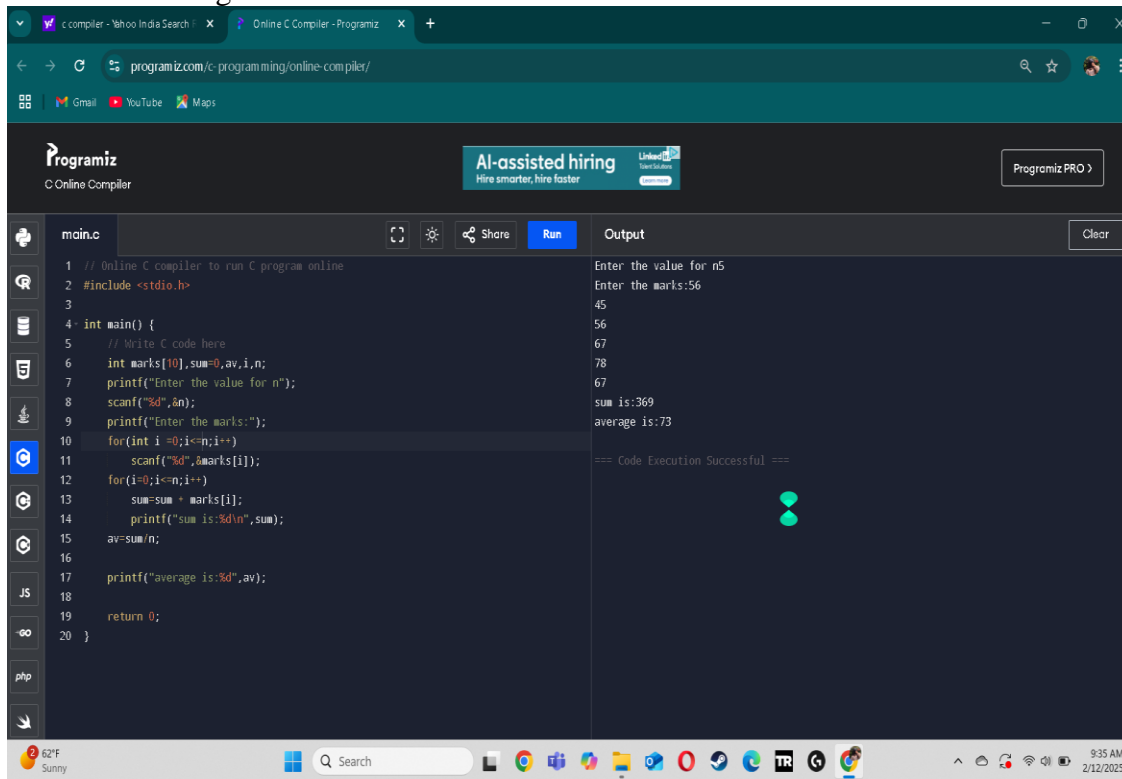


DS lab

1.sum and average



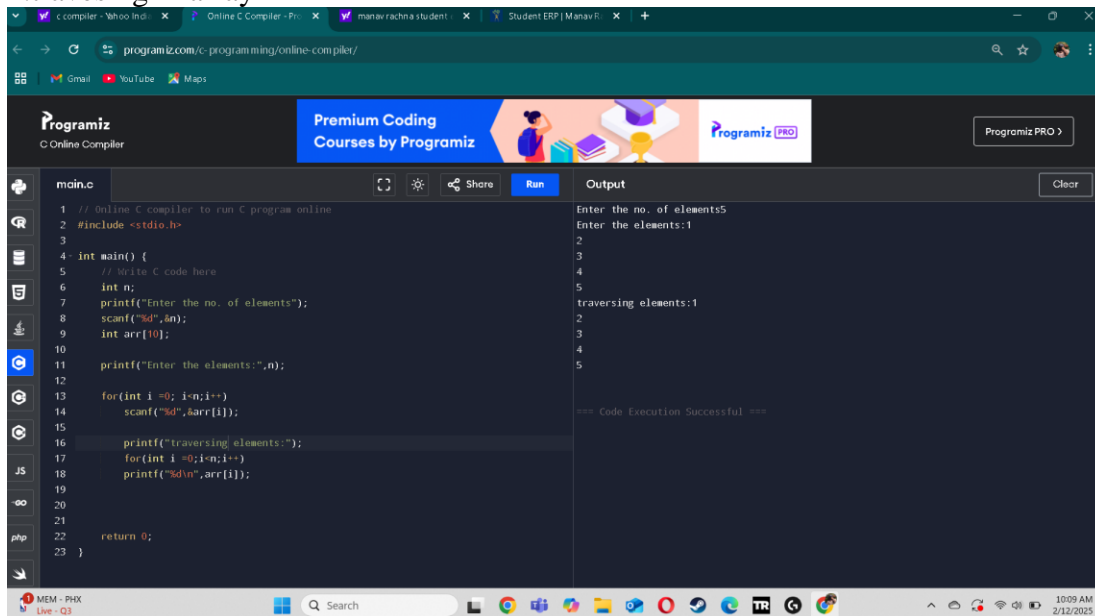
The screenshot shows the Programiz Online C Compiler interface. The code in `main.c` is as follows:

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     int marks[10], sum=0, av, i, n;
7     printf("Enter the value for n");
8     scanf("%d", &n);
9     printf("Enter the marks:");
10    for(int i =0; i<n; i++)
11        scanf("%d", &marks[i]);
12    for(i=0; i<n; i++)
13        sum=sum + marks[i];
14    printf("sum is:%d\n", sum);
15    av=sum/n;
16
17    printf("average is:%d", av);
18
19    return 0;
20 }
```

The output shows the program execution with the following input and output:

```
Enter the value for n5
Enter the marks:56
45
56
67
78
67
sum is:369
average is:73
=== Code Execution Successful ===
```

2.travesing in array



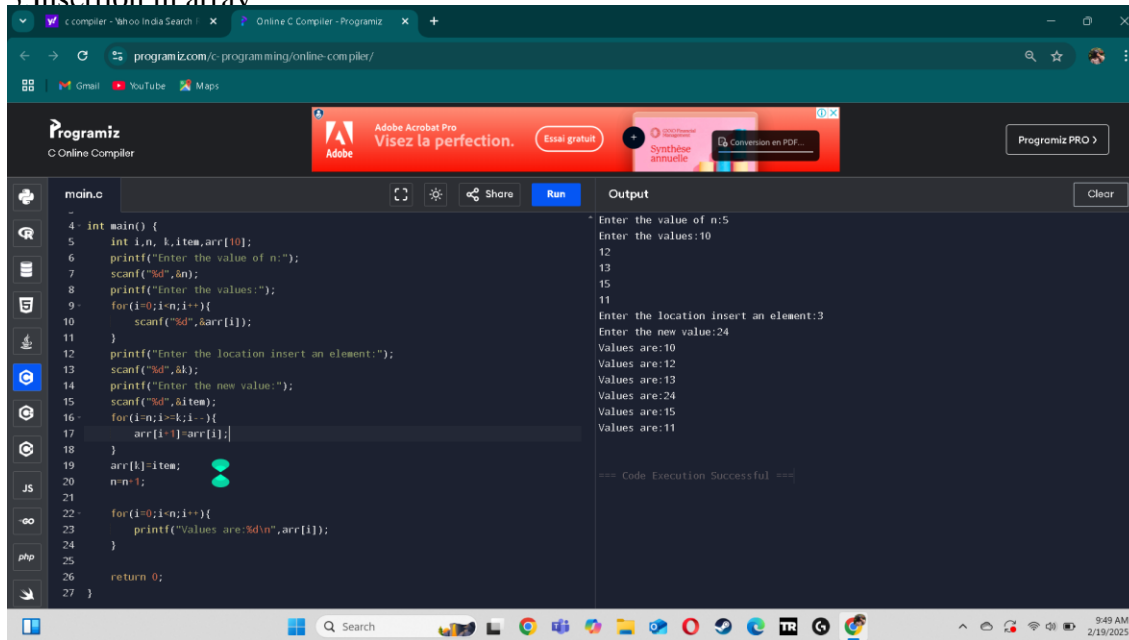
The screenshot shows the Programiz Online C Compiler interface. The code in `main.c` is as follows:

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     int n;
7     printf("Enter the no. of elements");
8     scanf("%d", &n);
9     int arr[10];
10
11    printf("Enter the elements:", n);
12
13    for(int i =0; i<n; i++)
14        scanf("%d", &arr[i]);
15
16    printf("traversing elements:");
17    for(int i =0; i<n; i++)
18        printf("%d\n", arr[i]);
19
20
21
22    return 0;
23 }
```

The output shows the program execution with the following input and output:

```
Enter the no. of elements5
Enter the elements:1
2
3
4
5
traversing elements:1
2
3
4
5
=== Code Execution Successful ===
```

3 Insertion in array



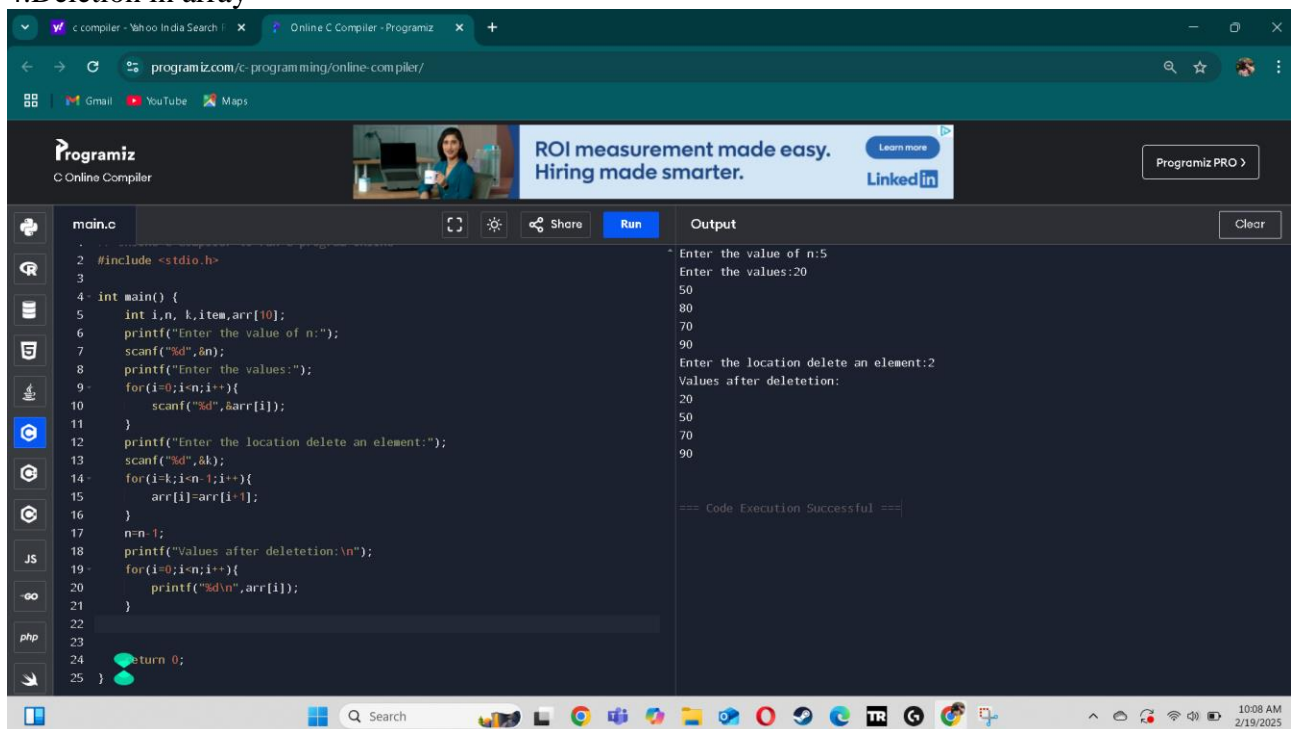
The screenshot shows the Programiz Online C Compiler interface. The code in `main.c` is as follows:

```
1 int main() {
2     int i, n, k, item, arr[10];
3     printf("Enter the value of n:");
4     scanf("%d", &n);
5     printf("Enter the values:");
6     for(i=0; i<n; i++){
7         scanf("%d", &arr[i]);
8     }
9     printf("Enter the location insert an element:");
10    scanf("%d", &k);
11    printf("Enter the new value:");
12    scanf("%d", &item);
13    for(i=n; i>=k; i--){
14        arr[i+1]=arr[i];
15    }
16    arr[k]=item;
17    n=n+1;
18    for(i=0; i<n; i++){
19        printf("Values are:%d\n", arr[i]);
20    }
21    return 0;
22 }
```

The output shows the user inputting `n=5`, values `10 12 13 15 11`, location `3`, and new value `24`. The final output is:

```
Enter the value of n:5
Enter the values:10
12
13
15
11
Enter the location insert an element:3
Enter the new value:24
Values are:10
Values are:12
Values are:13
Values are:24
Values are:15
Values are:11
=== Code Execution Successful ===
```

4. Deletion in array



The screenshot shows the Programiz Online C Compiler interface. The code in `main.c` is as follows:

```
1 #include <stdio.h>
2
3 int main() {
4     int i, n, k, item, arr[10];
5     printf("Enter the value of n:");
6     scanf("%d", &n);
7     printf("Enter the values:");
8     for(i=0; i<n; i++){
9         scanf("%d", &arr[i]);
10    }
11    printf("Enter the location delete an element:");
12    scanf("%d", &k);
13    for(i=k; i<n-1; i++){
14        arr[i]=arr[i+1];
15    }
16    n=n-1;
17    printf("Values after deletion:\n");
18    for(i=0; i<n; i++){
19        printf("%d\n", arr[i]);
20    }
21    return 0;
22 }
```


The output shows the user inputting `n=5`, values `50 80 70 90`, and location `2`. The final output is:

```
Enter the value of n:5
Enter the values:20
50
80
70
90
Enter the location delete an element:2
Values after deletion:
20
50
70
90
=== Code Execution Successful ===
```


5. Binary to decimal

Programiz

C Online Compiler



Bharat Mandapam
Multi Purpose Hall-L3, Delhi
Entry From Gate No. 7



main.c

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <math.h>
4
5 int main() {
6     long long binary;
7     int decimal = 0, i = 0, digit;
8
9     printf("Enter a binary number: ");
10    scanf("%lld", &binary);
11
12    while (binary != 0) {
13        digit = binary % 10;
14        decimal += digit * pow(2, i);
15        binary /= 10;
16        i++;
17    }
18
19    printf("Decimal equivalent: %d\n", decimal);
20
21    return 0;
22 }
```

Run

Share

Output


Enter a binary number: 3
Decimal equivalent: 3

=== Code Execution Successful ===

6. 2D multiplication in array

Programiz

C Online Compiler



**Better tools,
more opportunities.**

[Buy now](#)

**Adobe
Creative Cloud**

main.c

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 int main() {
4     int a[10][10], b[10][10], result[10][10];
5     int r1, c1, r2, c2;
6
7     printf("Enter rows and columns of first matrix: ");
8     scanf("%d %d", &r1, &c1);
9
10    printf("Enter rows and columns of second matrix: ");
11    scanf("%d %d", &r2, &c2);
12
13    if (c1 != r2) {
14        printf("Matrix multiplication not possible.\n");
15        return 0;
16    }
17    printf("Enter elements of first matrix:\n");
18    for (int i = 0; i < r1; i++)
19        for (int j = 0; j < c1; j++)
20            scanf("%d", &a[i][j]);
21    printf("Enter elements of second matrix:\n");
22    for (int i = 0; i < r2; i++)
23        for (int j = 0; j < c2; j++)
24            scanf("%d", &b[i][j]);
25    for (int i = 0; i < r1; i++)
26        for (int j = 0; j < c2; j++)
27            result[i][j] = 0;
28    for (int i = 0; i < r1; i++) {
29        for (int j = 0; j < c2; j++) {
30            for (int k = 0; k < c1; k++) {
31                result[i][j] += a[i][k] * b[k][j];
32            }
33        }
34    }
35
36    // Print result
```

Run

Share

Output



Enter rows and columns of first matrix: 4
2
Enter rows and columns of second matrix: 34
4
Matrix multiplication not possible.

=== Code Execution Successful ===5


7. transposing in Array


Programiz

C Online Compiler



Bharat Mandapam
Multi Purpose Hall-L3, Delhi
Entry From Gate No. 7





```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     int a[10][10], transpose[10][10];
6     int row, col;
7
8     printf("Enter number of rows and columns: ");
9     scanf("%d %d", &row, &col);
10
11     printf("Enter matrix elements:\n");
12     for (int i = 0; i < row; i++)
13         for (int j = 0; j < col; j++)
14             scanf("%d", &a[i][j]);
15
16     // Transpose logic
17     for (int i = 0; i < row; i++)
18         for (int j = 0; j < col; j++)
19             transpose[j][i] = a[i][j];
20
21     printf("Transpose of the matrix:\n");
22     for (int i = 0; i < col; i++) {
23         for (int j = 0; j < row; j++) {
24             printf("%d ", transpose[i][j]);
25         }
26         printf("\n");
27     }
28
29     return 0;
30 }
31
```

Run

Output

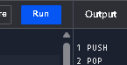

Enter number of rows and columns: 2
2
Enter matrix elements:
3
3
4
5
Transpose of the matrix:
3 4
3 5

=== Code Execution Successful ===


8. Stack operation - push and pop


Programiz

C Online Compiler



Bharat Mandapam
Multi Purpose Hall-L3, Delhi
Entry From Gate No. 7





```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4 #define max 10
5 int top = -1;
6 void push(int[]);
7 void pop(int[]);
8 void display(int[]);
9 void main() {
10     int stk[max], n;
11     char ch;
12     do {
13         printf("\n1 PUSH\n");
14         printf("\n2 POP\n");
15         printf("\n3 DISPLAY\n");
16         printf("\n4 exit\n");
17         printf("\npress: ");
18         scanf("%d", &n);
19         switch(n)
20         {
21             case 1: push(stk);
22                     break;
23             case 2: pop(stk);
24                     break;
25             case 3: display(stk);
26                     break;
27             case 4: exit(0);
28                     break;
29         }
30         printf("\ndo you want to continue(Y/N):");
31         while (getchar() != '\n');
32         scanf("%c", &ch);
33         while(ch=='Y'||ch=='y');
34     }
35 }
36 void push(int stk[])
37 {
38     int item;
39     printf("\nEnter the value that you want to insert:");
40     scanf("%d", &item);
41     if(top==max-1)
42         printf("\nstack is overflow insertion is not possible");
43     else
44         stk[++top]=item;
45 }
46 void pop(int stk[])
47 {
48     if(top==-1)
49         printf("\nstack is underflow deletion is not possible");
50     else

```

Run

Output

1 PUSH
2 POP
3 DISPLAY
4 exit

1press:2

stack is underflow deletion is not possible
DO YOU WANT TO CONTINUE(Y/N)

1 PUSH
2 POP
3 DISPLAY
4 exit

1press:

9. Queue operation - insertion deletion display

```
main.c
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4 #define max 10
5 int front = -1, rear = -1;
6
7 void insertion(int []);
8 void deletion(int []);
9 void display(int []);
10
11 void main() {
12     int queue[max], n;
13     char ch;
14
15     do {
16         printf("\n1-> Insertion");
17         printf("\n2-> Deletion");
18         printf("\n3-> Display");
19         printf("\n4-> Exit\n");
20         printf("\nEnter your choice: ");
21         scanf("%d", &n);
22
23         switch(n) {
24             case 1: insertion(queue); break;
25             case 2: deletion(queue); break;
26             case 3: display(queue); break;
27             case 4: return; // Exit the program
28             default: printf("Invalid choice. Please try again.\n");
29         }
30
31         printf("\nDo you want to continue (Y/N)? ");
32         while ((getchar() != '\n')); // Clear input buffer
33         scanf("%c", &ch);
34     } while(ch == 'Y' || ch == 'y');
35 }
36
37 void insertion(int queue[]) {
38     int item;
39     if((rear + 1) % max == front) {
40         printf("\nQueue overflow! Insertion not possible.");
41     } else {
42         printf("\nEnter the value to insert: ");
43         scanf("%d", &item);
44         if(front == -1) {
45             front = rear = 0;
46         } else {
47             rear = (rear + 1) % max;
48         }
49         queue[rear] = item;
50         printf("%d inserted into the queue.", item);
51     }
52 }
```

Output

1-> Insertion
2-> Deletion
3-> Display
4-> Exit

Enter your choice: 2

Queue underflow! Deletion not possible.
Do you want to continue (Y/N)? n

*** Code Exited With Errors ***

10. Linklist operation- insertion deletion display

```
main.c
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 // Node structure
6 struct Node {
7     int data;
8     struct Node* next;
9 };
10
11 struct Node* head = NULL;
12
13 // Insert at the end
14 void insert(int value) {
15     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
16     newNode->data = value;
17     newNode->next = NULL;
18
19     if (head == NULL) {
20         head = newNode;
21     } else {
22         struct Node* temp = head;
23         while (temp->next != NULL) {
24             temp = temp->next;
25         }
26         temp->next = newNode;
27         printf("Inserted %d\n", value);
28     }
29 }
30
31 // Delete a node by value
32 void delete(int value) {
33     struct Node* temp = head, *prev = NULL;
34
35     // If head needs to be deleted
36     if (temp != NULL && temp->data == value) {
37         head = temp->next;
38         free(temp);
39         printf("Deleted %d\n", value);
40         return;
41     }
42
43     // Search for the node
44     while (temp != NULL && temp->data != value) {
45         prev = temp;
46         temp = temp->next;
47     }
48
49     // Not found
50     if (temp == NULL) {
51         printf("Value %d not found.\n", value);
52     }
53 }
```

Output

Menu:
1. Insert
2. Delete
3. Display
4. Exit

Enter choice: 2

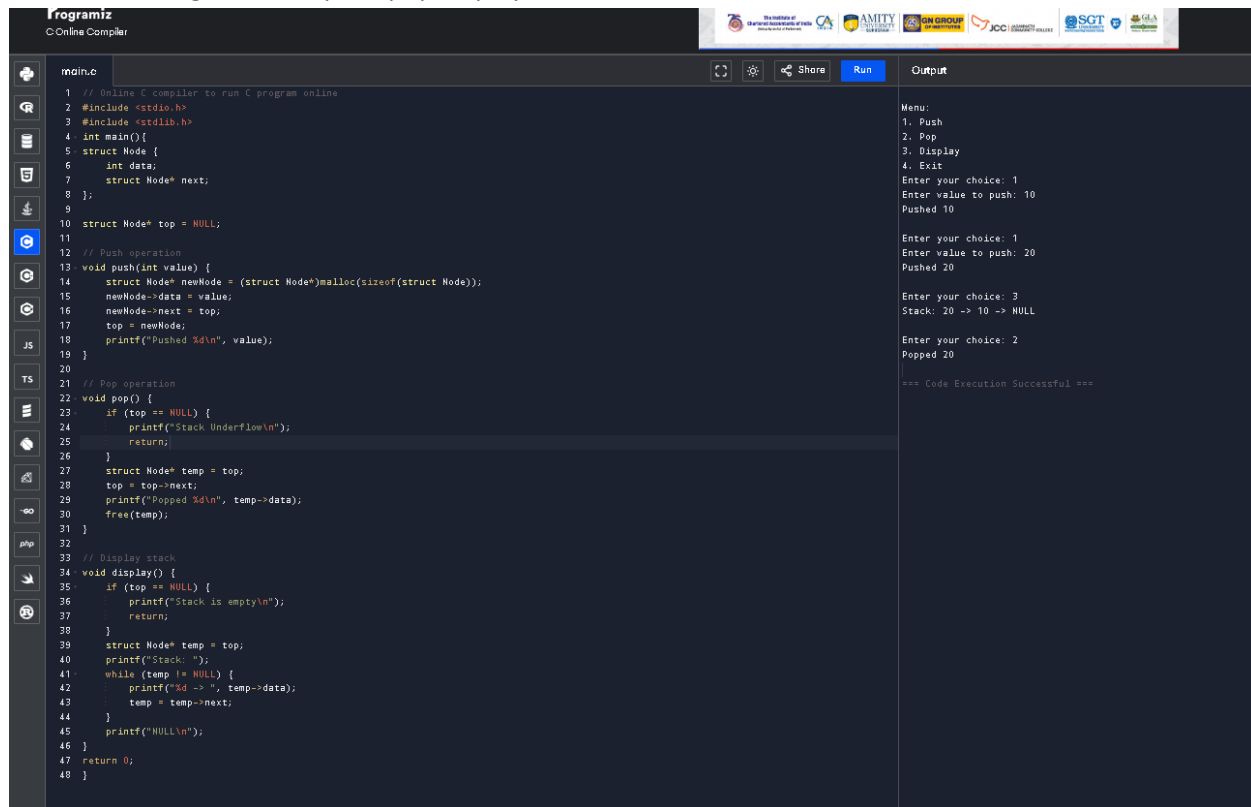
Enter value to delete: 3

Value 3 not found.

Menu:
1. Insert
2. Delete
3. Display
4. Exit

Enter choice: |

11. Stack through linklist - push pop display



```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4 int main() {
5     struct Node {
6         int data;
7         struct Node* next;
8     };
9
10    struct Node* top = NULL;
11
12    // Push operation
13    void push(int value) {
14        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
15        newNode->data = value;
16        newNode->next = top;
17        top = newNode;
18        printf("Pushed %d\n", value);
19    }
20
21    // Pop operation
22    void pop() {
23        if (top == NULL) {
24            printf("Stack Underflow\n");
25            return;
26        }
27        struct Node* temp = top;
28        top = top->next;
29        printf("Popped %d\n", temp->data);
30        free(temp);
31    }
32
33    // Display stack
34    void display() {
35        if (top == NULL) {
36            printf("Stack is empty\n");
37            return;
38        }
39        struct Node* temp = top;
40        printf("Stack: ");
41        while (temp != NULL) {
42            printf("%d -> ", temp->data);
43            temp = temp->next;
44        }
45        printf("NULL\n");
46    }
47    return 0;
48 }
```

Output

Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter value to push: 10
Pushed 10

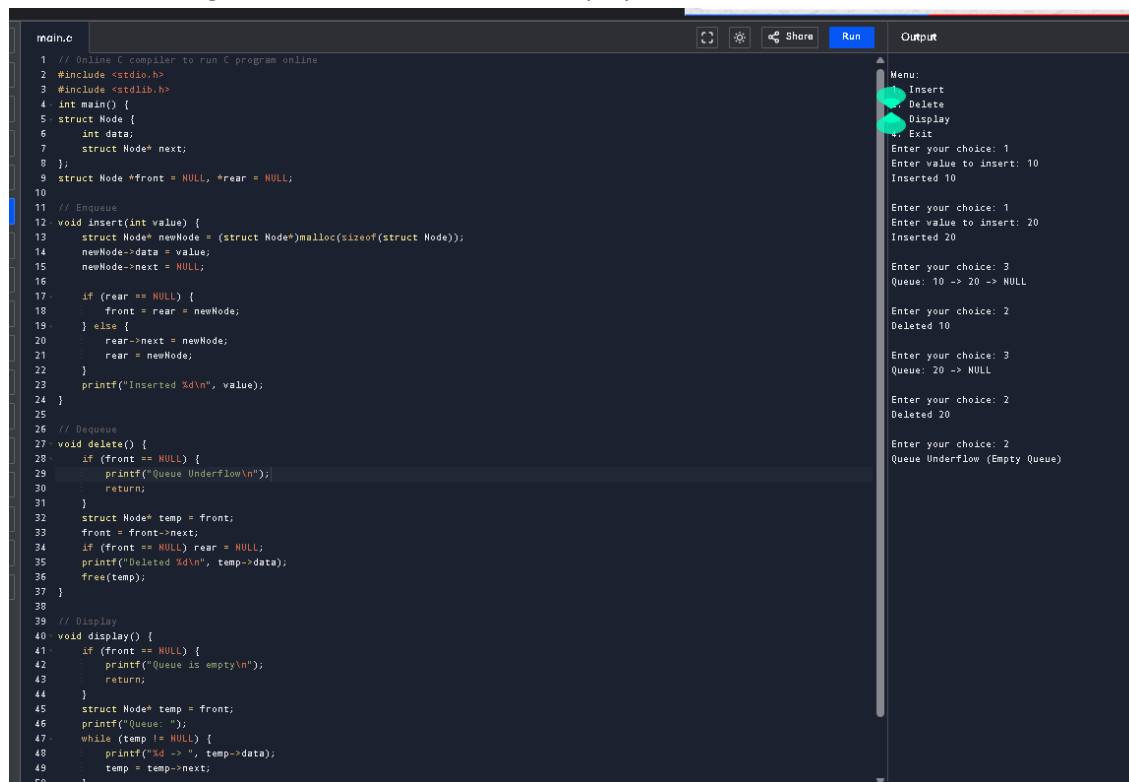
Enter your choice: 1
Enter value to push: 20
Pushed 20

Enter your choice: 3
Stack: 20 -> 10 -> NULL

Enter your choice: 2
Popped 20

*** Code Execution Successful ***

12. Queue through linklist- insertion deletion display



```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4 int main() {
5     struct Node {
6         int data;
7         struct Node* next;
8     };
9
10    struct Node *front = NULL, *rear = NULL;
11
12    // Enqueue
13    void insert(int value) {
14        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
15        newNode->data = value;
16        newNode->next = NULL;
17
18        if (rear == NULL) {
19            front = rear = newNode;
20        } else {
21            rear->next = newNode;
22            rear = newNode;
23        }
24        printf("Inserted %d\n", value);
25    }
26
27    // Dequeue
28    void delete() {
29        if (front == NULL) {
30            printf("Queue Underflow\n");
31            return;
32        }
33        struct Node* temp = front;
34        front = front->next;
35        if (front == NULL) rear = NULL;
36        printf("Deleted %d\n", temp->data);
37        free(temp);
38    }
39
40    // Display
41    void display() {
42        if (front == NULL) {
43            printf("Queue is empty\n");
44            return;
45        }
46        struct Node* temp = front;
47        printf("Queue: ");
48        while (temp != NULL) {
49            printf("%d -> ", temp->data);
50            temp = temp->next;
51        }
52        printf("NULL\n");
53    }
54 }
```

Output

Menu:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 10
Inserted 10

Enter your choice: 1
Enter value to insert: 20
Inserted 20

Enter your choice: 3
Queue: 10 -> 20 -> NULL

Enter your choice: 2
Deleted 10

Enter your choice: 3
Queue: 20 -> NULL

Enter your choice: 2
Deleted 20

Enter your choice: 2
Queue Underflow (Empty Queue)

13. Tree traversal

Programiz

C Online Compiler

Special Partners

AMITY

IGN GROUP

JCC

SGT

GLA

main.c

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 struct Node {
6     int data;
7     struct Node* left;
8     struct Node* right;
9 };
10 struct Node* createNode(int value) {
11     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
12     newNode->data = value;
13     newNode->left = newNode->right = NULL;
14     return newNode;
15 }
16 struct Node* insert(struct Node* root, int value) {
17     if (root == NULL) return createNode(value);
18     if (value < root->data)
19         root->left = insert(root->left, value);
20     else
21         root->right = insert(root->right, value);
22     return root;
23 }
24 void inorder(struct Node* root) {
25     if (root != NULL) {
26         inorder(root->left);
27         printf("%d ", root->data);
28         inorder(root->right);
29     }
30 }
31 void preorder(struct Node* root) {
32     if (root != NULL) {
33         printf("%d ", root->data);
34         preorder(root->left);
35         preorder(root->right);
36     }
37 }
38
39 void postorder(struct Node* root) {
40     if (root != NULL) {
41         postorder(root->left);
42         postorder(root->right);
43         printf("%d ", root->data);
44     }
45 }
46
```

Output

Inorder traversal: 20 30 40 50 60 70 80
Preorder traversal: 50 30 20 40 70 60 80
Postorder traversal: 20 40 30 60 80 70 50

14. Tree Searching - Quick sort

```
Online Compiler
main.c
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4 struct Node {
5     int data;
6     struct Node* left;
7     struct Node* right;
8 };
9 struct Node* createNode(int value) {
10     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
11     newNode->data = value;
12     newNode->left = newNode->right = NULL;
13     return newNode;
14 }
15 struct Node* insert(struct Node* root, int value) {
16     if (root == NULL)
17         return createNode(value);
18     if (value < root->data)
19         root->left = insert(root->left, value);
20     else
21         root->right = insert(root->right, value);
22     return root;
23 }
24 int search(struct Node* root, int key) {
25     if (root == NULL)
26         return 0;
27     if (root->data == key)
28         return 1;
29     if (key < root->data)
30         return search(root->left, key);
31     else
32         return search(root->right, key);
33 }
34 void inorder(struct Node* root) {
35     if (root != NULL) {
36         inorder(root->left);
37         printf("%d ", root->data);
38         inorder(root->right);
39     }
40 }
41 int partition(int arr[], int low, int high) {
42     int pivot = arr[high]; // last element as pivot
43     int i = low - 1; // smaller element index
44     for (int j = low; j < high; j++) {
45         if (arr[j] < pivot) {
46             i++;
47             int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
48         }
49     }
50     int temp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = temp;
51     return (i + 1);
52 }
53 void quickSort(int arr[], int low, int high) {
54     if (low < high) {
55         int pi = partition(arr, low, high);
56         quickSort(arr, low, pi - 1);
57         quickSort(arr, pi + 1, high);
58     }
59 }
60 int main() {
61     // --- Tree Searching ---
62     struct Node* root = NULL;
63     int treeData[] = {50, 30, 70, 20, 40, 60, 80};
64     int n = sizeof(treeData)/sizeof(treeData[0]);
65
66     for (int i = 0; i < n; i++)
67         root = insert(root, treeData[i]);
68
69     printf("BST Inorder Traversal: ");
70     inorder(root);
71     printf("\n");
72
73     int key;
74     printf("Enter element to search in tree: ");
75     scanf("%d", &key);
76
77     if (search(root, key))
78         printf("%d found in the tree.\n", key);
79     else
80         printf("%d not found in the tree.\n", key);
81
82     int arr[] = {25, 10, 99, 5, 30, 75};
83     int size = sizeof(arr) / sizeof(arr[0]);
84
85     printf("\nOriginal array: ");
86     for (int i = 0; i < size; i++) printf("%d ", arr[i]);
87
88     quickSort(arr, 0, size - 1);
89
90     printf("\nSorted array (Quick Sort): ");
91     for (int i = 0; i < size; i++) printf("%d ", arr[i]);
92
93     return 0;
94 }
```

Output

```
BST Inorder Traversal: 20 30 40 50 60 70 80
Enter element to search in tree: 30
30 found in the tree.

Original array: 25 10 99 5 30 75
Sorted array (Quick Sort): 5 10 25 30 75 99

*** Code Execution Successful ***
```