

1. Write a C Program to implement following operations

a) traverse



The screenshot shows a C program in a code editor. The program defines a constant `MAX_SIZE` as 100 and implements a `traverse` function that iterates through an array and prints its elements. The `main` function prompts the user for the array size and elements, then calls `traverse`. The output shows the array elements 1, 2, 3, 4, and 5.

```
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 void traverse(int arr[], int size);
4 int main() {
5     int arr[MAX_SIZE];
6     int size;
7     printf("Enter the size of the array (max %d): ", MAX_SIZE);
8     scanf("%d", &size);
9     printf("Enter %d elements:\n", size);
10    for (int i = 0; i < size; ++i) {
11        scanf("%d", &arr[i]);
12    }
13    printf("Array elements: ");
14    traverse(arr, size);
15    return 0;
16 }
17 void traverse(int arr[], int size) {
18     for (int i = 0; i < size; ++i) {
19         printf("%d ", arr[i]);
20     }
21     printf("\n");
22 }
23
```

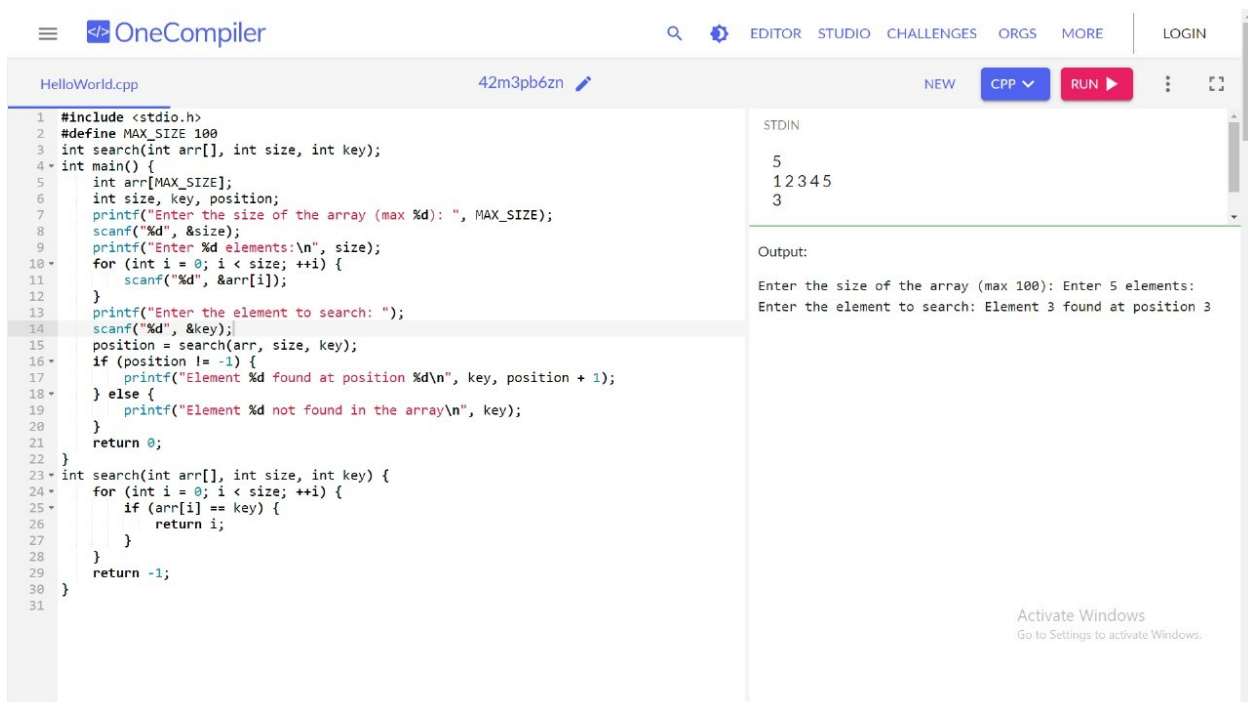
STDIN

```
5
1 2 3 4 5
```

Output:

```
Enter the size of the array (max 100): Enter 5 elements:
Array elements: 1 2 3 4 5
```

b) search



The screenshot shows a C program in a code editor. The program defines a constant `MAX_SIZE` as 100 and implements a `search` function that iterates through an array to find a specific key. The `main` function prompts the user for the array size, elements, and the key to search for. The output shows the key 3 found at position 3.

```
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 int search(int arr[], int size, int key);
4 int main() {
5     int arr[MAX_SIZE];
6     int size, key, position;
7     printf("Enter the size of the array (max %d): ", MAX_SIZE);
8     scanf("%d", &size);
9     printf("Enter %d elements:\n", size);
10    for (int i = 0; i < size; ++i) {
11        scanf("%d", &arr[i]);
12    }
13    printf("Enter the element to search: ");
14    scanf("%d", &key);
15    position = search(arr, size, key);
16    if (position != -1) {
17        printf("Element %d found at position %d\n", key, position + 1);
18    } else {
19        printf("Element %d not found in the array\n", key);
20    }
21    return 0;
22 }
23 int search(int arr[], int size, int key) {
24     for (int i = 0; i < size; ++i) {
25         if (arr[i] == key) {
26             return i;
27         }
28     }
29     return -1;
30 }
31
```

STDIN

```
5
1 2 3 4 5
3
```

Output:

```
Enter the size of the array (max 100): Enter 5 elements:
Enter the element to search: Element 3 found at position 3
```

c) insert

```
HelloWorld.cpp 42m3pb6zn NEW CPP RUN
```

```
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 void insert(int arr[], int *size, int element, int position);
4 int main() {
5     int arr[MAX_SIZE];
6     int size, element, position;
7     printf("Enter the current size of the array (max %d): ", MAX_SIZE);
8     scanf("%d", &size);
9     printf("Enter %d elements:\n", size);
10    for (int i = 0; i < size; ++i) {
11        scanf("%d", &arr[i]);
12    }
13    printf("Enter the element to insert: ");
14    scanf("%d", &element);
15    printf("Enter the position to insert (1 to %d): ", size + 1);
16    scanf("%d", &position);
17    if (position < 1 || position > size + 1) {
18        printf("Invalid position to insert.\n");
19    } else {
20        insert(arr, &size, element, position - 1);
21        printf("Array after insertion: ");
22        for (int i = 0; i < size; ++i) {
23            printf("%d ", arr[i]);
24        }
25        printf("\n");
26    }
27    return 0;
28 }
29
30 void insert(int arr[], int *size, int element, int position) {
31     for (int i = *size - 1; i >= position; --i) {
32         arr[i + 1] = arr[i];
33     }
34     arr[position] = element;
35     *size += 1;
36 }
37
```

```
5
15694
2

Enter the array (max 100): Enter 5 elements:
Insert: Enter the position to insert (1 to 6): Array after insertion: 1 5 2 6 9 4
```

Activate Windows
Go to Settings to activate Windows.

d) delete

```
HelloWorld.cpp 42m3pb6zn NEW CPP RUN
```

```
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 int main() {
4     int array[MAX_SIZE];
5     int size, i, pos;
6     printf("Enter size of the array: ");
7     scanf("%d", &size);
8     printf("Enter elements of the array:\n");
9     for (i = 0; i < size; i++) {
10        scanf("%d", &array[i]);
11    }
12    printf("Enter the position of the element to delete (0-indexed): ");
13    scanf("%d", &pos);
14    if (pos < 0 || pos >= size) {
15        printf("Invalid position!\n");
16    } else {
17        for (i = pos; i < size - 1; i++) {
18            array[i] = array[i + 1];
19        }
20        size--;
21        printf("Array after deletion:\n");
22        for (i = 0; i < size; i++) {
23            printf("%d ", array[i]);
24        }
25        printf("\n");
26    }
27    return 0;
28 }
29
30
```

```
5
10 20 30 40 50
2

Output:
Enter size of the array: Enter elements of the array:
Enter the position of the element to delete (0-indexed): Array
10 20 40 50
```

Activate Windows

e)update



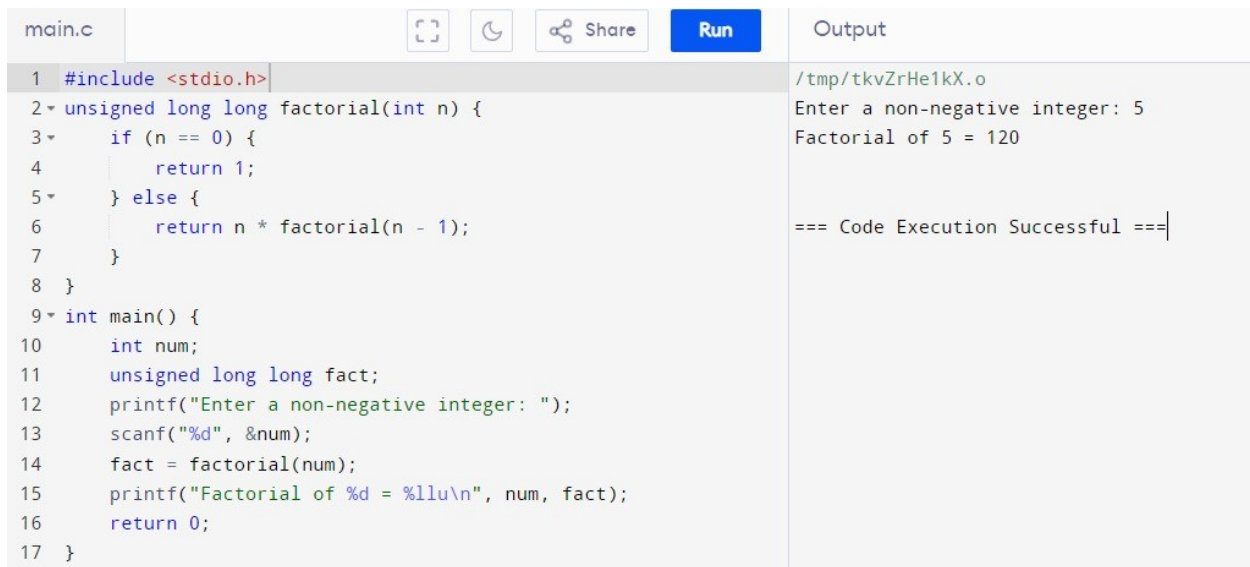
The screenshot shows a C++ IDE with a file named 'HelloWorld.cpp'. The code defines a constant 'MAX_SIZE' as 100 and implements a 'main' function. It prompts the user to enter the size of the array, then the elements of the array, then the position of the element to update (0-indexed), and finally the new value. After updating, it prints the array again. The output window shows the user input: size 5, elements 3 5 7 8 0, position 4, and new value 9. The updated array is then printed as 3 5 7 8 9.

```
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 int main() {
4     int array[MAX_SIZE];
5     int size, i, pos, new_value;
6     printf("Enter size of the array: ");
7     scanf("%d", &size);
8     printf("Enter elements of the array:\n");
9     for (i = 0; i < size; i++) {
10         scanf("%d", &array[i]);
11     }
12     printf("Enter the position of the element to update (0-indexed): ");
13     scanf("%d", &pos);
14     if (pos < 0 || pos >= size) {
15         printf("Invalid position!\n");
16     } else {
17         printf("Enter the new value: ");
18         scanf("%d", &new_value);
19         array[pos] = new_value;
20         printf("Array after updating:\n");
21         for (i = 0; i < size; i++) {
22             printf("%d ", array[i]);
23         }
24         printf("\n");
25     }
26     return 0;
27 }
28
29
```

Output:

Enter size of the array: Enter elements of the array:
Enter the position of the element to update (0-indexed): Enter
3 5 7 8 0
4
9
3 5 7 8 9

2. Writing a recursive function to calculate the factorial of a number.



The screenshot shows a C IDE with a file named 'main.c'. The code defines a recursive function 'factorial' that calculates the factorial of a non-negative integer 'n'. The 'main' function prompts the user to enter a non-negative integer, calls the 'factorial' function, and prints the result. The output window shows the user input 5 and the calculated factorial 120.

```
1 #include <stdio.h>
2 unsigned long long factorial(int n) {
3     if (n == 0) {
4         return 1;
5     } else {
6         return n * factorial(n - 1);
7     }
8 }
9 int main() {
10     int num;
11     unsigned long long fact;
12     printf("Enter a non-negative integer: ");
13     scanf("%d", &num);
14     fact = factorial(num);
15     printf("Factorial of %d = %llu\n", num, fact);
16     return 0;
17 }
```

Output

/tmp/tkvZrHe1kX.o
Enter a non-negative integer: 5
Factorial of 5 = 120
=== Code Execution Successful ===

3. Write a C Program to find duplicate element in an array

main.c	Output
<pre>1 #include <stdio.h> 2 int main() { 3 int arr[] = {1, 2, 3, 4, 2, 7, 8, 8, 3}; 4 int n = sizeof(arr) / sizeof(arr[0]); 5 for (int i = 0; i < n - 1; i++) { 6 for (int j = i + 1; j < n; j++) { 7 if (arr[i] == arr[j]) { 8 printf("Duplicate element: %d\n", arr[j]); 9 } 10 } 11 } 12 return 0; 13 }</pre>	<pre>/tmp/B0nWQqg52G.o Duplicate element: 2 Duplicate element: 3 Duplicate element: 8 === Code Execution Successful ===</pre>

4. Write a C Program to find Max and Min from an array elements

main.c	Output
<pre>1 #include <stdio.h> 2 int main() { 3 int arr[] = {3, 9, 2, 8, 5, 1}; 4 int n = sizeof(arr) / sizeof(arr[0]); 5 int max = arr[0], min = arr[0]; 6 for (int i = 1; i < n; i++) { 7 if (arr[i] > max) { 8 max = arr[i]; 9 } 10 if (arr[i] < min) { 11 min = arr[i]; 12 } 13 } 14 printf("Maximum element in the array: %d\n", max); 15 printf("Minimum element in the array: %d\n", min); 16 return 0; 17 }</pre>	<pre>/tmp/zyfPmqff7d.o Maximum element in the array: 9 Minimum element in the array: 1 === Code Execution Successful ===</pre>

5. Given a number n. the task is to print the Fibonacci series and the sum of the series using recursion.

input: n=10

output: Fibonacci series

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Sum: 88

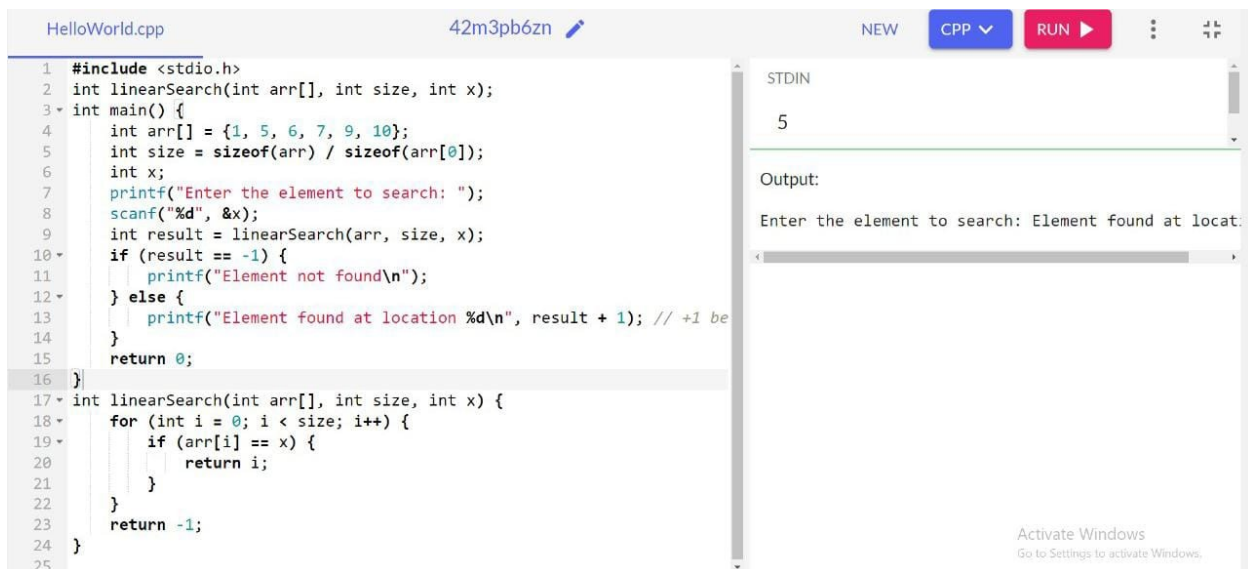
6. You are given an array arr in increasing order. Find the element x from arr using linear search.

Example 1: arr={ 1,5,6,7,9,10},X=6

Output : Element found at location 2

Example 2: arr={ 1,5,6,7,9,10},X=11

Output : Element not found at location 2



```
1 #include <stdio.h>
2 int linearSearch(int arr[], int size, int x);
3 int main() {
4     int arr[] = {1, 5, 6, 7, 9, 10};
5     int size = sizeof(arr) / sizeof(arr[0]);
6     int x;
7     printf("Enter the element to search: ");
8     scanf("%d", &x);
9     int result = linearSearch(arr, size, x);
10    if (result == -1) {
11        printf("Element not found\n");
12    } else {
13        printf("Element found at location %d\n", result + 1); // +1 because location starts from 1
14    }
15    return 0;
16 }
17 int linearSearch(int arr[], int size, int x) {
18     for (int i = 0; i < size; i++) {
19         if (arr[i] == x) {
20             return i;
21         }
22     }
23     return -1;
24 }
25
```

STDIN

5

Output:

Enter the element to search: Element found at location 2

Activate Windows
Go to Settings to activate Windows.