

1. Write a C program to perform Matrix Multiplication.

Aim:

To write a C program that performs Matrix Multiplication on two given matrices and displays the resulting matrix.

Algorithm:

1. Start
2. Read matrix A of size $m \times n$
3. Read matrix B of size $n \times p$
4. Initialize matrix C of size $m \times p$ with all zeros
5. For each row i from 0 to m-1
 For each column j from 0 to p-1
 Set $C[i][j] = 0$
 For each k from 0 to n-1
 $C[i][j] += A[i][k] * B[k][j]$
6. Display matrix C
7. End

Input:

Enter rows and columns of matrix A =2,3

Enter rows and columns of second matrix=3,2

Output:

Enter no of elements of matrix A

1 2 3

4 5 6

Enter no of elements of matrix B

7 8

9 10

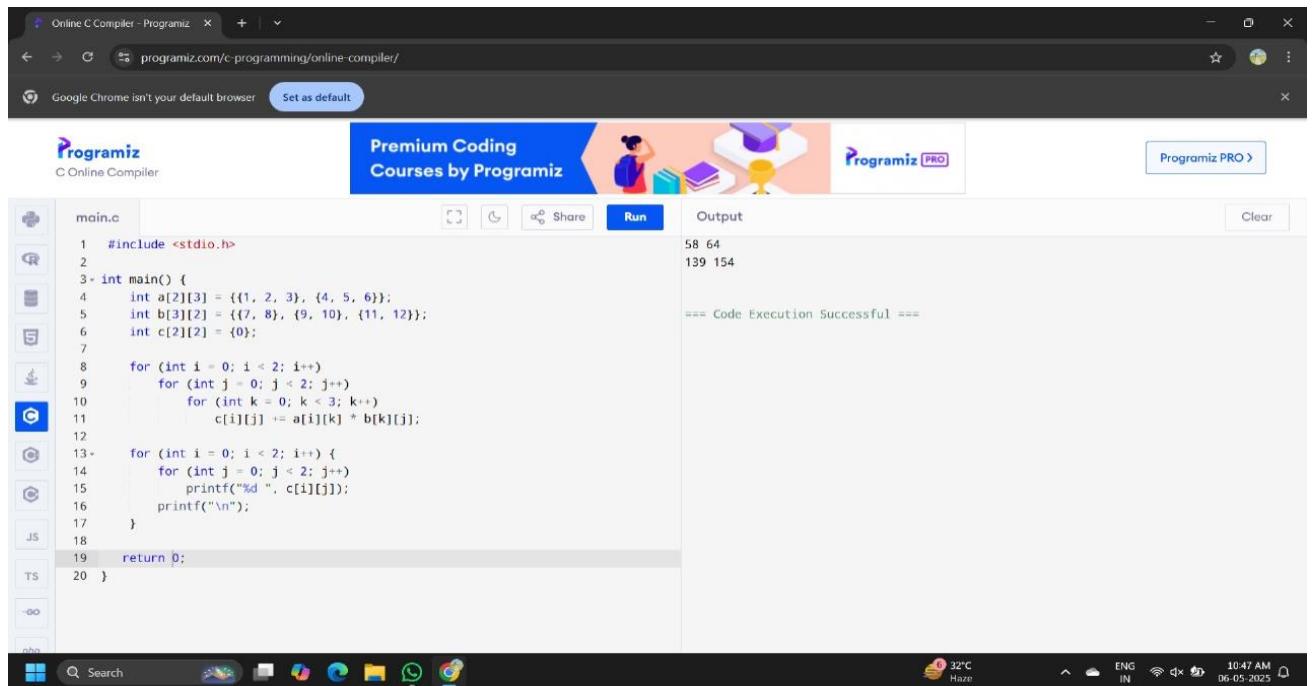
11 12

Resultant matrix after multiplication

58 64

139 154

Code:



The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features the Programiz logo and a banner for "Premium Coding Courses by Programiz". The main content area displays a C program in a code editor, with the filename `main.c`. The code defines two 2x3 matrices, `a` and `b`, and a 2x3 matrix `c` initialized to zero. It then performs matrix multiplication using nested loops and prints the resulting matrix `c`. The output window on the right shows the results: `58 64` and `139 154`, followed by the message `=== Code Execution Successful ===`. The bottom of the browser window shows the Windows taskbar with the search bar, task view button, and system tray displaying the date and time as 10:47 AM on 06-05-2025.

```
1 #include <stdio.h>
2
3 int main() {
4     int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
5     int b[3][2] = {{7, 8}, {9, 10}, {11, 12}};
6     int c[2][2] = {0};
7
8     for (int i = 0; i < 2; i++)
9         for (int j = 0; j < 2; j++)
10             for (int k = 0; k < 3; k++)
11                 c[i][j] += a[i][k] * b[k][j];
12
13     for (int i = 0; i < 2; i++) {
14         for (int j = 0; j < 2; j++)
15             printf("%d ", c[i][j]);
16         printf("\n");
17     }
18
19     return 0;
20 }
```

Output

```
58 64
139 154

=== Code Execution Successful ===
```

2. Write a C program to find Odd or Even number from a given set of numbers.

Aim:

To write a C program that determines whether a given set of numbers are odd or even

Algorithm:

1. Start
2. Read the total number of elements, say `n`
3. For `i = 0` to `n - 1`
 - a. Read number `x`
 - b. If `x % 2 == 0`
Display "x is Even"
 - c. Else
Display "x is Odd"

4. End

Input:

Enter how many numbers = 5

Enter 5 numbers:

12 7 9 4 10

Output:

12 is Even

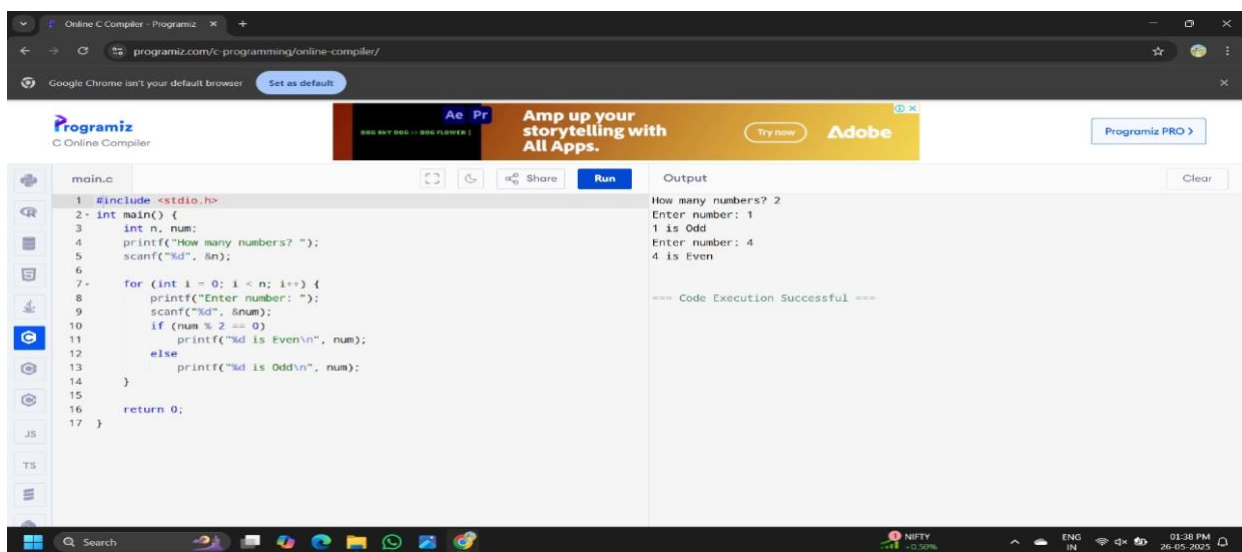
7 is odd

9 is odd

4 is Even

10 is Even

Code:



The screenshot shows the Programiz Online C Compiler interface. The code editor on the left contains the following C program:

```
1 #include <stdio.h>
2 int main() {
3     int n, num;
4     printf("How many numbers? ");
5     scanf("%d", &n);
6
7     for (int i = 0; i < n; i++) {
8         printf("Enter number: ");
9         scanf("%d", &num);
10        if (num % 2 == 0)
11            printf("%d is Even\n", num);
12        else
13            printf("%d is Odd\n", num);
14    }
15
16    return 0;
17 }
```

The output window on the right shows the execution results:

```
How many numbers? 2
Enter number: 1
1 is Odd
Enter number: 4
4 is Even

=== Code Execution Successful ===
```

The browser's address bar shows the URL `programiz.com/c-programming/online-compiler/`. The Windows taskbar at the bottom indicates the time is 01:38 PM on 26-05-2023.

3. Write a C program to find Factorial of a given number without using Recursion.

Aim:

To write a C program to find the factorial of a given number using an iterative (non-recursive) approach.

Algorithm:

1. Start
2. Read the integer n
3. If $n < 0$, display "Invalid input" and stop
4. Initialize a variable fact = 1
5. For $i = 1$ to n
 fact = fact * i

6. Display fact

7. End

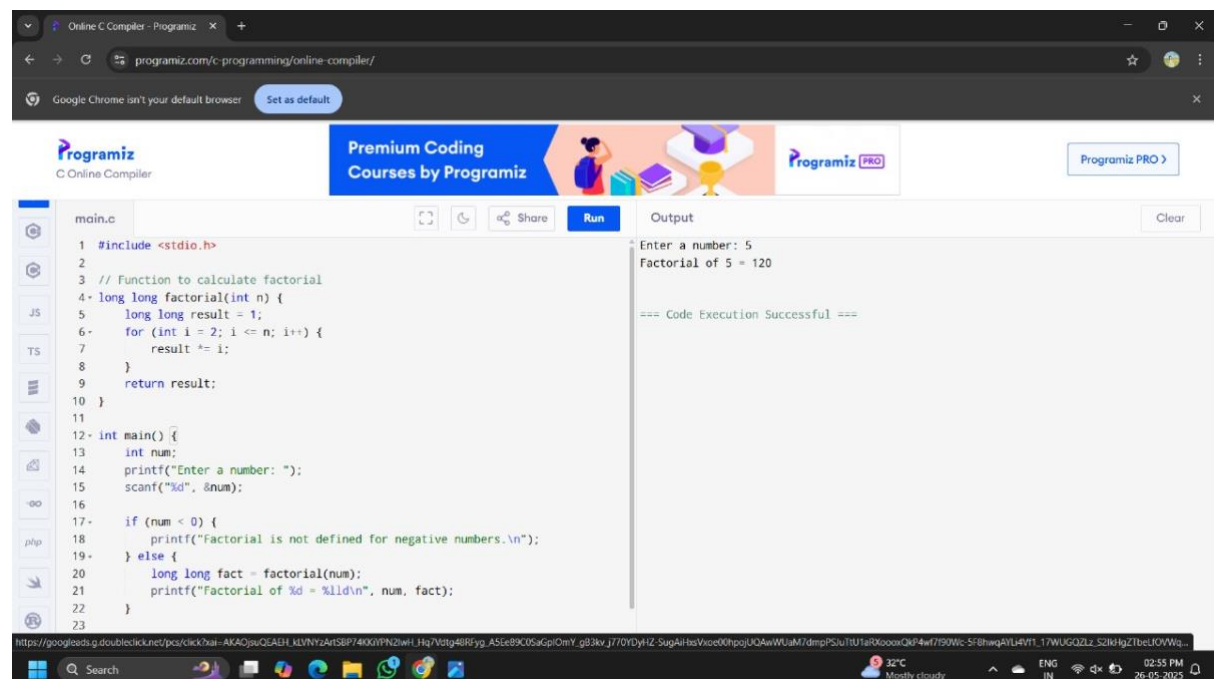
Input:

Enter a number = 5

Output:

Factorial of 5 = 120

Code:



The screenshot displays a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features the Programiz logo and navigation links for "Premium Coding Courses by Programiz" and "Programiz PRO". The main content area is divided into two panels: a code editor on the left and an output window on the right. The code editor shows a C program named `main.c` that calculates the factorial of a number using an iterative approach. The code includes a function `factorial` that takes an integer `n` and returns the factorial as a `long long` integer. The `main` function prompts the user to enter a number, reads the input, and calls the `factorial` function. The output window shows the result of the program execution: "Enter a number: 5" followed by "Factorial of 5 = 120". A status message at the bottom of the output window indicates "=== Code Execution Successful ===". The browser's address bar and taskbar are visible at the bottom of the image.

```
1 #include <stdio.h>
2
3 // Function to calculate factorial
4 long long factorial(int n) {
5     long long result = 1;
6     for (int i = 2; i <= n; i++) {
7         result *= i;
8     }
9     return result;
10 }
11
12 int main() {
13     int num;
14     printf("Enter a number: ");
15     scanf("%d", &num);
16
17     if (num < 0) {
18         printf("Factorial is not defined for negative numbers.\n");
19     } else {
20         long long fact = factorial(num);
21         printf("Factorial of %d = %lld\n", num, fact);
22     }
23 }
```

Output

Enter a number: 5
Factorial of 5 = 120

=== Code Execution Successful ===

4. Write a C program to find Fibonacci series without using Recursion.

Aim:

To write a C program to generate the Fibonacci series up to n terms without using recursion.

Algorithm:

1. Start.

2. Read the number of terms n.

3. Initialize three variables:

first = 0,

second = 1,

next.

4. Print first and second.

5. Repeat steps 6–7 for i from 3 to n:

6. Set next = first + second

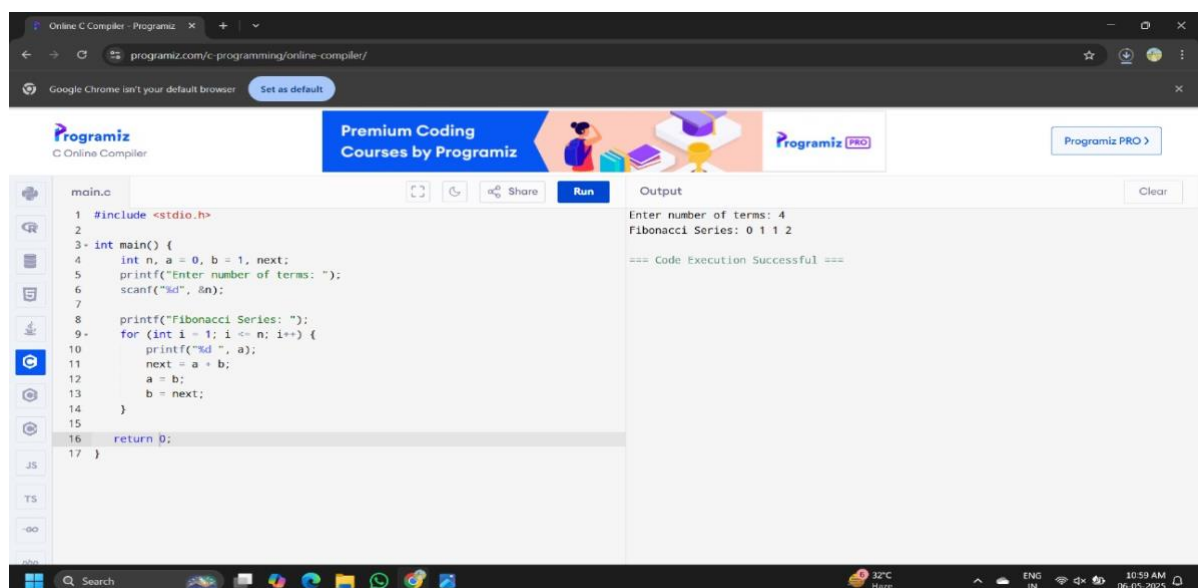
7. Print next.

8. Update first = second, second = next.

9. Stop.

Input: Enter the number of terms : 4

Output : Fibonacci Series: 0 1 1 2



The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features the Programiz logo and a banner for "Premium Coding Courses by Programiz". Below the banner is a code editor with a file named `main.c`. The code is as follows:

```
1 #include <stdio.h>
2
3 int main() {
4     int n, a = 0, b = 1, next;
5     printf("Enter number of terms: ");
6     scanf("%d", &n);
7
8     printf("Fibonacci Series: ");
9     for (int i = 1; i <= n; i++) {
10        printf("%d ", a);
11        next = a + b;
12        a = b;
13        b = next;
14    }
15
16    return 0;
17 }
```

To the right of the code editor is an "Output" panel. It displays the following text:

```
Enter number of terms: 4
Fibonacci Series: 0 1 1 2
=== Code Execution Successful ===
```

The bottom of the image shows a Windows taskbar with the search bar, task view button, and several open applications. The system tray on the right indicates a temperature of 32°C, language set to ENG IN, and the date/time as 10:59 AM on 06-05-2025.

5. Write a C program to find Factorial of a given number using Recursion.

Aim:

To write a C program to find the factorial of a given number using recursion.

Algorithm:

1. Start.

2. Define a recursive function factorial(int n):

If $n == 0$ or $n == 1$, return 1.

Else return $n * \text{factorial}(n - 1)$.

3. In the main() function:

Read an integer number n from the user.

Call the recursive function factorial(n) and store the result.

Display the result.

4. Stop.

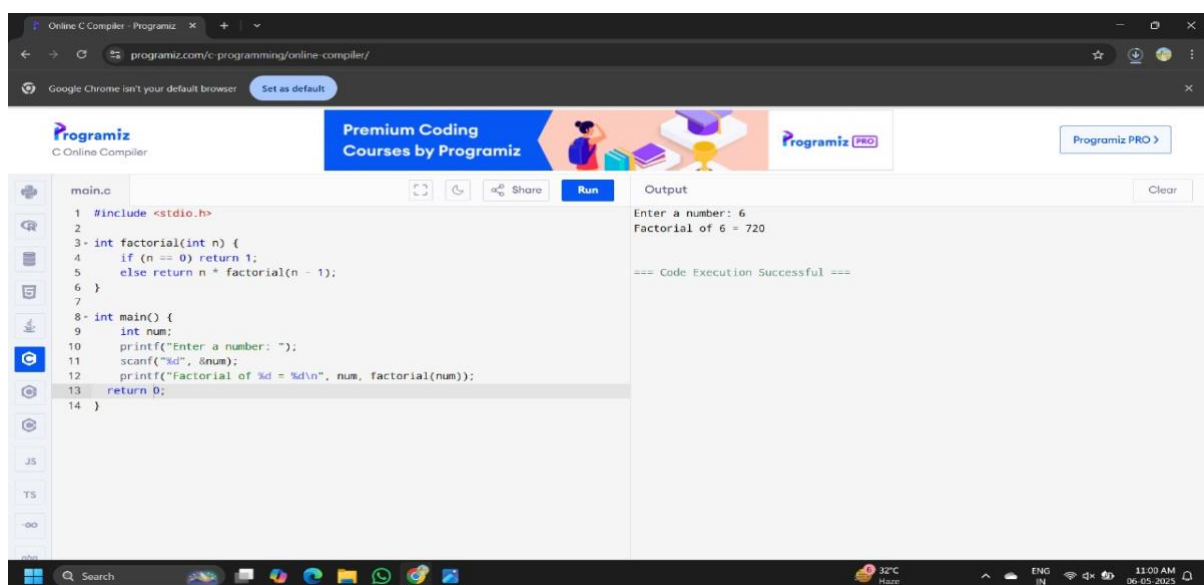
Input:

Enter a Number : 6

Output:

Factorial of 6 = 720

Code:



The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features the Programiz logo and a banner for "Premium Coding Courses by Programiz". Below the banner is a code editor with a file named `main.c`. The code defines a recursive function `factorial` and a `main` function that prompts the user for a number, reads it, and prints the factorial result. The code is as follows:

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

To the right of the code editor is an "Output" window. It displays the input "Enter a number: 6", the output "Factorial of 6 = 720", and a status message "=== Code Execution Successful ===". The browser's address bar and Windows taskbar are also visible at the top and bottom of the screen.

6. Write a C program to find Fibonacci series using Recursion.

Aim:

To write a C program to generate the Fibonacci series up to n terms using recursion.

Algorithm:

1. Start.

2. Define a recursive function fibonacci(int n):

If $n == 0$, return 0.

If $n == 1$, return 1.

Else return $\text{fibonacci}(n - 1) + \text{fibonacci}(n - 2)$.

3. In the main() function:

Read the number of terms n.

Loop from 0 to n - 1:

Call fibonacci(i) and print the result.

4. Stop.

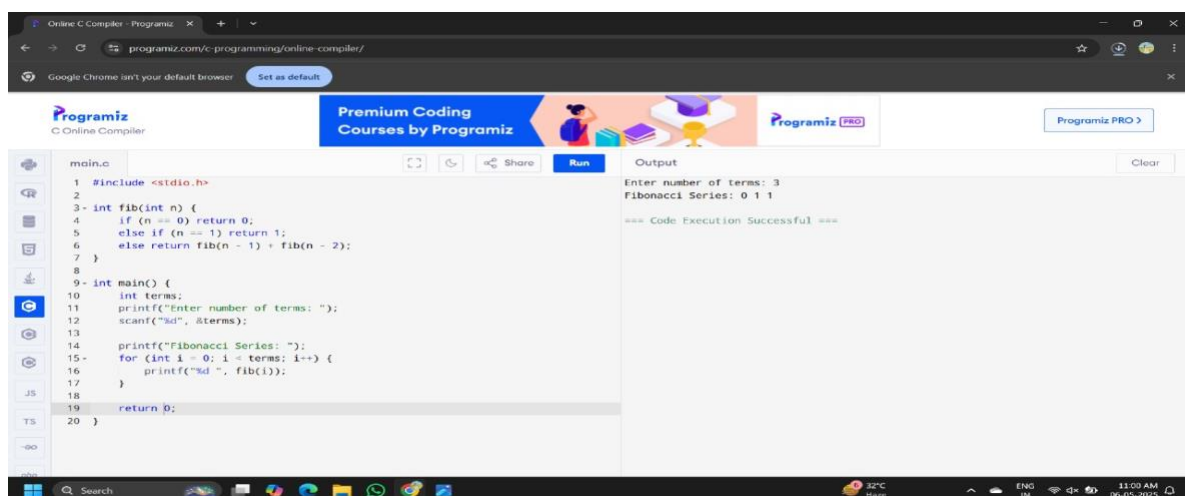
Input:

Enter the number of terms: 3

Output:

Fibonacci Series: 0 1 1

Code:



The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features the Programiz logo and a banner for "Premium Coding Courses by Programiz". The main content area displays a C program for calculating the Fibonacci series using recursion. The code is as follows:

```
1 #include <stdio.h>
2
3 int fib(int n) {
4     if (n == 0) return 0;
5     else if (n == 1) return 1;
6     else return fib(n - 1) + fib(n - 2);
7 }
8
9 int main() {
10     int terms;
11     printf("Enter number of terms: ");
12     scanf("%d", &terms);
13
14     printf("Fibonacci Series: ");
15     for (int i = 0; i < terms; i++) {
16         printf("%d ", fib(i));
17     }
18
19     return 0;
20 }
```

The output window on the right shows the execution results:

```
Enter number of terms: 3
Fibonacci Series: 0 1 1
=== Code Execution Successful ===
```

The Windows taskbar at the bottom indicates the system time is 11:00 AM on 06-05-2025.

7. Write a C program to implement Array operations such as Insert, Delete and Display.

Aim:

To write a C program that performs basic array operations: insertion of an element, deletion of an element, and displaying the array elements.

Algorithm:

1. Start

2. Declare an array and variables for size and position.

3. Display a menu with choices: Insert, Delete, Display, and Exit.

4. For Insert:

Input the element and the position.

Shift elements to the right from the position to the end.

Insert the element.

Increment size.

5. For Delete:

Input the position.

Shift elements to the left from the position to the end.

Decrease size.

6. For Display:

Print all elements up to current size.

7. Repeat until Exit is selected.

8. End

Input: 1. Insert

2. delete

3. display

4.Exit

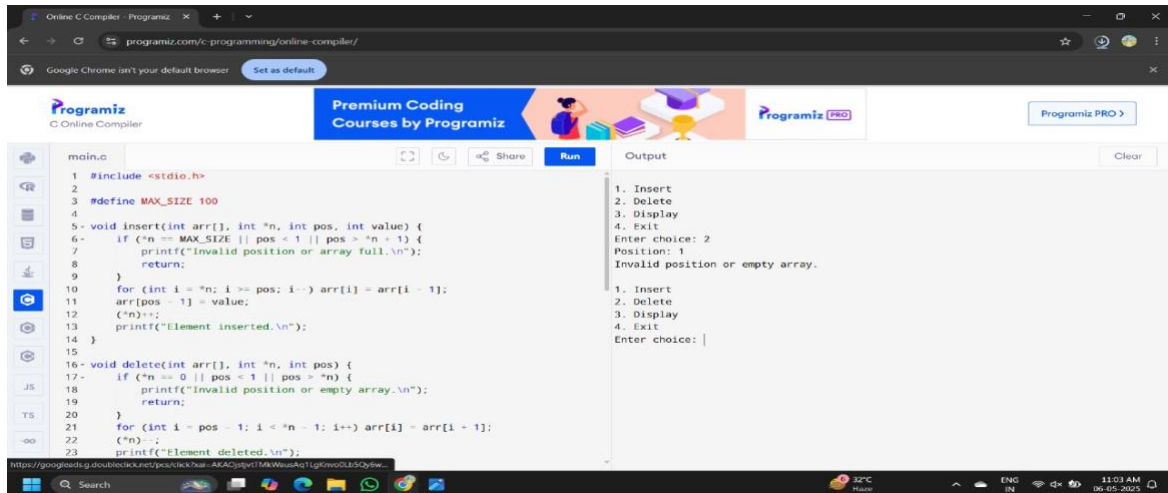
Enter choice : 2

Position : 1

Output:

Invalid position or empty array

Code:



```
1 #include <stdio.h>
2
3 #define MAX_SIZE 100
4
5 void insert(int arr[], int *n, int pos, int value) {
6     if (*n == MAX_SIZE || pos < 1 || pos > *n + 1) {
7         printf("Invalid position or array full.\n");
8         return;
9     }
10    for (int i = *n; i >= pos; i--) arr[i] = arr[i - 1];
11    arr[pos - 1] = value;
12    (*n)++;
13    printf("Element inserted.\n");
14 }
15
16 void delete(int arr[], int *n, int pos) {
17     if (*n == 0 || pos < 1 || pos > *n) {
18         printf("Invalid position or empty array.\n");
19         return;
20     }
21     for (int i = pos - 1; i < *n - 1; i++) arr[i] = arr[i + 1];
22     (*n)--;
23     printf("Element deleted.\n");
24 }
```

Output

```
1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 2
Position: 1
Invalid position or empty array.

1. Insert
2. Delete
3. Display
4. Exit
Enter choice: |
```

8. Write a C program to search a number using Linear Search method.

Aim

To search for a given number in an array using the Linear Search method.

Algorithm

1. Start from the first element of the array.
2. Compare the target element with each element of the array.
3. If the target element is found, return its index.
4. If the target element is not found after traversing the entire array, return -1 (or a suitable indicator).

Input:

Enter no of elements in array : 3

Enter three integers: 1 2 4

Enter the no to search : 4

Output:

4 found at position 3

9. Write a C program to search a number using Binary Search method.

Aim

To search for a given number in a sorted array using the Binary Search method.

Algorithm

1. Find the middle element of the array.
2. Compare the target element with the middle element.
3. If the target element is equal to the middle element, return its index.
4. If the target element is less than the middle element, repeat the process with the left half of the array.
5. If the target element is greater than the middle element, repeat the process with the right half of the array.
6. If the target element is not found after the search space is empty, return -1 (or a suitable indicator).

Input:

Enter number of elements in array: 2

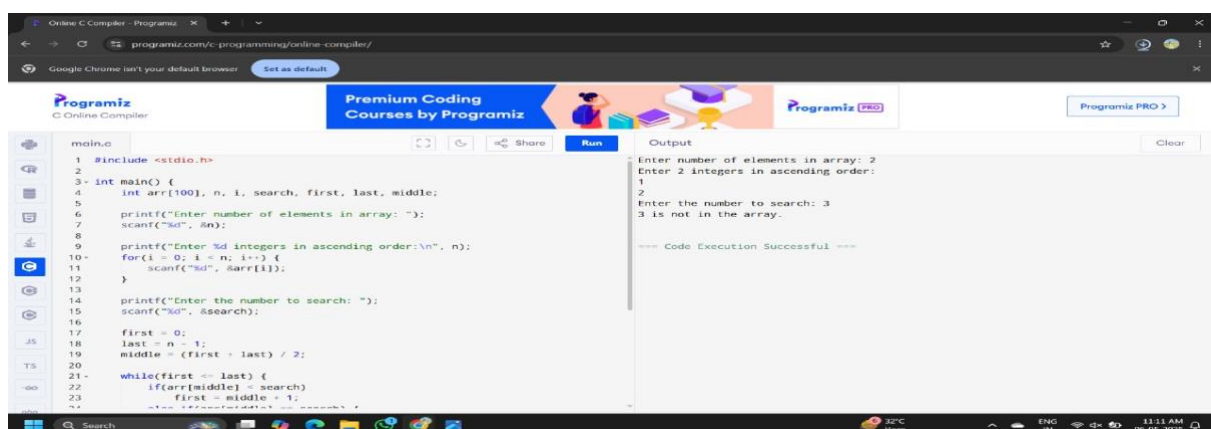
Enter 2 integers in ascending order: 1 2

Enter the no to search : 3

Output:

3 is not in the array

Code:



```
main.c
1 #include <stdio.h>
2
3 int main() {
4     int arr[100], n, i, search, first, last, middle;
5
6     printf("Enter number of elements in array: ");
7     scanf("%d", &n);
8
9     printf("Enter %d integers in ascending order:\n", n);
10    for(i = 0; i < n; i++) {
11        scanf("%d", &arr[i]);
12    }
13
14    printf("Enter the number to search: ");
15    scanf("%d", &search);
16
17    first = 0;
18    last = n - 1;
19    middle = (first + last) / 2;
20
21    while(first <= last) {
22        if(arr[middle] == search)
23            first = middle + 1;
24        else if(arr[middle] < search)
25            first = middle + 1;
26        else
27            last = middle - 1;
28        middle = (first + last) / 2;
29    }
30
31    if(arr[middle] == search)
32        printf("Number found at index %d", middle);
33    else
34        printf("Number not found");
35
36    return 0;
37 }
```

Output

```
Enter number of elements in array: 2
Enter 2 integers in ascending order:
1
2
Enter the number to search: 3
3 is not in the array.

==== Code Execution Successful ====
```

10. Write a C program to implement Linked list operations.

Aim

To implement basic operations on a singly linked list, including insertion, deletion, and traversal.

Algorithm

1. Node structure: Define a node structure with an integer data field and a pointer to the next node.

2. Insertion:

- Insert at the beginning: Create a new node and update the head pointer.
- Insert at the end: Traverse the list to find the last node and append a new node.

3. Deletion:

- Delete from the beginning: Update the head pointer to the next node.
- Delete from the end: Traverse the list to find the second last node and update its next pointer to NULL.

4. Traversal: Traverse the list and print the data of each node.

Input :

1. Insert

2. delete

3. display

4.Exit

Enter your choice:3

Output:

Linked list : 2 > null

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // Define a node
5 struct Node {
6     int data;
7     struct Node* next;
8 };
9
10 struct Node* head = NULL;
11
12 // Function to insert a node at the end
13 void insert(int value) {
14     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
15     struct Node* temp = head;
16     newNode->data = value;
17     newNode->next = NULL;
18     if (head == NULL) {
19         head = newNode;
20     } else {
21         while (temp->next != NULL) {
22             temp = temp->next;
23         }
24         temp->next = newNode;
25     }
26 }
27
28 // Function to delete a node
29 void deleteNode() {
30     struct Node* temp = head;
31     if (temp == NULL) {
32         printf("List is empty\n");
33     } else {
34         printf("Enter value to delete: ");
35         int val;
36         while (temp != NULL) {
37             printf("%d ", temp->data);
38             temp = temp->next;
39         }
40         printf("\n");
41         while (1) {
42             val = 0;
43             temp = head;
44             while (temp != NULL) {
45                 if (temp->data == val) {
46                     if (temp == head) {
47                         head = temp->next;
48                     } else {
49                         struct Node* prev = head;
50                         while (prev->next != temp) {
51                             prev = prev->next;
52                         }
53                         prev->next = temp->next;
54                     }
55                     free(temp);
56                     break;
57                 }
58                 temp = temp->next;
59             }
60             printf("Enter value to delete: ");
61             if (val == 0) {
62                 break;
63             }
64         }
65     }
66 }
67
68 // Function to display the list
69 void display() {
70     struct Node* temp = head;
71     while (temp != NULL) {
72         printf("%d ", temp->data);
73         temp = temp->next;
74     }
75     printf("\n");
76 }
77
78 // Function to exit
79 void exit() {
80     printf("Exiting...\n");
81     exit(0);
82 }
83
84 int main() {
85     int choice;
86     while (1) {
87         printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
88         printf("Enter your choice: ");
89         choice = 0;
90         while (choice < 1 || choice > 4) {
91             printf("Invalid choice\n");
92             printf("Enter your choice: ");
93             choice = 0;
94         }
95         switch (choice) {
96             case 1:
97                 printf("Enter value to insert: ");
98                 int val;
99                 while (1) {
100                     val = 0;
101                     while (val == 0) {
102                         printf("Enter value to insert: ");
103                         val = 0;
104                     }
105                     insert(val);
106                     printf("Enter value to insert: ");
107                     val = 0;
108                 }
109                 break;
110             case 2:
111                 deleteNode();
112                 break;
113             case 3:
114                 display();
115                 break;
116             case 4:
117                 exit();
118                 break;
119         }
120     }
121 }
```

