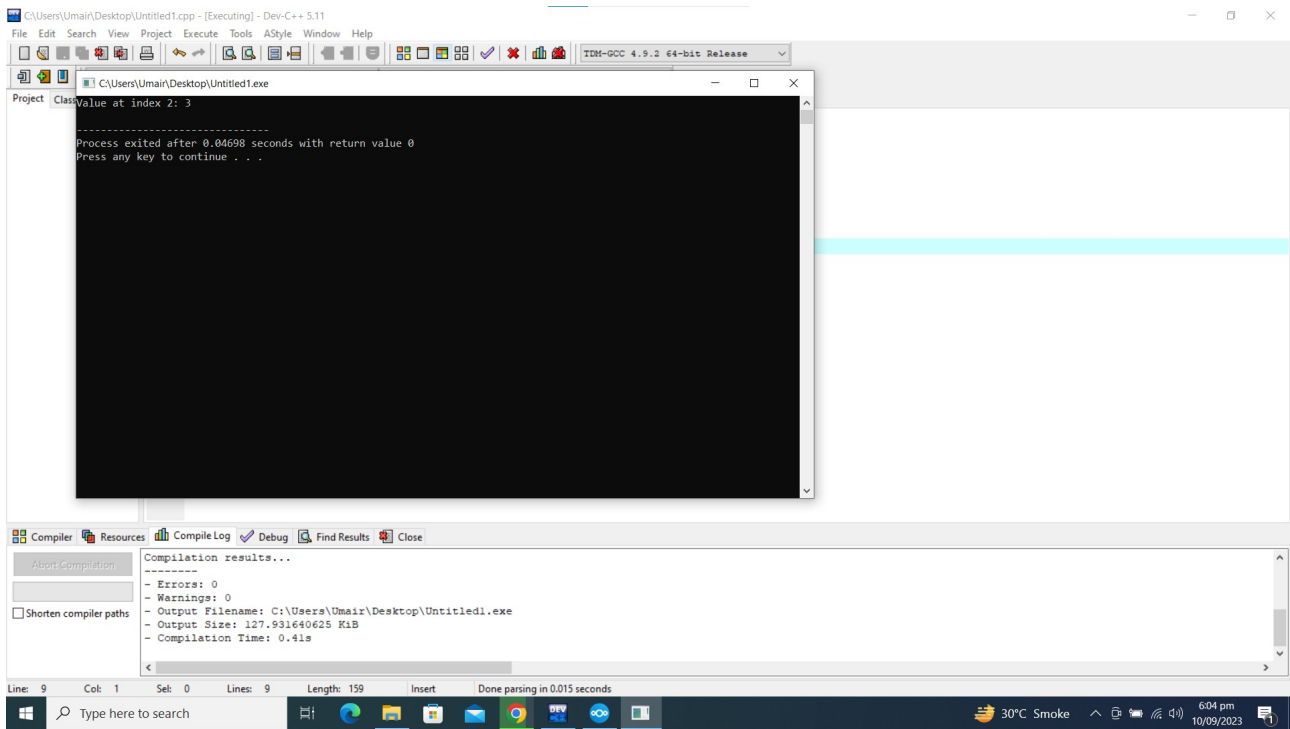
```
int main() {  
    int num = 42;  
    int *ptr = &num;  
    printf("Value of num: %d\n", *ptr);  
    return 0;  
}
```



Program no 02:

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

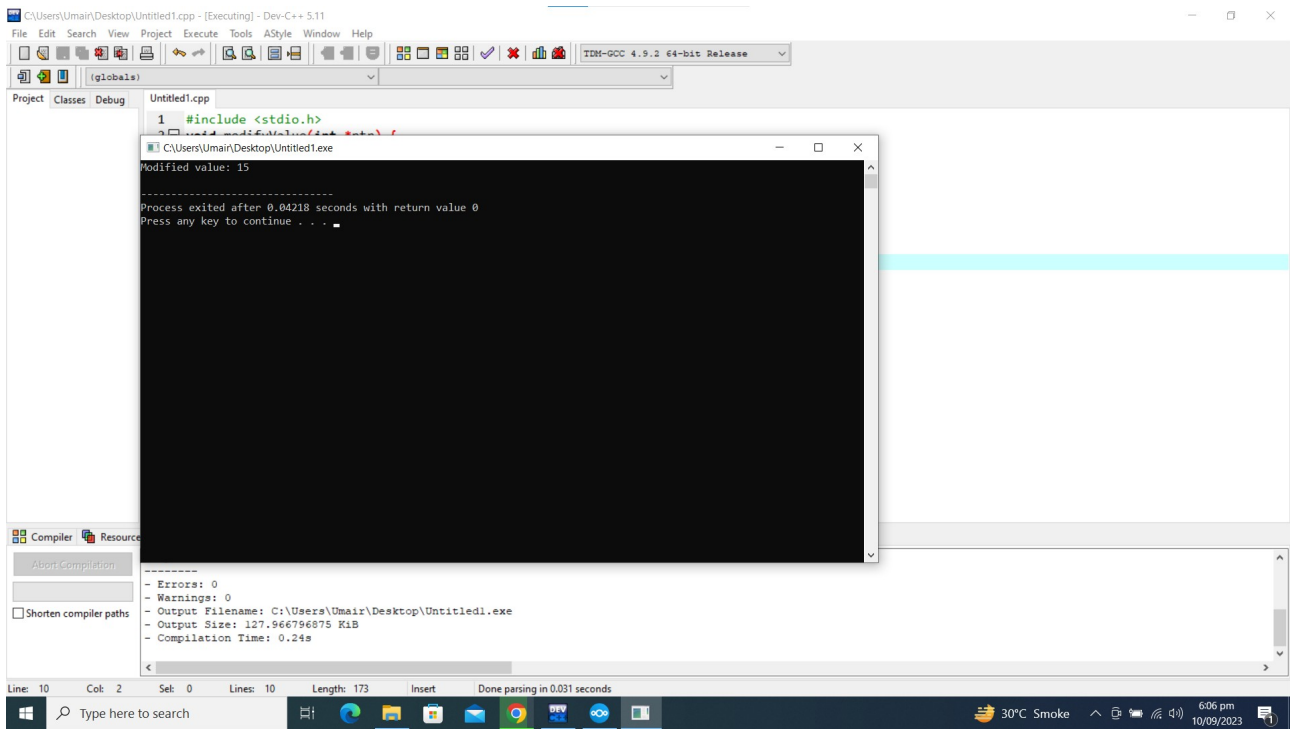
```
int main() {  
    int arr[] = {1, 2, 3, 4, 5};  
    int *ptr = arr;  
    ptr = ptr + 2;  
    printf("Value at index 2: %d\n", *ptr);  
    return 0;  
}
```



Program no 03:

#include <stdio.h>

```
int main() {  
    int num = 42;  
    int *ptr1 = &num;  
    int **ptr2 = &ptr1;  
    printf("Value of num via ptr2: %d\n", **ptr2);  
    return 0;  
}
```

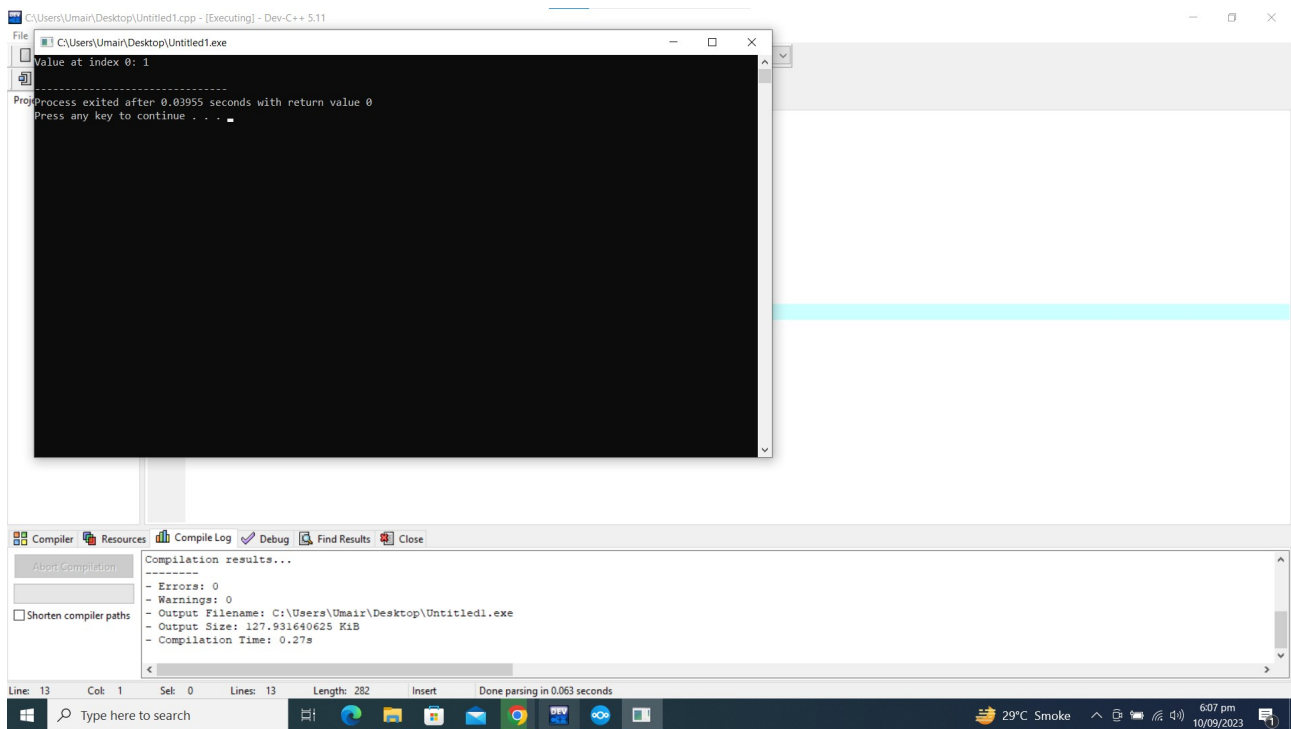


Program no 04:

#include <stdio.h>

```
void modifyValue(int *ptr) {
    (*ptr) += 10;
}
```

```
int main() {
    int num = 5;
    modifyValue(&num);
    printf("Modified value: %d\n", num);
    return 0;
}
```

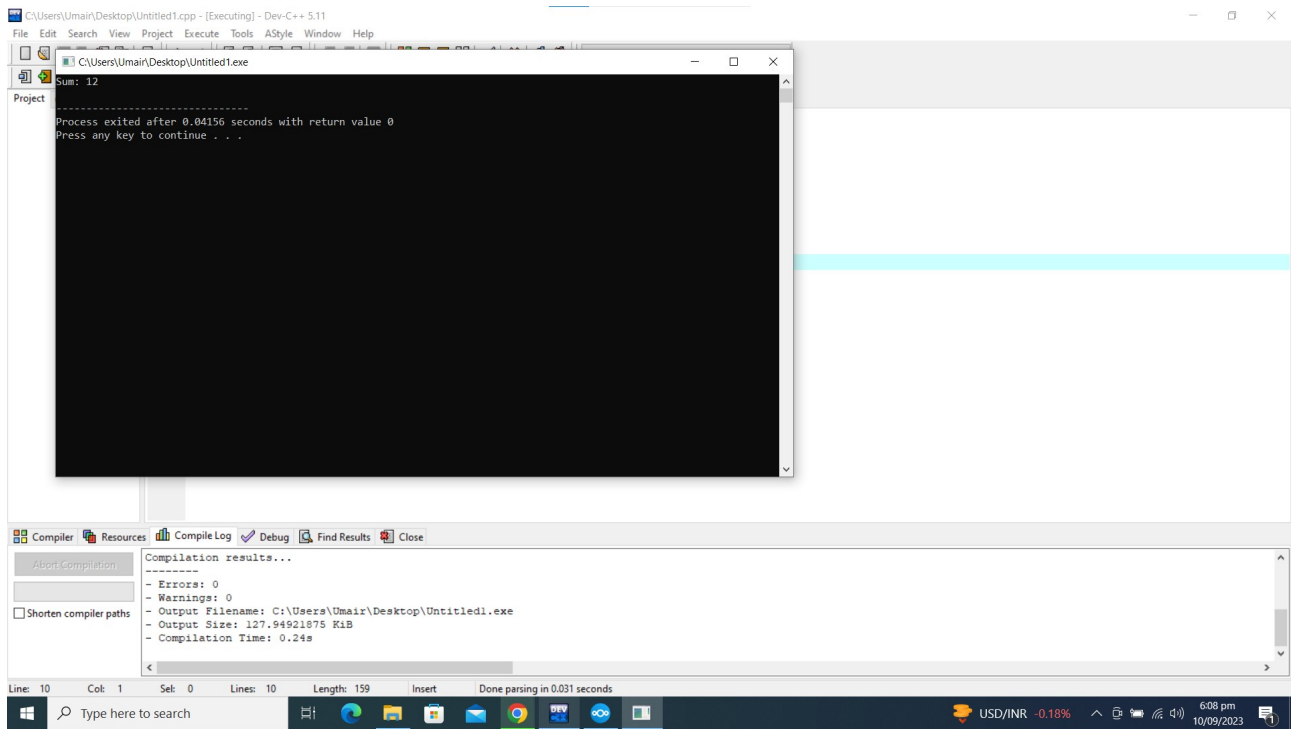


Program no 05:

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

```
#include <stdlib.h>
```

```
int main() {  
    int *arr = (int *)malloc(5 * sizeof(int));  
    if (arr != NULL) {  
        // Initialize and use arr  
        arr[0] = 1;  
        printf("Value at index 0: %d\n", arr[0]);  
        free(arr); // Don't forget to free allocated memory  
    }  
    return 0;  
}
```

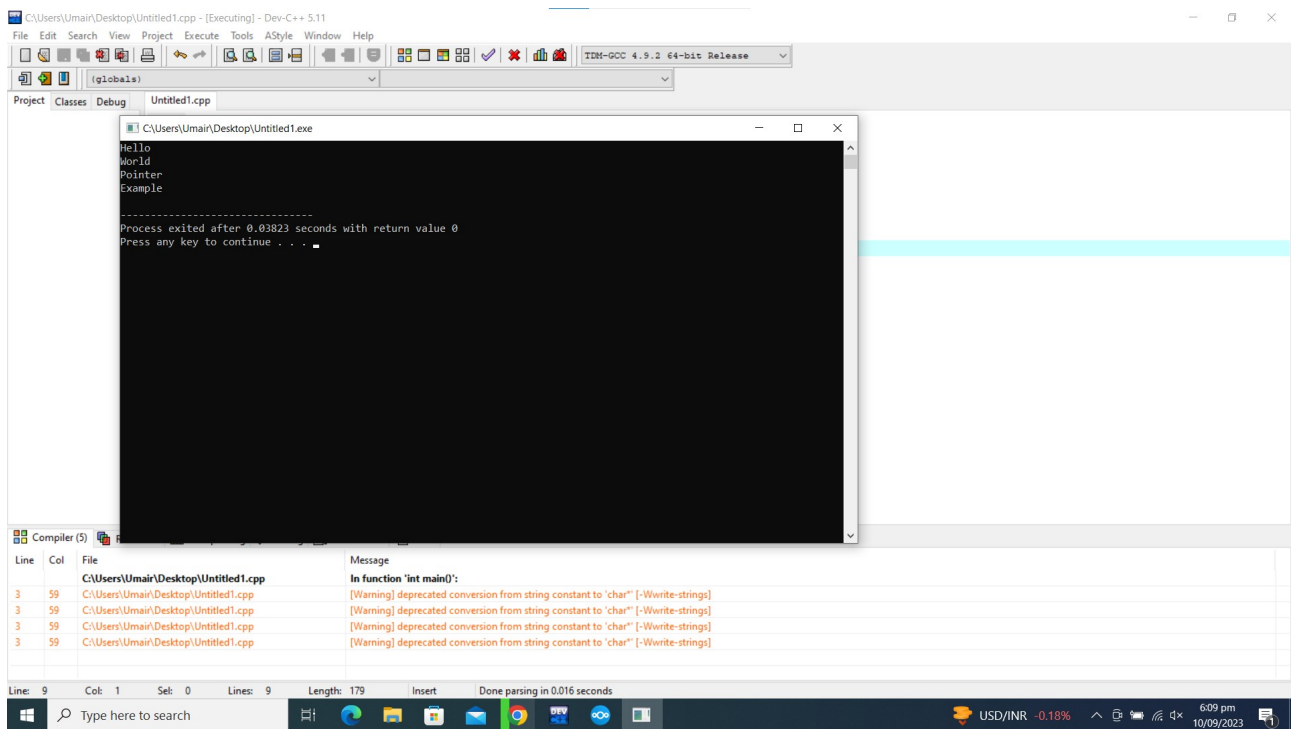


Program no 06:

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

```
int add(int a, int b) {  
    return a + b;  
}
```

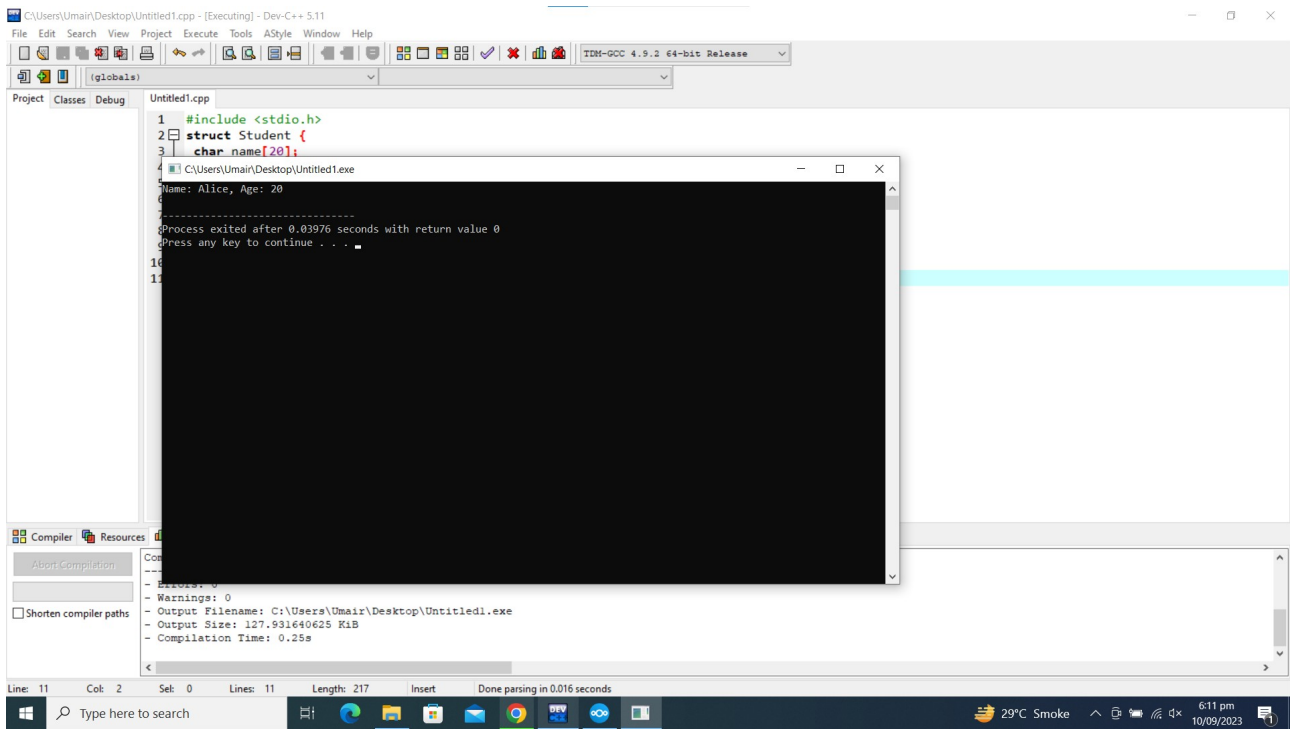
```
int main() {  
    int (*ptr)(int, int) = add;  
    printf("Sum: %d\n", ptr(5, 7));  
    return 0;  
}
```



Program no 07:

#include <stdio.h>

```
int main() {  
    char *strings[] = {"Hello", "World", "Pointer", "Example"};  
    for (int i = 0; i < 4; i++) {  
        printf("%s\n", strings[i]);  
    }  
    return 0;  
}
```

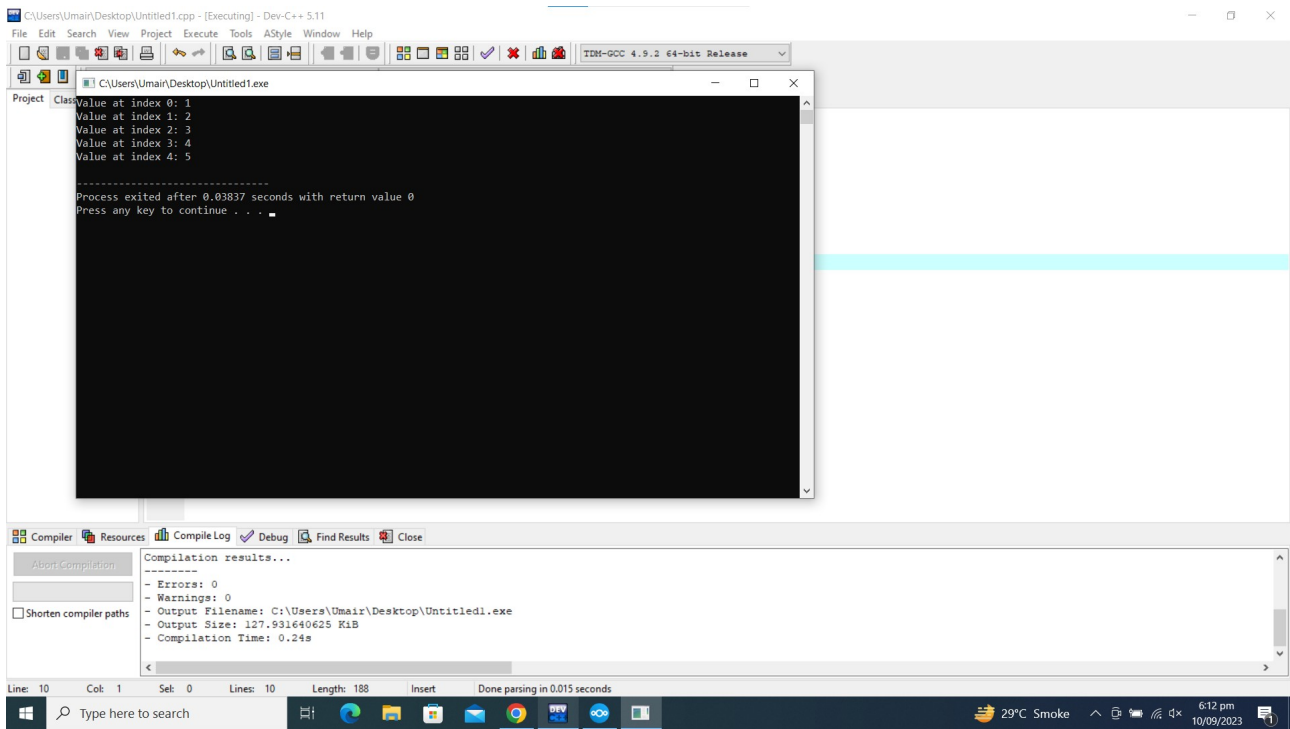


Program no 08:

#include <stdio.h>

```
struct Student {  
    char name[20];  
    int age;  
};
```

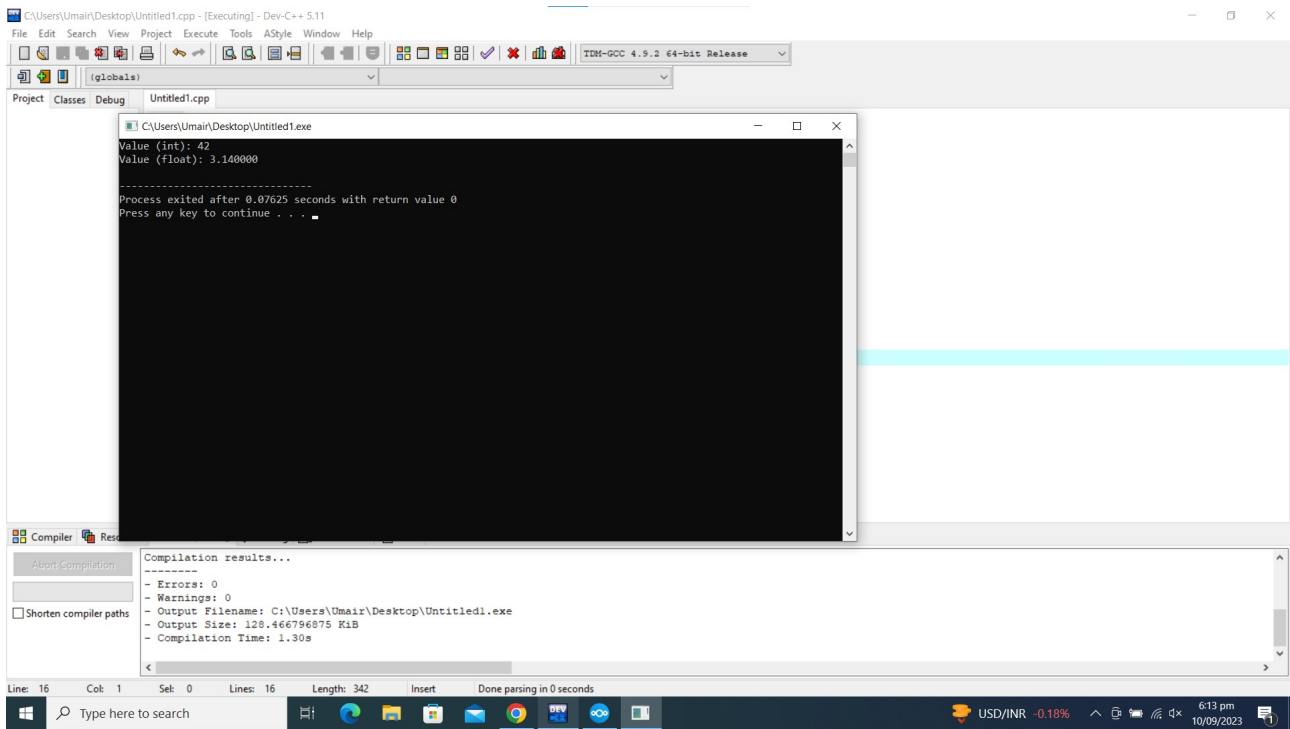
```
int main() {  
    struct Student s = {"Alice", 20};  
    struct Student *ptr = &s;  
    printf("Name: %s, Age: %d\n", ptr->name, ptr->age);  
    return 0;  
}
```

Program no 09:

#include <stdio.h>

```
int main() {  
    int arr[] = {1, 2, 3, 4, 5};  
    int *ptr = arr;  
    for (int i = 0; i < 5; i++) {  
        printf("Value at index %d: %d\n", i, *(ptr + i));  
    }  
    return 0;  
}
```

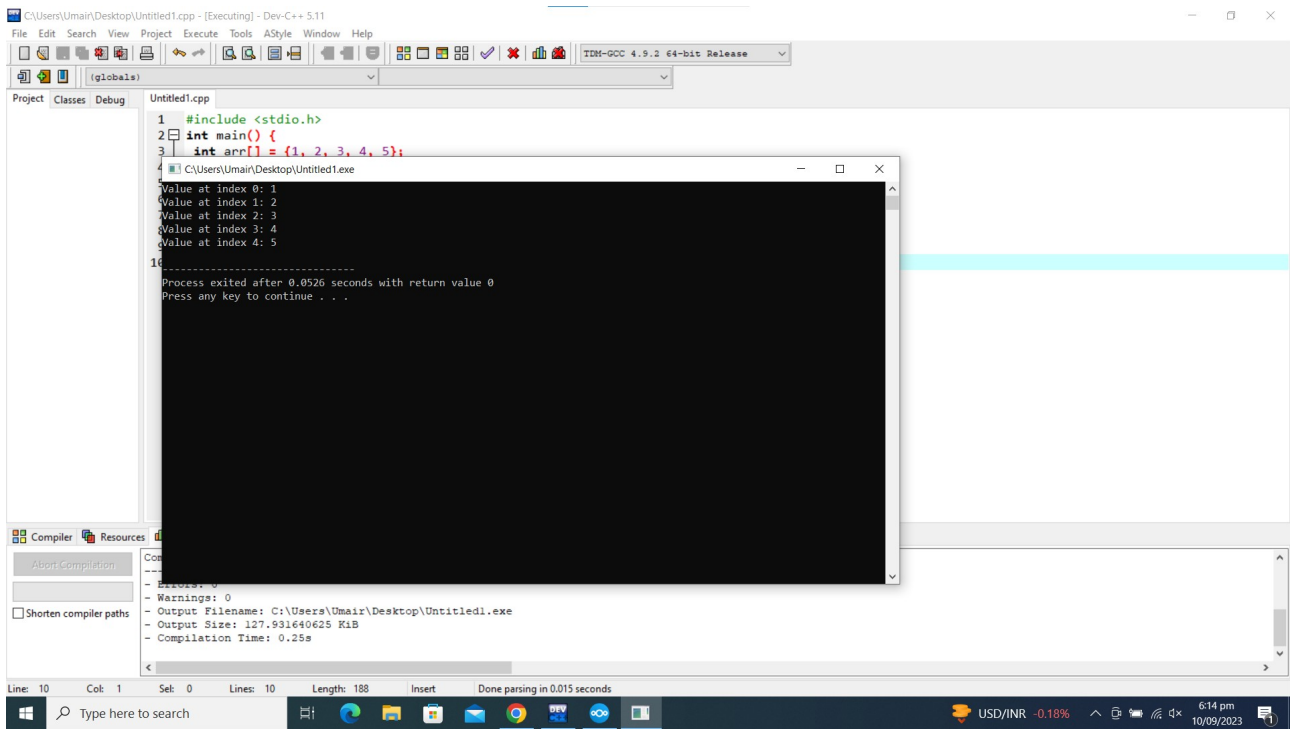


Program no 10:

#include <stdio.h>

```
void printValue(void *ptr, char type) {  
    if (type == 'i') {  
        printf("Value (int): %d\n", *((int *)ptr));  
    } else if (type == 'f') {  
        printf("Value (float): %f\n", *((float *)ptr));  
    }  
}
```

```
int main() {  
    int iValue = 42;  
    float fValue = 3.14;  
  
    printValue(&iValue, 'i');  
    printValue(&fValue, 'f');  
  
    return 0;  
}
```



Program no 11:

#include <stdio.h>

int main() {

int num1 = 42;

int num2 = 50;

const int *ptr1 = &num1; // Pointer to constant

int const *ptr2 = &num1; // Also pointer to constant

int *const ptr3 = &num2; // Constant pointer

// Modify num1 through ptr3 (error) and num2 through ptr1 (ok)

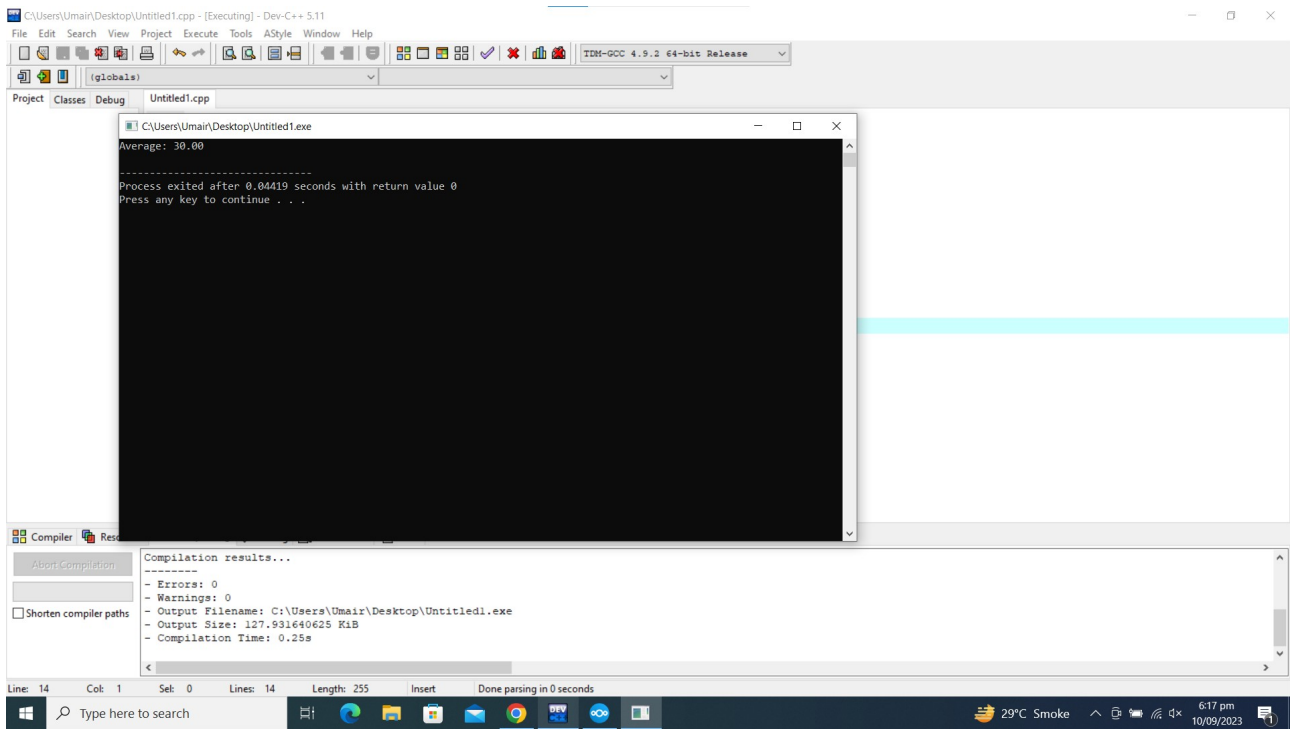
*ptr3 = 60; // Error

*ptr1 = 55; // OK

printf("num1: %d, num2: %d\n", num1, num2);

return 0;

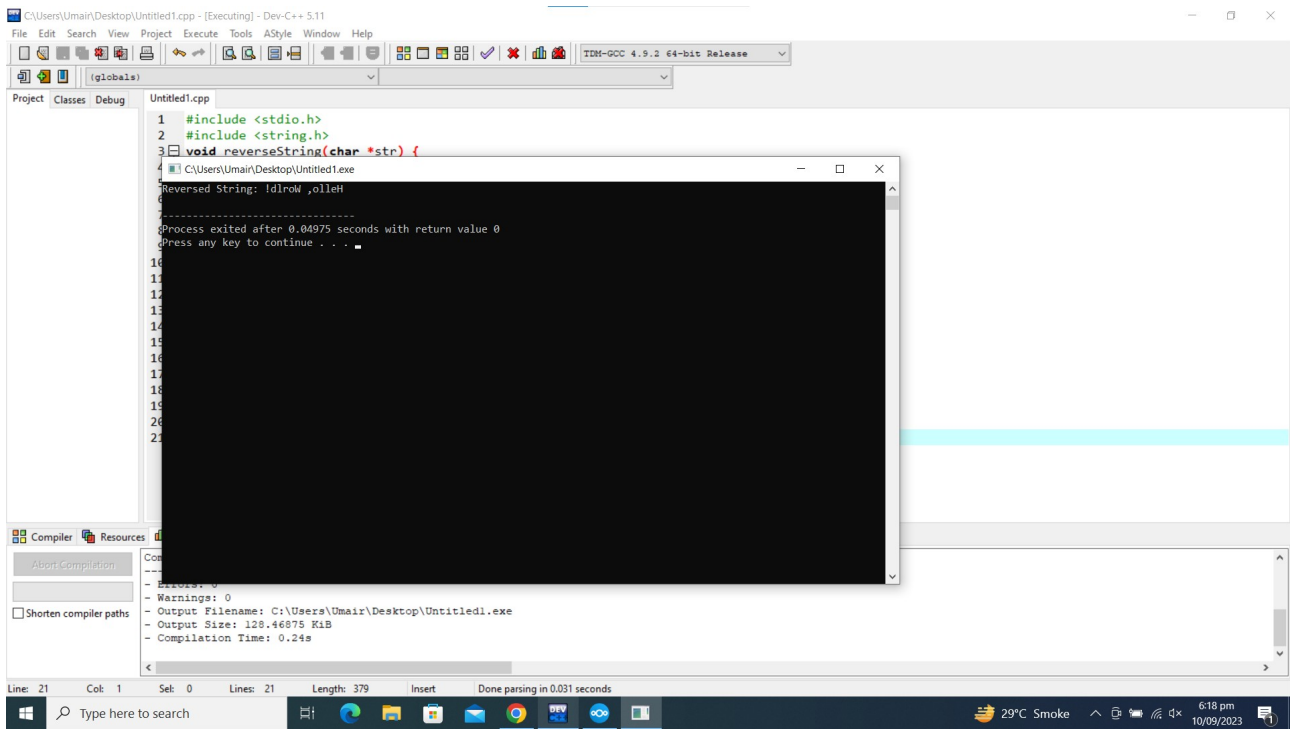
}



Program no 12:

#include <stdio.h>

```
int main() {  
    int arr[] = {10, 20, 30, 40, 50};  
    int *ptr = arr;  
    int sum = 0;  
  
    for (int i = 0; i < 5; i++) {  
        sum += *ptr;  
        ptr++;  
    }  
  
    double average = (double)sum / 5;  
    printf("Average: %.2lf\n", average);  
  
    return 0;  
}
```



Program no 13:

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

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

```
void reverseString(char *str) {
```

```
    int len = strlen(str);
```

```
    char *start = str;
```

```
    char *end = str + len - 1;
```

```
    while (start < end) {
```

```
        char temp = *start;
```

```
        *start = *end;
```

```
        *end = temp;
```

```
        start++;
```

```
        end--;
```

```
    }
```

```
}
```

```
int main() {
```

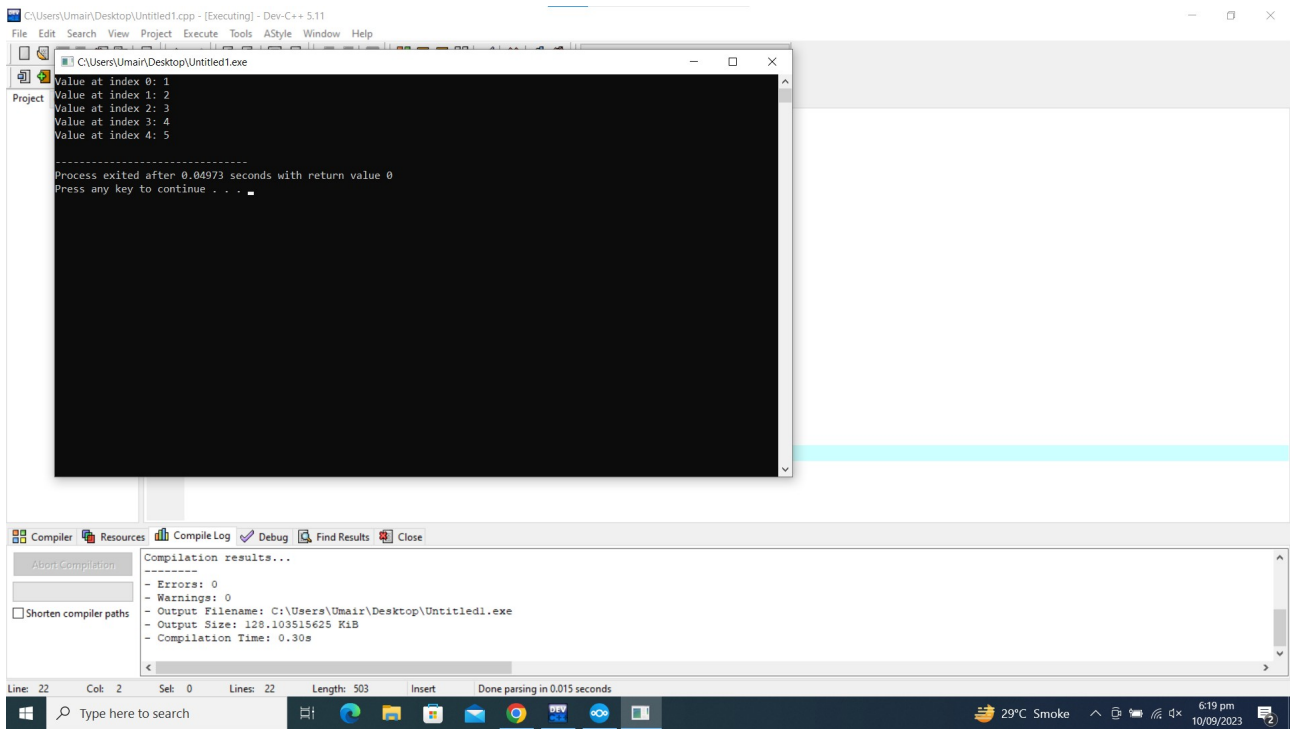
```
    char str[] = "Hello, World!";
```

```
    reverseString(str);
```

```
    printf("Reversed String: %s\\n", str);
```

```
    return 0;
```

```
}
```

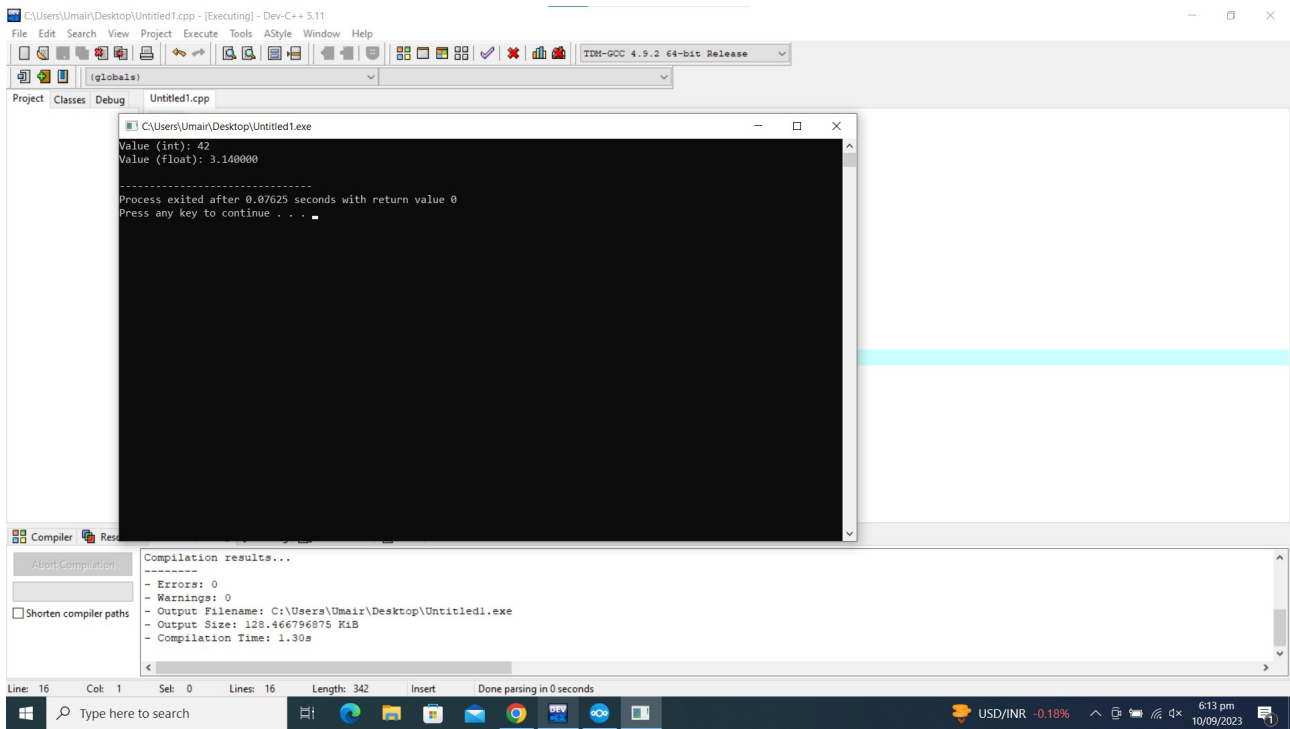


Program no 14:

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

```
#include <stdlib.h>
```

```
int main() {  
    int *arr = (int *)malloc(3 * sizeof(int)); // Initial allocation  
    if (arr != NULL) {  
        arr[0] = 1;  
        arr[1] = 2;  
        arr[2] = 3;  
  
        // Resize the array to hold 5 elements  
        int *temp = (int *)realloc(arr, 5 * sizeof(int));  
        if (temp != NULL) {  
            arr = temp;  
            arr[3] = 4;  
            arr[4] = 5;  
  
            for (int i = 0; i < 5; i++) {  
                printf("Value at index %d: %d\n", i, arr[i]);  
            }  
        }  
        free(arr); // Don't forget to free allocated memory  
    }  
    return 0;  
}
```



Program no 15:

#include <stdio.h>

```
void findMinMax(const int *arr, int size, int *min, int *max) {  
    if (size <= 0) {  
        // Handle the case where the array is empty or invalid.  
        return;  
    }  
}
```

```
*min = *max = arr[0]; // Initialize min and max with the first element
```

```
for (int i = 1; i < size; i++) {  
    if (arr[i] < *min) {  
        // Update min if a smaller value is found  
        *min = arr[i];  
    } else if (arr[i] > *max) {  
        // Update max if a larger value is found  
        *max = arr[i];  
    }  
}  
}
```

```
int main() {  
    int arr[] = {12, 5, 67, 2, 42, 8, 31};  
    int size = sizeof(arr) / sizeof(arr[0]);  
    int min, max;
```

```
findMinMax(arr, size, &min, &max);
```

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

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

```
return 0;
```

```
}
```

