

Day 3

Class work

1. Write a program to delete an element from an array using pointers.

```
1 #include <stdio.h>
2 void deleteElement(int *arr, int *size, int index) {
3     if (index < 0 || index >= *size) {
4         printf("Invalid index\n");
5         return;
6     }
7     for (int i = index; i < *size - 1; i++) {
8         arr[i] = arr[i + 1];
9     }
10    (*size)--;
11 }
12 int main() {
13     int arr[] = {1, 2, 3, 4, 5};
14     int size = sizeof(arr) / sizeof(arr[0]);
15     int index;
16     printf("Enter the index of the element to delete: ");
17     scanf("%d", &index);
18     deleteElement(arr, &size, index);
19     printf("Array after deletion:\n");
20     for (int i = 0; i < size; i++) {
21         printf("%d ", arr[i]);
22     }
23     printf("\n");
24     return 0;
25 }
```

Enter the index of the element to delete: 4
Array after deletion:
1 2 3 4

Process exited after 12.75 seconds with return value 0
Press any key to continue . . .

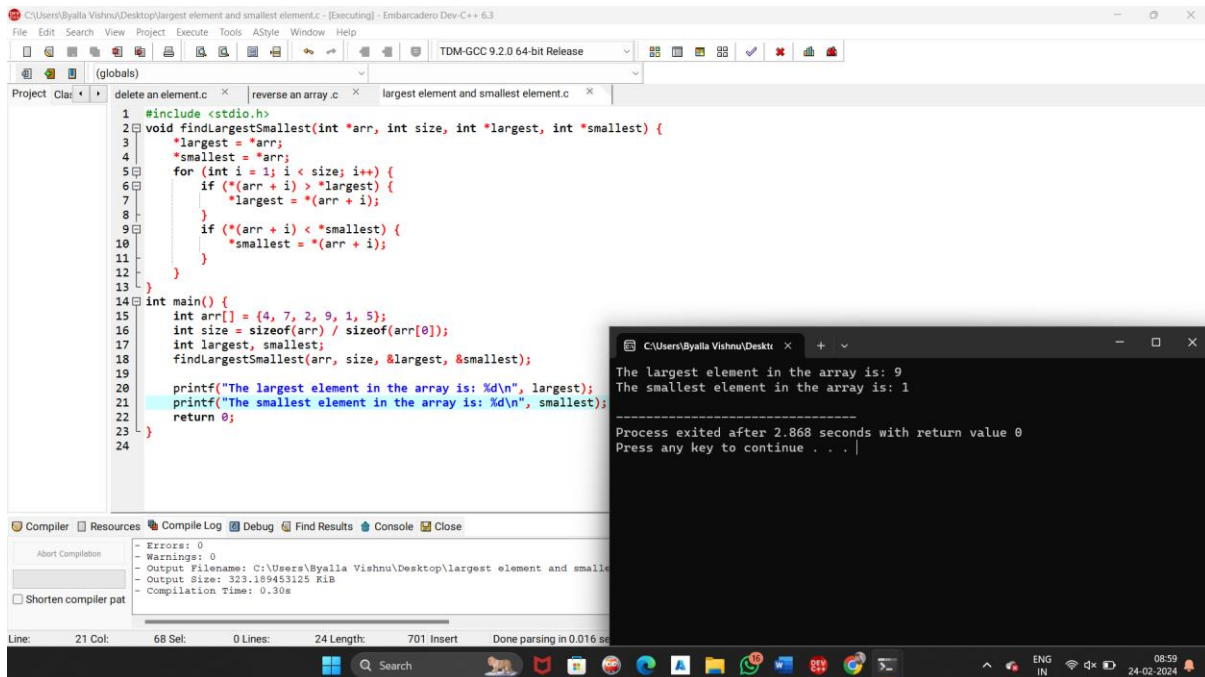
2. Write a program to reverse an array using pointers.

```
1 #include <stdio.h>
2 void reverseArray(int *arr, int size) {
3     int *start = arr;
4     int *end = arr + size - 1;
5     while (start < end) {
6         int temp = *start;
7         *start = *end;
8         *end = temp;
9         start++;
10        end--;
11    }
12 }
13 int main() {
14     int arr[] = {1, 2, 3, 4, 5};
15     int size = sizeof(arr) / sizeof(arr[0]);
16     printf("Original array:\n");
17     for (int i = 0; i < size; i++) {
18         printf("%d ", arr[i]);
19     }
20     printf("\n");
21     reverseArray(arr, size);
22     printf("Reversed array:\n");
23     for (int i = 0; i < size; i++) {
24         printf("%d ", arr[i]);
25     }
26     printf("\n");
27 }
```

Original array:
1 2 3 4 5
Reversed array:
5 4 3 2 1

Process exited after 3.252 seconds with return value 0
Press any key to continue . . .

3. Write a program to find the largest and smallest elements in an array using pointers.

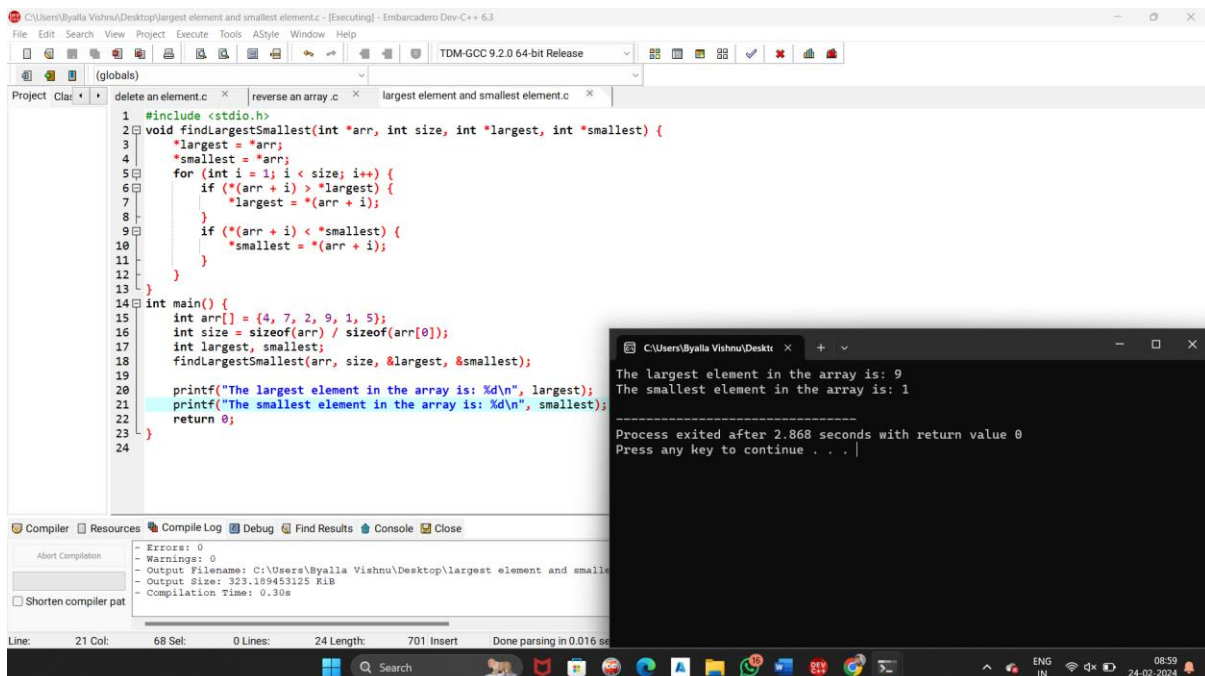


```
1 #include <stdio.h>
2 void findLargestSmallest(int *arr, int size, int *largest, int *smallest) {
3     *largest = *arr;
4     *smallest = *arr;
5     for (int i = 1; i < size; i++) {
6         if (*(arr + i) > *largest) {
7             *largest = *(arr + i);
8         }
9         if (*(arr + i) < *smallest) {
10            *smallest = *(arr + i);
11        }
12    }
13 }
14 int main() {
15     int arr[] = {4, 7, 2, 9, 1, 5};
16     int size = sizeof(arr) / sizeof(arr[0]);
17     int largest, smallest;
18     findLargestSmallest(arr, size, &largest, &smallest);
19
20     printf("The largest element in the array is: %d\n", largest);
21     printf("The smallest element in the array is: %d\n", smallest);
22     return 0;
23 }
24
```

The largest element in the array is: 9
The smallest element in the array is: 1

Process exited after 2.868 seconds with return value 0
Press any key to continue . . .

4. Write a program to copy the contents of one array to another using pointers.

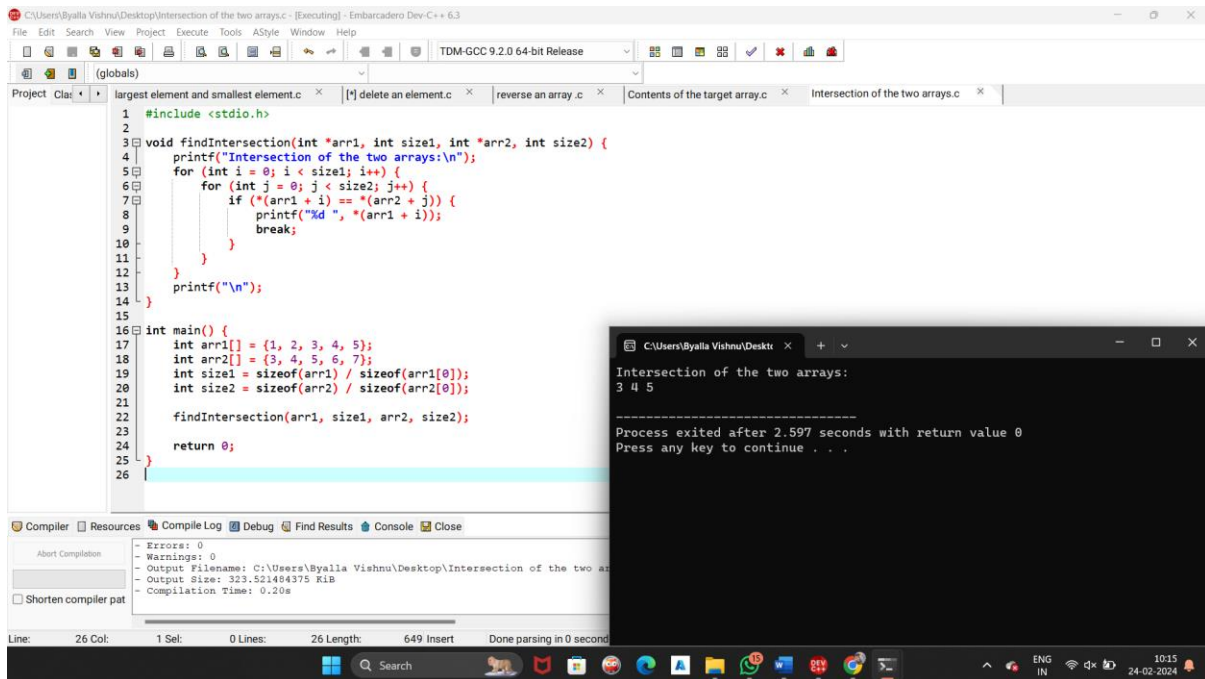


```
1 #include <stdio.h>
2 void findLargestSmallest(int *arr, int size, int *largest, int *smallest) {
3     *largest = *arr;
4     *smallest = *arr;
5     for (int i = 1; i < size; i++) {
6         if (*(arr + i) > *largest) {
7             *largest = *(arr + i);
8         }
9         if (*(arr + i) < *smallest) {
10            *smallest = *(arr + i);
11        }
12    }
13 }
14 int main() {
15     int arr[] = {4, 7, 2, 9, 1, 5};
16     int size = sizeof(arr) / sizeof(arr[0]);
17     int largest, smallest;
18     findLargestSmallest(arr, size, &largest, &smallest);
19
20     printf("The largest element in the array is: %d\n", largest);
21     printf("The smallest element in the array is: %d\n", smallest);
22     return 0;
23 }
24
```

The largest element in the array is: 9
The smallest element in the array is: 1

Process exited after 2.868 seconds with return value 0
Press any key to continue . . .

5. Write a program to find the intersection of two arrays using pointers

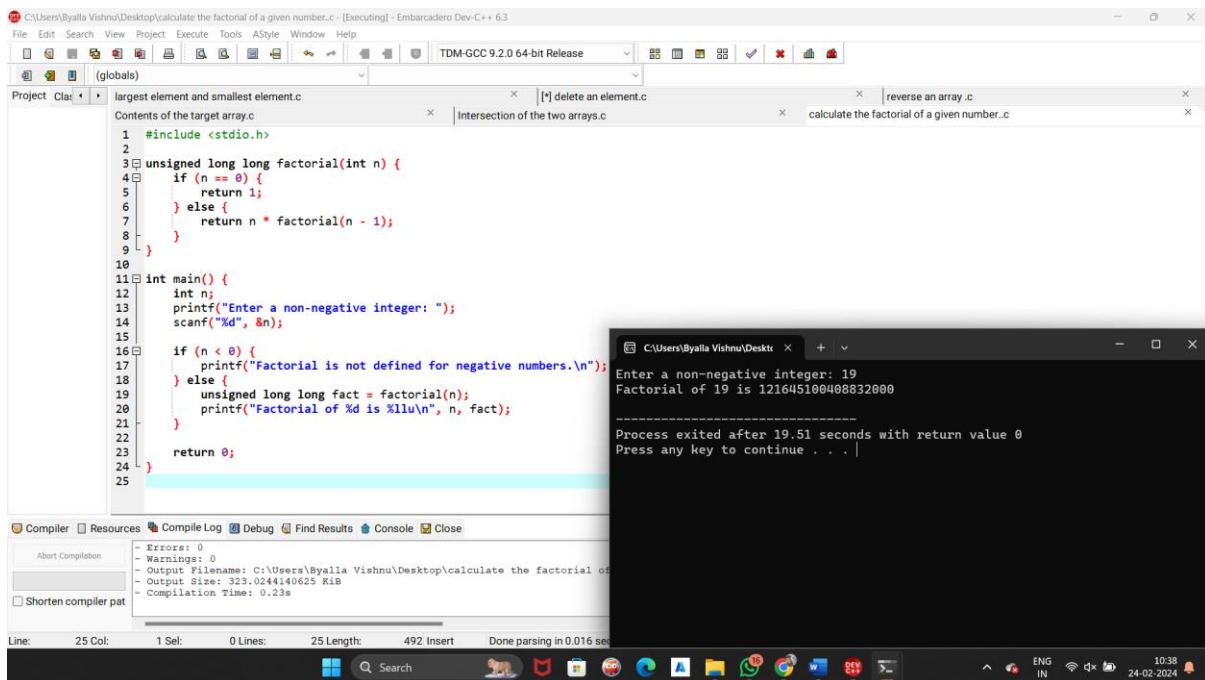


```
1 #include <stdio.h>
2
3 void findIntersection(int *arr1, int size1, int *arr2, int size2) {
4     printf("Intersection of the two arrays:\n");
5     for (int i = 0; i < size1; i++) {
6         for (int j = 0; j < size2; j++) {
7             if (*(arr1 + i) == *(arr2 + j)) {
8                 printf("%d ", *(arr1 + i));
9                 break;
10            }
11        }
12    }
13    printf("\n");
14 }
15
16 int main() {
17     int arr1[] = {1, 2, 3, 4, 5};
18     int arr2[] = {3, 4, 5, 6, 7};
19     int size1 = sizeof(arr1) / sizeof(arr1[0]);
20     int size2 = sizeof(arr2) / sizeof(arr2[0]);
21
22     findIntersection(arr1, size1, arr2, size2);
23
24     return 0;
25 }
26
```

Intersection of the two arrays:
3 4 5

Process exited after 2.597 seconds with return value 0
Press any key to continue . . .

6. Write a recursive program to calculate the factorial of a given number.

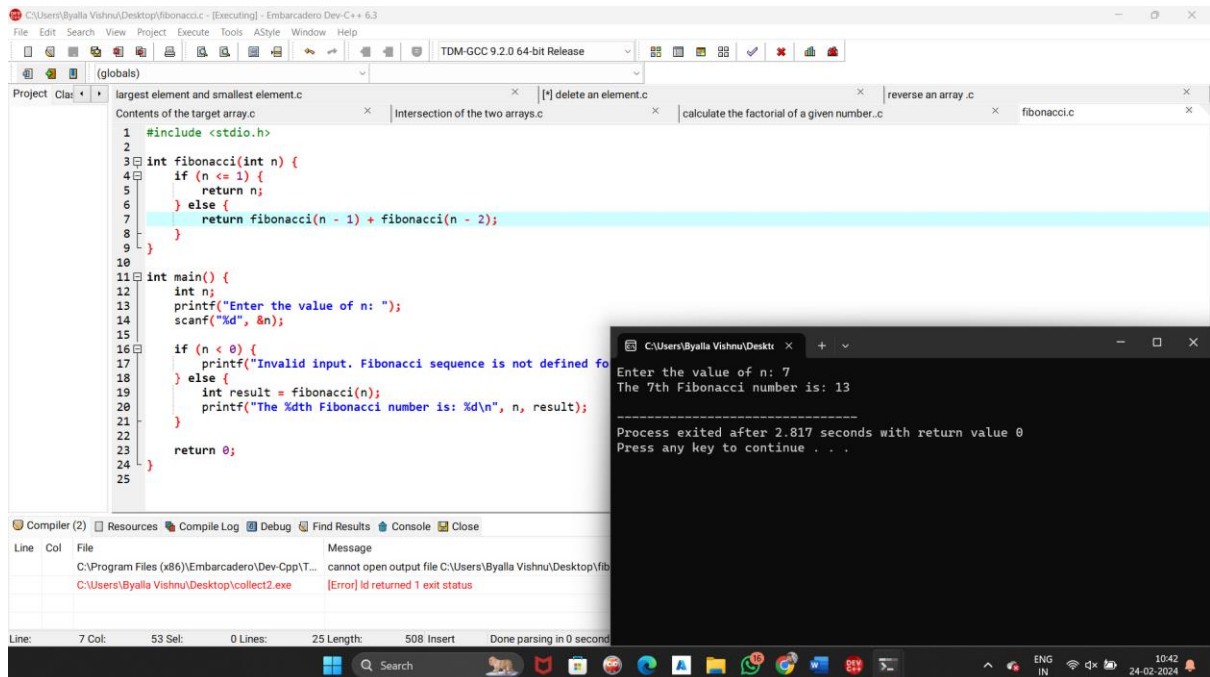


```
1 #include <stdio.h>
2
3 unsigned long long factorial(int n) {
4     if (n == 0) {
5         return 1;
6     } else {
7         return n * factorial(n - 1);
8     }
9 }
10
11 int main() {
12     int n;
13     printf("Enter a non-negative integer: ");
14     scanf("%d", &n);
15
16     if (n < 0) {
17         printf("Factorial is not defined for negative numbers.\n");
18     } else {
19         unsigned long long fact = factorial(n);
20         printf("Factorial of %d is %llu\n", n, fact);
21     }
22
23     return 0;
24 }
25
```

Enter a non-negative integer: 19
Factorial of 19 is 121645100408832000

Process exited after 19.51 seconds with return value 0
Press any key to continue . . .

7. Implement a recursive program to find the nth Fibonacci number



The screenshot shows an IDE with a C++ project. The main file, `fibonacci.c`, contains the following code:

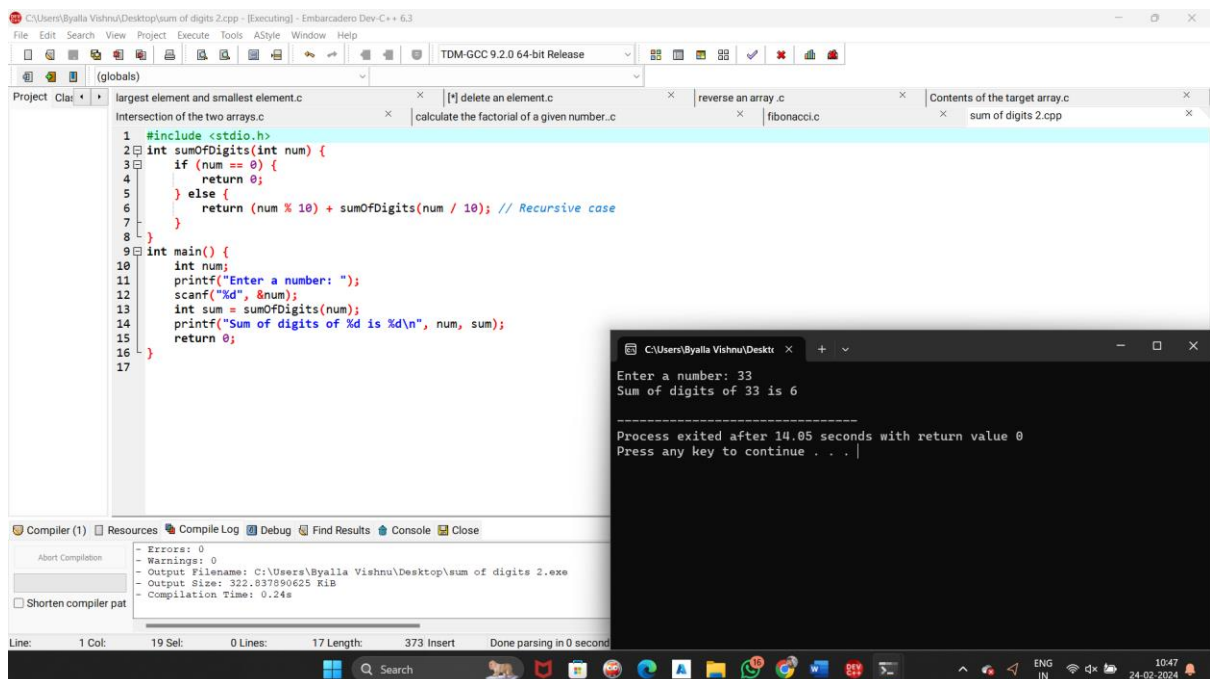
```
1 #include <stdio.h>
2
3 int fibonacci(int n) {
4     if (n <= 1) {
5         return n;
6     } else {
7         return fibonacci(n - 1) + fibonacci(n - 2);
8     }
9 }
10
11 int main() {
12     int n;
13     printf("Enter the value of n: ");
14     scanf("%d", &n);
15
16     if (n < 0) {
17         printf("Invalid input. Fibonacci sequence is not defined for negative numbers.");
18     } else {
19         int result = fibonacci(n);
20         printf("The %dth Fibonacci number is: %d\n", n, result);
21     }
22
23     return 0;
24 }
```

The console output shows the program running successfully for `n = 7`:

```
Enter the value of n: 7
The 7th Fibonacci number is: 13

Process exited after 2.817 seconds with return value 0
Press any key to continue . . .
```

8. Write a program to recursively calculate the sum of digits in a given number



The screenshot shows an IDE with a C++ project. The main file, `sum of digits 2.cpp`, contains the following code:

```
1 #include <stdio.h>
2
3 int sumOfDigits(int num) {
4     if (num == 0) {
5         return 0;
6     } else {
7         return (num % 10) + sumOfDigits(num / 10); // Recursive case
8     }
9 }
10
11 int main() {
12     int num;
13     printf("Enter a number: ");
14     scanf("%d", &num);
15     int sum = sumOfDigits(num);
16     printf("Sum of digits of %d is %d\n", num, sum);
17     return 0;
18 }
```

The console output shows the program running successfully for `num = 33`:

```
Enter a number: 33
Sum of digits of 33 is 6

Process exited after 14.05 seconds with return value 0
Press any key to continue . . .
```

