

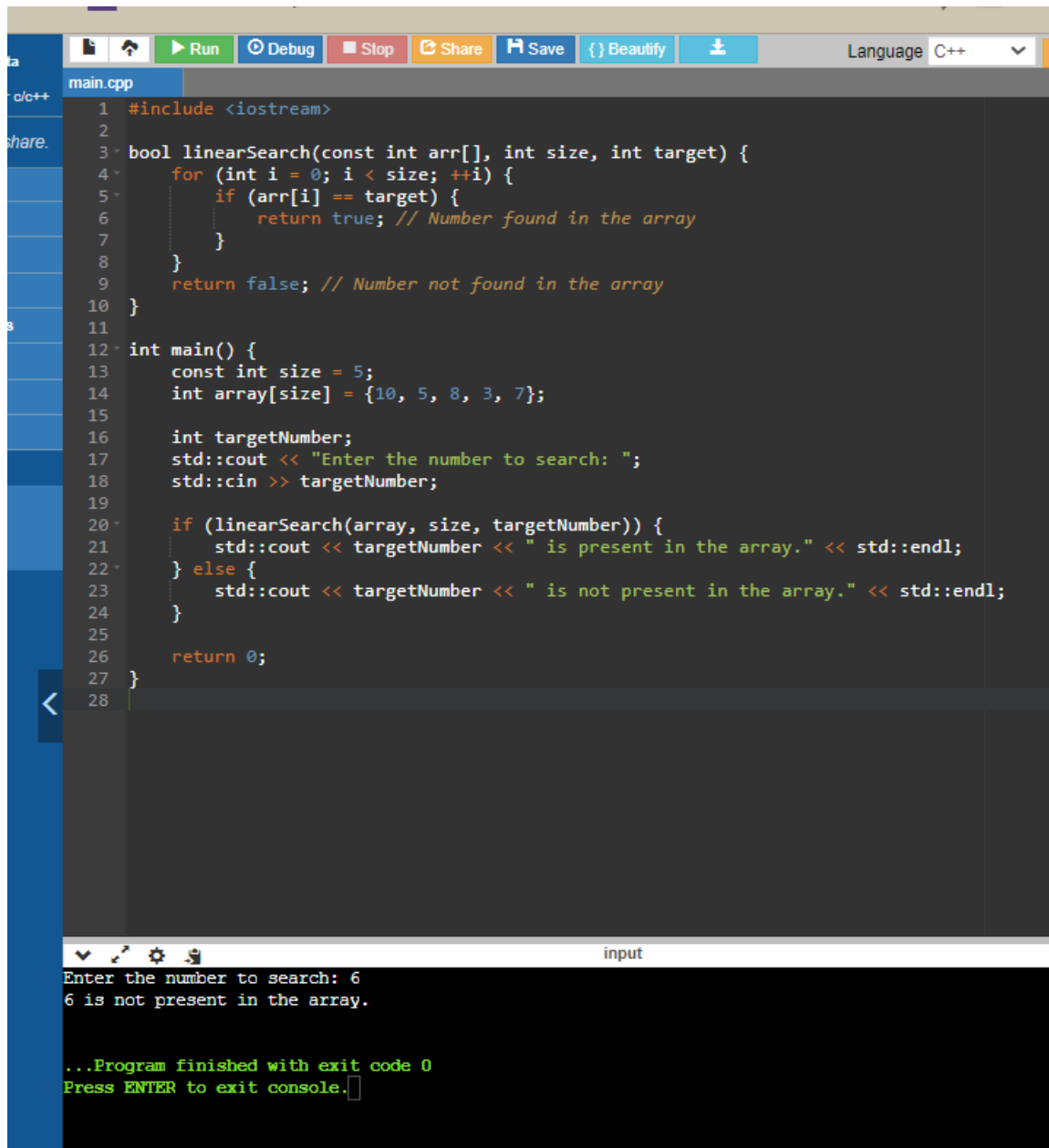
KANISHK BEHL

102105058

3EIC-2

DSA ASSIGN-2

WAP a program to check whether a given number is present in array or not(linear search).



The image shows a screenshot of a C++ IDE. The top toolbar includes buttons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to C++. The editor displays a file named 'main.cpp' with the following code:

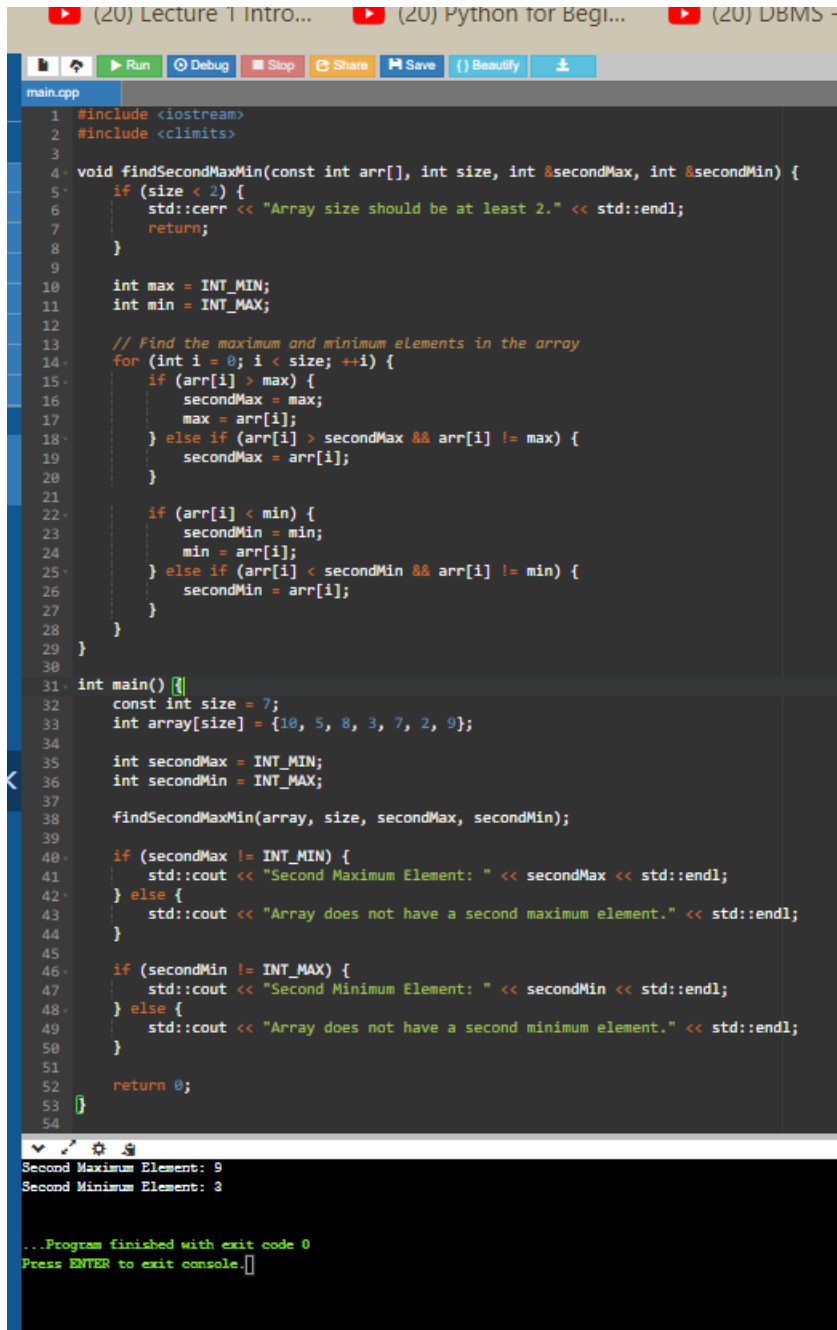
```
1 #include <iostream>
2
3 bool linearSearch(const int arr[], int size, int target) {
4     for (int i = 0; i < size; ++i) {
5         if (arr[i] == target) {
6             return true; // Number found in the array
7         }
8     }
9     return false; // Number not found in the array
10 }
11
12 int main() {
13     const int size = 5;
14     int array[size] = {10, 5, 8, 3, 7};
15
16     int targetNumber;
17     std::cout << "Enter the number to search: ";
18     std::cin >> targetNumber;
19
20     if (linearSearch(array, size, targetNumber)) {
21         std::cout << targetNumber << " is present in the array." << std::endl;
22     } else {
23         std::cout << targetNumber << " is not present in the array." << std::endl;
24     }
25
26     return 0;
27 }
28
```

Below the editor, the console output is shown:

```
input
Enter the number to search: 6
6 is not present in the array.

...Program finished with exit code 0
Press ENTER to exit console.
```

WAP a program to get the second maximum and second minimum elements in an array.



The screenshot shows a C++ IDE with a file named `main.cpp`. The code defines a function `findSecondMaxMin` that takes an array, its size, and references to variables for the second maximum and second minimum. The function iterates through the array, updating these values as it finds elements that are greater than the current second maximum or less than the current second minimum. The `main` function initializes an array `{10, 5, 8, 3, 7, 2, 9}` and calls the function. It then prints the results or messages indicating if a second maximum or minimum exists.

```
1 #include <iostream>
2 #include <limits>
3
4 void findSecondMaxMin(const int arr[], int size, int &secondMax, int &secondMin) {
5     if (size < 2) {
6         std::cerr << "Array size should be at least 2." << std::endl;
7         return;
8     }
9
10    int max = INT_MIN;
11    int min = INT_MAX;
12
13    // Find the maximum and minimum elements in the array
14    for (int i = 0; i < size; ++i) {
15        if (arr[i] > max) {
16            secondMax = max;
17            max = arr[i];
18        } else if (arr[i] > secondMax && arr[i] != max) {
19            secondMax = arr[i];
20        }
21
22        if (arr[i] < min) {
23            secondMin = min;
24            min = arr[i];
25        } else if (arr[i] < secondMin && arr[i] != min) {
26            secondMin = arr[i];
27        }
28    }
29 }
30
31 int main() {
32     const int size = 7;
33     int array[size] = {10, 5, 8, 3, 7, 2, 9};
34
35     int secondMax = INT_MIN;
36     int secondMin = INT_MAX;
37
38     findSecondMaxMin(array, size, secondMax, secondMin);
39
40     if (secondMax != INT_MIN) {
41         std::cout << "Second Maximum Element: " << secondMax << std::endl;
42     } else {
43         std::cout << "Array does not have a second maximum element." << std::endl;
44     }
45
46     if (secondMin != INT_MAX) {
47         std::cout << "Second Minimum Element: " << secondMin << std::endl;
48     } else {
49         std::cout << "Array does not have a second minimum element." << std::endl;
50     }
51
52     return 0;
53 }
```

Second Maximum Element: 9
Second Minimum Element: 3

...Program finished with exit code 0
Press ENTER to exit console.

WAP a program to perform insertion (any location), deletion (any location) and traversal in an array.

```
main.cpp
3 // Function to insert an element at a specified location in the array
4 void insertElement(int arr[], int &size, int position, int value) {
5     if (position < 0 || position > size) {
6         std::cerr << "Invalid position for insertion." << std::endl;
7         return;
8     }
9
10    // Shift elements to make space for the new element
11    for (int i = size - 1; i >= position; --i) {
12        arr[i + 1] = arr[i];
13    }
14
15    // Insert the new element
16    arr[position] = value;
17
18    // Increase the size of the array
19    ++size;
20 }
21
22 // Function to delete an element from a specified location in the array
23 void deleteElement(int arr[], int &size, int position) {
24     if (position < 0 || position >= size) {
25         std::cerr << "Invalid position for deletion." << std::endl;
26         return;
27     }
28
29    // Shift elements to fill the gap left by the deleted element
30    for (int i = position; i < size - 1; ++i) {
31        arr[i] = arr[i + 1];
32    }
33
34    // Decrease the size of the array
35    --size;
36 }
37
38 // Function to traverse and display the elements of the array
39 void traverseArray(const int arr[], int size) {
40     std::cout << "Array Elements: ";
41     for (int i = 0; i < size; ++i) {
42         std::cout << arr[i] << " ";
43     }
44     std::cout << std::endl;
45 }
46
47 int main() {
48     const int maxSize = 100; // Maximum size of the array
49     int arr[maxSize];
50     int size = 0; // Current size of the array
51
52     // Insert elements into the array
53     insertElement(arr, size, 0, 10);
54     insertElement(arr, size, 1, 20);
55     insertElement(arr, size, 2, 30);
56     insertElement(arr, size, 3, 40);
57
58     // Display the array after insertion
59     traverseArray(arr, size);
60
61     // Insert an element at a specified location
62     insertElement(arr, size, 2, 25);
63
64     // Display the array after insertion
65     traverseArray(arr, size);
66
67     // Delete an element from a specified location
68     deleteElement(arr, size, 1);
69
70     // Display the array after deletion
71     traverseArray(arr, size);
72
73     return 0;
74 }
75
```

Array Elements: 10 20 30 40
Array Elements: 10 20 25 30 40
Array Elements: 10 25 30 40

...Program finished with exit code 0
Press ENTER to exit console.

Write a menu drive driven program to perform the addition, multiplication and subtraction of 2 arrays.

```

1 #include <iostream>
2
3 // Function to perform addition of two arrays
4 void addArrays(const int arr1[], const int arr2[], int result[], int size) {
5     for (int i = 0; i < size; ++i) {
6         result[i] = arr1[i] + arr2[i];
7     }
8 }
9
10 // Function to perform subtraction of two arrays
11 void subtractArrays(const int arr1[], const int arr2[], int result[], int size) {
12     for (int i = 0; i < size; ++i) {
13         result[i] = arr1[i] - arr2[i];
14     }
15 }
16
17 // Function to perform multiplication of two arrays
18 void multiplyArrays(const int arr1[], const int arr2[], int result[], int size) {
19     for (int i = 0; i < size; ++i) {
20         result[i] = arr1[i] * arr2[i];
21     }
22 }
23
24 // Function to display the elements of an array
25 void displayArray(const int arr[], int size) {
26     std::cout << "Array Elements: ";
27     for (int i = 0; i < size; ++i) {
28         std::cout << arr[i] << " ";
29     }
30     std::cout << std::endl;
31 }
32
33 int main() {
34     const int maxSize = 5; // Maximum size of the arrays
35     int array1[maxSize], array2[maxSize], result[maxSize];
36     int size;
37
38     // Input size of the arrays
39     std::cout << "Enter the size of the arrays: ";
40     std::cin >> size;
41
42     if (size <= 0 || size > maxSize) {
43         std::cerr << "Invalid size of the arrays." << std::endl;
44         return 1;
45     }
46
47     // Input elements of the first array
48     std::cout << "Enter elements of the first array:" << std::endl;
49     for (int i = 0; i < size; ++i) {
50         std::cout << "Element " << i + 1 << ": ";
51         std::cin >> array1[i];
52     }
53
54     // Input elements of the second array
55     std::cout << "Enter elements of the second array:" << std::endl;
56     for (int i = 0; i < size; ++i) {
57         std::cout << "Element " << i + 1 << ": ";
58         std::cin >> array2[i];
59     }
60
61     int choice;
62     do {
63         // Display menu
64         std::cout << "\nMenu:\n";
65         std::cout << "1. Addition\n";
66         std::cout << "2. Subtraction\n";
67         std::cout << "3. Multiplication\n";
68         std::cout << "4. Exit\n";
69         std::cout << "Enter your choice: ";
70         std::cin >> choice;
71
72         switch (choice) {
73             case 1:
74                 addArrays(array1, array2, result, size);
75                 std::cout << "Result of Addition:" << std::endl;
76                 displayArray(result, size);
77                 break;
78
79             case 2:
80                 subtractArrays(array1, array2, result, size);
81                 std::cout << "Result of Subtraction:" << std::endl;
82                 displayArray(result, size);
83                 break;
84
85             case 3:
86                 multiplyArrays(array1, array2, result, size);
87                 std::cout << "Result of Multiplication:" << std::endl;
88                 displayArray(result, size);
89                 break;
90
91             case 4:
92                 std::cout << "Exiting the program.\n";
93                 break;
94
95             default:
96                 std::cout << "Invalid choice. Please enter a valid option.\n";
97                 break;
98         }
99     } while (choice != 4);
100
101     return 0;
102 }
103
104

```

Enter the size of the arrays: 5

WAP to perform sorting while merging(merge 2 sorted arrays into one sorted array).

```
main.cpp
1 #include <iostream>
2
3 void mergeSortedArrays(const int arr1[], int size1, const int arr2[], int size2, int result[]) {
4     int i = 0, j = 0, k = 0;
5
6     while (i < size1 && j < size2) {
7         if (arr1[i] <= arr2[j]) {
8             result[k++] = arr1[i++];
9         } else {
10             result[k++] = arr2[j++];
11         }
12     }
13
14     // Copy the remaining elements of arr1, if any
15     while (i < size1) {
16         result[k++] = arr1[i++];
17     }
18
19     // Copy the remaining elements of arr2, if any
20     while (j < size2) {
21         result[k++] = arr2[j++];
22     }
23 }
24
25 int main() {
26     const int maxSize = 50; // Maximum size of the arrays
27     int array1[maxSize], array2[maxSize], result[maxSize * 2]; // Assuming maximum size for the merged array
28
29     int size1, size2;
30
31     // Input size of the first array
32     std::cout << "Enter the size of the first sorted array: ";
33     std::cin >> size1;
34
35     if (size1 <= 0 || size1 > maxSize) {
36         std::cerr << "Invalid size of the first array." << std::endl;
37         return 1;
38     }
39
40     // Input elements of the first sorted array
41     std::cout << "Enter elements of the first sorted array:" << std::endl;
42     for (int i = 0; i < size1; ++i) {
43         std::cout << "Element " << i + 1 << ": ";
44         std::cin >> array1[i];
45     }
46
47     // Input size of the second array
48     std::cout << "Enter the size of the second sorted array: ";
49     std::cin >> size2;
50
51     if (size2 <= 0 || size2 > maxSize) {
52         std::cerr << "Invalid size of the second array." << std::endl;
53         return 1;
54     }
55
56     // Input elements of the second sorted array
57     std::cout << "Enter elements of the second sorted array:" << std::endl;
58     for (int i = 0; i < size2; ++i) {
59         std::cout << "Element " << i + 1 << ": ";
60         std::cin >> array2[i];
61     }
62
63     // Merge the two sorted arrays
64     mergeSortedArrays(array1, size1, array2, size2, result);
65
66     // Display the merged and sorted array
67     std::cout << "Merged and Sorted Array:" << std::endl;
68     for (int i = 0; i < size1 + size2; ++i) {
69         std::cout << result[i] << " ";
70     }
71
72     ...Program finished with exit code 0
73     Press ENTER to exit console.
```