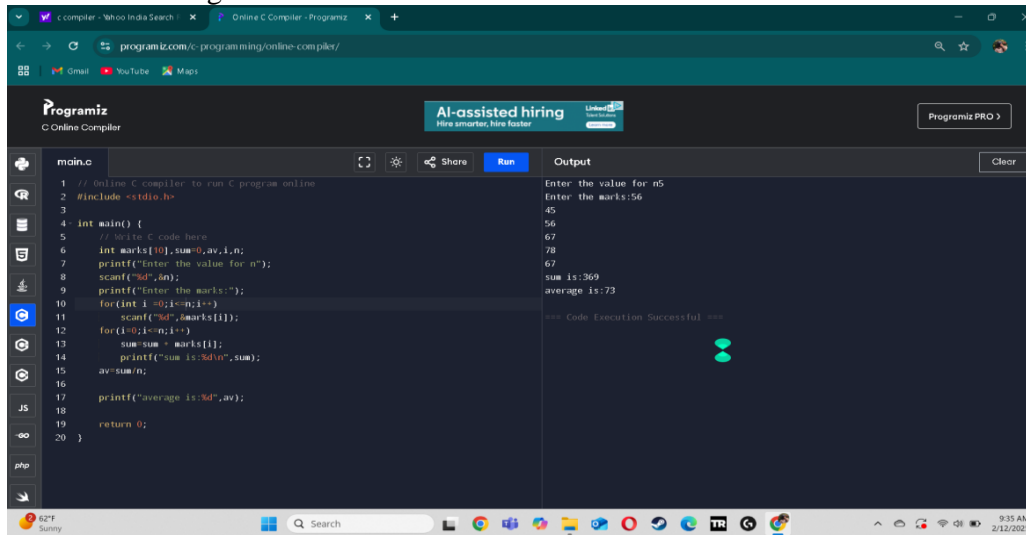


DS lab

1.sum and average



The screenshot shows the Programiz Online C Compiler interface. The code editor contains a C program that calculates the sum and average of an array of marks. The output window shows the program's execution results.

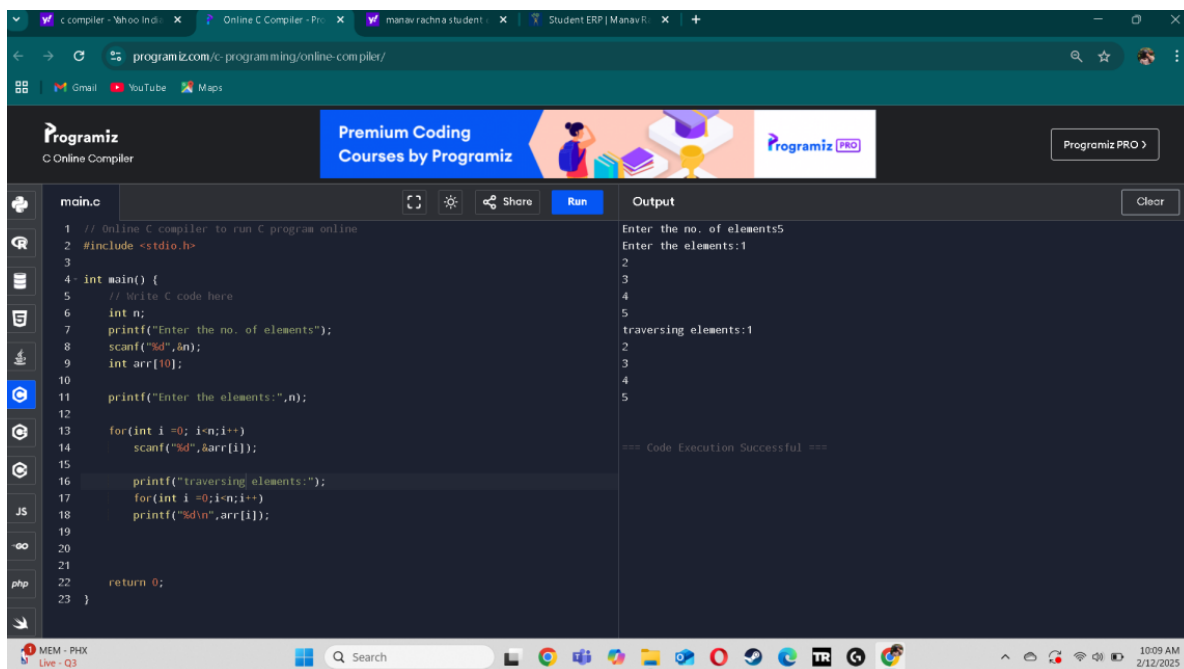
```
1 // online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     int marks[10], sum=0, av, i, n;
7     printf("Enter the value for n");
8     scanf("%d", &n);
9     printf("Enter the marks:");
10    for(int i =0; i<n; i++)
11        scanf("%d",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

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 editor contains a C program that traverses an array and prints its elements. The output window shows the program's execution results.

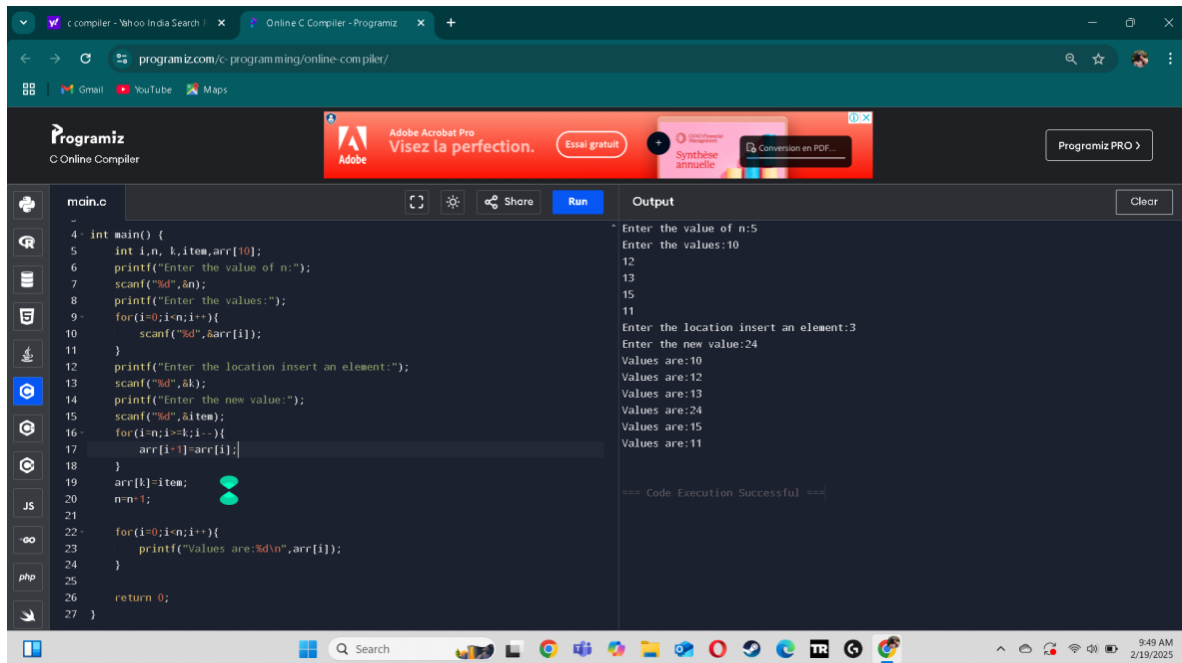
```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     int n;
7     printf("Enter the no. of elements");
8     scanf("%d", &n);
9     int arr[10];
10
11    printf("Enter the elements:", n);
12
13    for(int i =0; i<n; i++)
14        scanf("%d",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

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



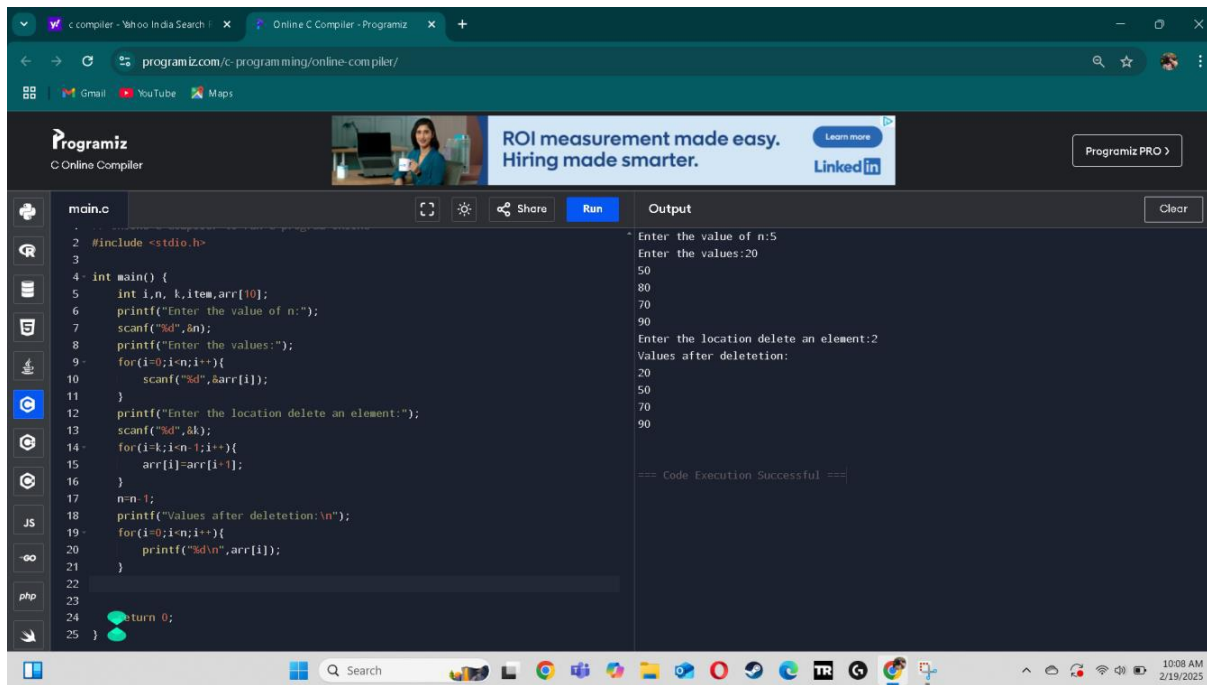
The screenshot shows the Programiz Online C Compiler interface. The code in `main.c` is as follows:

```
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 execution results:

```
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 execution results:

```
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 }
```

Output



Enter a binary number: 3
Decimal equivalent: 3

=== Code Execution Successful ===


6. 2D multiplication in array

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 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
```

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

main.c

Run

Output

```

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

```

```

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 ***

```

JS

TS

php

8. Stack operation - push and pop

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 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");

14 printf("\n2 POP");

15 printf("\n3 DISPLAY");

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 }

29 printf("\nDO YOU WANT TO CONTINUE(Y/N)");

30 while (getchar() != '\n');

31 scanf("%c", &ch);

32 while(ch=='Y' || ch=='y');

33 }

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

51 printf("\nPop element: %d", stk[top]);

52 stk[top--];

53 }

54 }

55 }

56 }

57 }

58 }

59 }

60 }

61 }

62 }

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

72 }

73 }

74 }

75 }

76 }

77 }

78 }

79 }

80 }

81 }

82 }

83 }

84 }

85 }

86 }

87 }

88 }

89 }

90 }

91 }

92 }

93 }

94 }

95 }

96 }

97 }

98 }

99 }

100 }

Output

1 PUSH

2 POP

3 DISPLAY

4 exit

press:2

stack is underflow deletion is not possible

DO YOU WANT TO CONTINUE(Y/N)y

1 PUSH

2 POP

3 DISPLAY

4 exit

press:

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

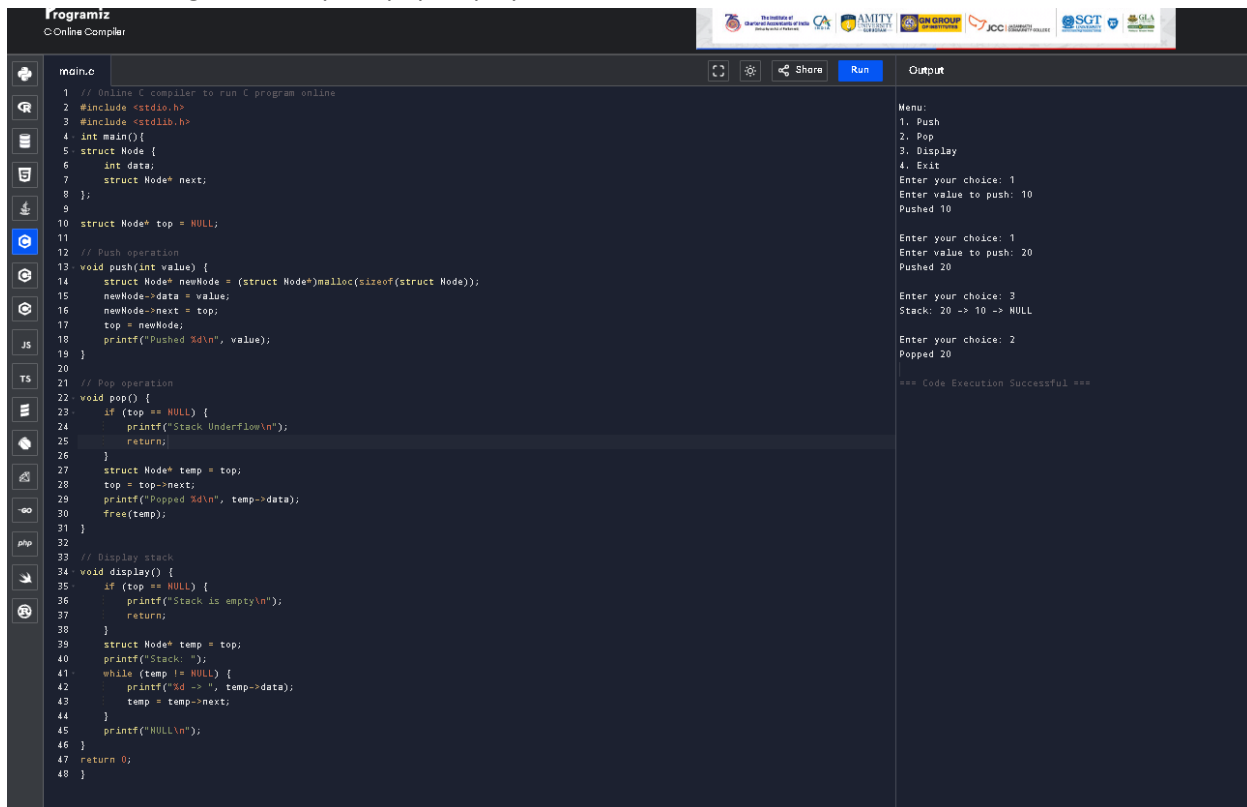
```
main.c
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 // Node structure
6 struct Node {
7     int data;
8     struct Node* next;
9 };
10
11 struct Node* head = NULL;
12
13 // Insert at the end
14 void insert(int value) {
15     struct Node* newNode = (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     }
28     printf("Inserted %d\n", value);
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: 3
3
Menu:
1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 4
```

11. Stack through linklist - push pop display



```
main.c
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

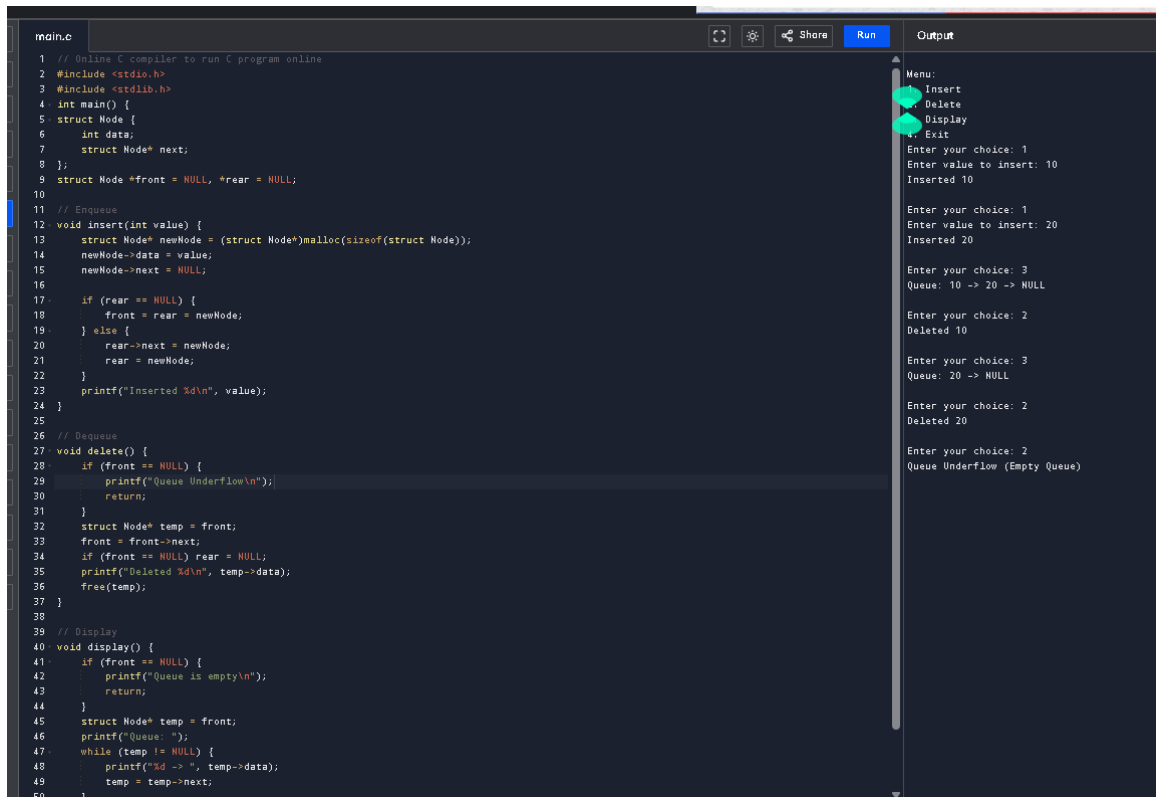
Enter your choice: 1
Enter value to push: 20
Pushed 20

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

Enter your choice: 2
Popped 20

*** Code Execution Successful ***

12. Queue through linklist- insertion deletion display



```
main.c
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     struct Node *front = NULL, *rear = NULL;
10
11    // Enqueue
12    void insert(int value) {
13        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
14        newNode->data = value;
15        newNode->next = NULL;
16
17        if (rear == NULL) {
18            front = rear = newNode;
19        } else {
20            rear->next = newNode;
21            rear = newNode;
22        }
23        printf("Inserted %d\n", value);
24    }
25
26    // Dequeue
27    void delete() {
28        if (front == NULL) {
29            printf("Queue Underflow\n");
30            return;
31        }
32        struct Node* temp = front;
33        front = front->next;
34        if (front == NULL) rear = NULL;
35        printf("Deleted %d\n", temp->data);
36        free(temp);
37    }
38
39    // Display
40    void display() {
41        if (front == NULL) {
42            printf("Queue is empty\n");
43            return;
44        }
45        struct Node* temp = front;
46        printf("Queue: ");
47        while (temp != NULL) {
48            printf("%d -> ", temp->data);
49            temp = temp->next;
50        }
51    }
```

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

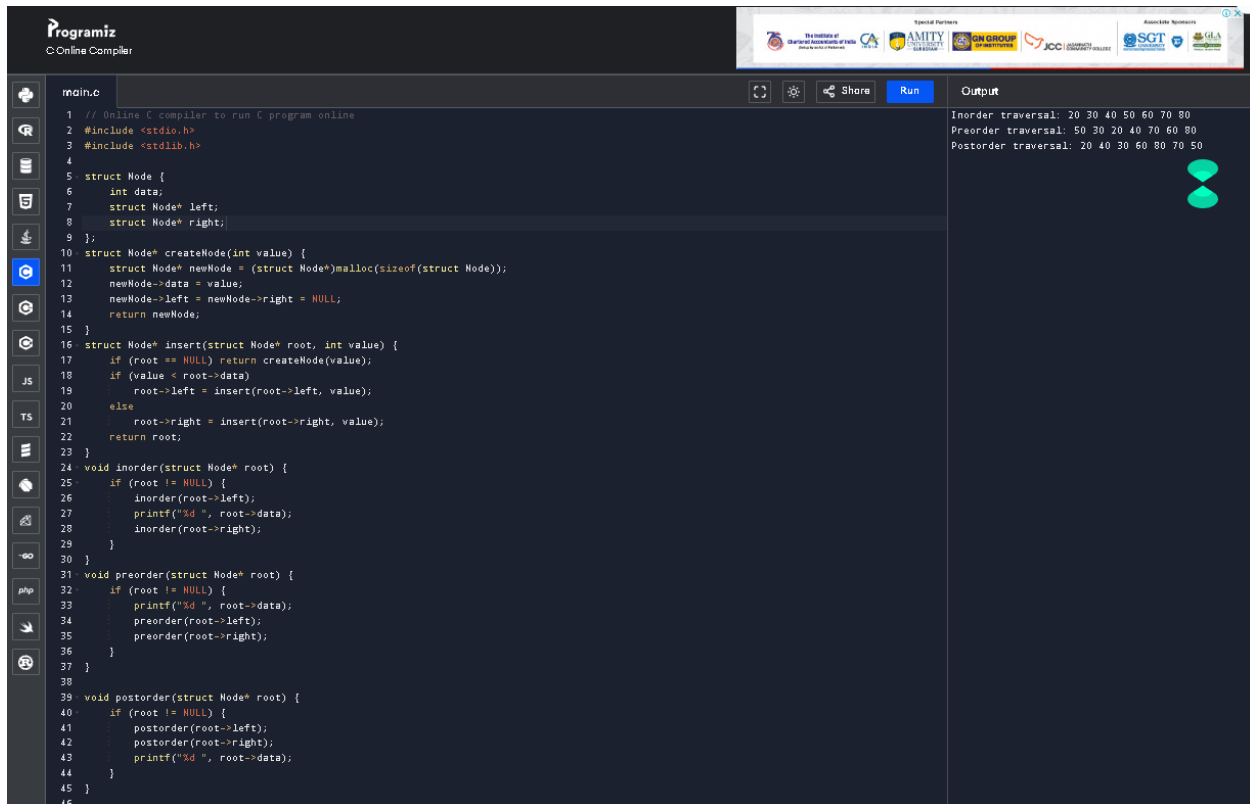
Enter your choice: 2
Deleted 10

Enter your choice: 3
Queue: 20 -> NULL

Enter your choice: 2
Deleted 20

Enter your choice: 2
Queue Underflow (Empty Queue)

13. Tree traversal

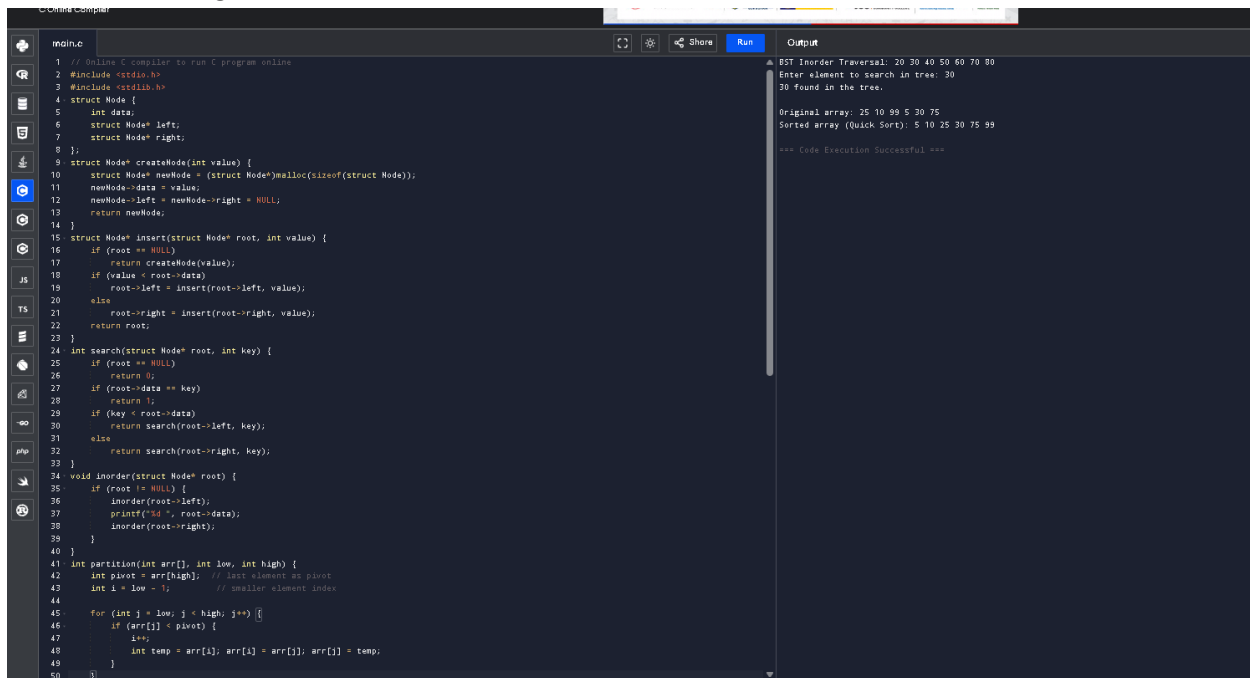


```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 struct Node {
6     int data;
7     struct Node* left;
8     struct Node* right;
9 };
10
11 struct Node* createNode(int value) {
12     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
13     newNode->data = value;
14     newNode->left = newNode->right = NULL;
15     return newNode;
16 }
17
18 struct Node* insert(struct Node* root, int value) {
19     if (root == NULL) return createNode(value);
20     if (value < root->data)
21         root->left = insert(root->left, value);
22     else
23         root->right = insert(root->right, value);
24     return root;
25 }
26
27 void inorder(struct Node* root) {
28     if (root != NULL) {
29         inorder(root->left);
30         printf("%d ", root->data);
31         inorder(root->right);
32     }
33 }
34
35 void preorder(struct Node* root) {
36     if (root != NULL) {
37         printf("%d ", root->data);
38         preorder(root->left);
39         preorder(root->right);
40     }
41 }
42
43 void postorder(struct Node* root) {
44     if (root != NULL) {
45         postorder(root->left);
46         postorder(root->right);
47         printf("%d ", root->data);
48     }
49 }
50 }
```

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



```
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
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
17 struct Node* insert(struct Node* root, int value) {
18     if (root == NULL)
19         return createNode(value);
20     if (value < root->data)
21         root->left = insert(root->left, value);
22     else
23         root->right = insert(root->right, value);
24     return root;
25 }
26
27 int search(struct Node* root, int key) {
28     if (root == NULL)
29         return 0;
30     if (root->data == key)
31         return 1;
32     if (key < root->data)
33         return search(root->left, key);
34     else
35         return search(root->right, key);
36 }
37
38 void inorder(struct Node* root) {
39     if (root != NULL) {
40         inorder(root->left);
41         printf("%d ", root->data);
42         inorder(root->right);
43     }
44 }
45
46 int partition(int arr[], int low, int high) {
47     int pivot = arr[high]; // last element as pivot
48     int i = low - 1; // smaller element index
49
50     for (int j = low; j < high; j++) {
51         if (arr[j] < pivot) {
52             i++;
53             int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
54         }
55     }
56     int temp = arr[i]; arr[i] = arr[high]; arr[high] = temp;
57     return i;
58 }
```

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 ===
```

```

    int temp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = temp;
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    // --- Tree Searching ---
    struct Node* root = NULL;
    int treeData[] = {50, 30, 70, 20, 40, 60, 80};
    int n = sizeof(treeData)/sizeof(treeData[0]);

    for (int i = 0; i < n; i++)
        root = insert(root, treeData[i]);

    printf("BST Inorder Traversal: ");
    inorder(root);
    printf("\n");

    int key;
    printf("Enter element to search in tree: ");
    scanf("%d", &key);

    if (search(root, key))
        printf("%d found in the tree.\n", key);
    else
        printf("%d not found in the tree.\n", key);

    int arr[] = {25, 10, 99, 5, 30, 75};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("\nOriginal array: ");
    for (int i = 0; i < size; i++) printf("%d ", arr[i]);

    quickSort(arr, 0, size - 1);

    printf("\nSorted array (Quick Sort): ");
    for (int i = 0; i < size; i++) printf("%d ", arr[i]);

    return 0;
}

```

15. Binary Search

method

main.c	Run	Output
<pre>1 #include <stdio.h> 2 3 int binarySearch(int arr[], int n, int key) { 4 int low = 0, high = n - 1; 5 while (low <= high) { 6 int mid = (low + high) / 2; 7 if (arr[mid] == key) 8 return mid; 9 else if (arr[mid] < key) 10 low = mid + 1; 11 else 12 high = mid - 1; 13 } 14 return -1; 15 } 16 17 int main() { 18 int arr[] = {2, 4, 6, 8, 10, 12}; 19 int n = sizeof(arr) / sizeof(arr[0]); 20 int key = 10; 21 int result = binarySearch(arr, n, key); 22 23 if (result == -1) 24 printf("Element not found\n"); 25 else 26 printf("Element found at index %d\n", result); 27 28 return 0; 29 }</pre>	Run	<pre>Element found at index 4 === Code Execution Successful ===</pre>

16. Sorting using quick sort

main.c	Run	Output
<pre>1 #include <stdio.h> 2 void quickSort(int arr[], int low, int high); 3 int partition(int arr[], int low, int high); 4 5 void quickSort(int arr[], int low, int high) { 6 if (low < high) { 7 int pivot = partition(arr, low, high); 8 quickSort(arr, low, pivot - 1); 9 quickSort(arr, pivot + 1, high); 10 } 11 } 12 13 int partition(int arr[], int low, int high) { 14 int pivot = arr[high]; 15 int i = low - 1; 16 for (int j = low; j < high; j++) { 17 if (arr[j] < pivot) { 18 i++; 19 int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp; 20 } 21 } 22 int temp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = temp; 23 return i + 1; 24 } 25 26 int main() { 27 int arr[] = {10, 7, 8, 9, 1, 5}; 28 int n = sizeof(arr) / sizeof(arr[0]); 29 30 quickSort(arr, 0, n - 1); 31 32 printf("Sorted array: "); 33 for (int i = 0; i < n; i++) 34 printf("%d ", arr[i]); 35 36 return 0; 37 }</pre>	Run	<pre>Sorted array: 1 5 7 8 9 10 === Code Execution Successful ===</pre>

17. Linear search method

main.c	Output
<pre>1 // Online C compiler to run C program online 2 #include <stdio.h> 3 4 int linearSearch(int arr[], int n, int key) { 5 for (int i = 0; i < n; i++) { 6 if (arr[i] == key) 7 return i; 8 } 9 return -1; 10 } 11 12 int main() { 13 int arr[] = {15, 23, 7, 9, 31}; 14 int n = sizeof(arr) / sizeof(arr[0]); 15 int key = 9; 16 17 int result = linearSearch(arr, n, key); 18 19 if (result == -1) 20 printf("Element not found\n"); 21 else 22 printf("Element found at index %d\n", result); 23 24 return 0; 25 }</pre>	<p>Element found at index 3</p> <p>=== Code Execution Successful ===</p>

18. Concatenation of 2 linked lists

<pre>#include <stdio.h> #include <stdlib.h> struct Node { int data; struct Node* next; }; void printList(struct Node* head) { while (head != NULL) { printf("%d -> ", head->data); head = head->next; } printf("NULL\n"); } struct Node* createNode(int data) { struct Node* newNode = (struct Node*)malloc(sizeof(struct Node)); newNode->data = data; newNode->next = NULL; return newNode; } void concatenate(struct Node* head1, struct Node* head2) { if (head1 == NULL) return; struct Node* temp = head1; while (temp->next != NULL) temp = temp->next; temp->next = head2; } int main() { struct Node* head1 = createNode(1); head1->next = createNode(2); head1->next->next = createNode(3); struct Node* head2 = createNode(4); head2->next = createNode(5); concatenate(head1, head2); printList(head1); return 0; }</pre>	<p>Concatenated List: 1 -> 2 -> 3 -> 4 -> 5 -> NULL</p> <p>=== Code Execution Successful ===</p>

19. Reverse of a linked list

```

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* next;
7 };
8 struct Node* createNode(int data) {
9     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
10    newNode->data = data;
11    newNode->next = NULL;
12    return newNode;
13 }
14 void printList(struct Node* head) {
15     while (head != NULL) {
16         printf("%d -> ", head->data);
17         head = head->next;
18     }
19     printf("NULL\n");
20 }
21 struct Node* reverse(struct Node* head) {
22     struct Node *prev = NULL, *curr = head, *next = NULL;
23     while (curr != NULL) {
24         next = curr->next;
25         curr->next = prev;
26         prev = curr;
27         curr = next;
28     }
29     return prev; // new head
30 }
31 int main() {
32     // Original list: 1 -> 2 -> 3 -> 4
33     struct Node* head = createNode(1);
34     head->next = createNode(2);
35     head->next->next = createNode(3);
36     head->next->next->next = createNode(4);
37     printf("Original List: ");
38     printList(head);
39     head = reverse(head);
40     printf("Reversed List: ");
41     printList(head);
42     return 0;
43 }

```

Original List: 1 -> 2 -> 3 -> 4 -> NU
Reversed List: 4 -> 3 -> 2 -> 1 -> NU

=== Code Execution Successful ===

20. Sorting using merge sort

```

1 #include <stdio.h>
2
3 void mergeSort(int arr[], int l, int r);
4 void merge(int arr[], int l, int m, int r);
5
6 void mergeSort(int arr[], int l, int r) {
7     if (l < r) {
8         int m = (l + r) / 2;
9         mergeSort(arr, l, m);
10        mergeSort(arr, m + 1, r);
11        merge(arr, l, m, r);
12    }
13 }
14
15 void merge(int arr[], int l, int m, int r) {
16     int n1 = m - l + 1, n2 = r - m;
17     int L[n1], R[n2];
18
19     for (int i = 0; i < n1; i++) L[i] = arr[l + i];
20     for (int j = 0; j < n2; j++) R[j] = arr[m + 1 + j];
21
22     int i = 0, j = 0, k = l;
23     while (i < n1 && j < n2)
24         arr[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];
25
26     while (i < n1) arr[k++] = L[i++];
27     while (j < n2) arr[k++] = R[j++];
28 }
29
30 int main() {
31     int arr[] = {38, 27, 43, 3, 9, 82, 10};
32     int n = sizeof(arr) / sizeof(arr[0]);
33
34     mergeSort(arr, 0, n - 1);
35
36     printf("Sorted array: ");
37     for (int i = 0; i < n; i++)
38         printf("%d ", arr[i]);
39     return 0;
40 }

```

Sorted array: 3 9 10 27 38 43 82

=== Code Execution Successful ===