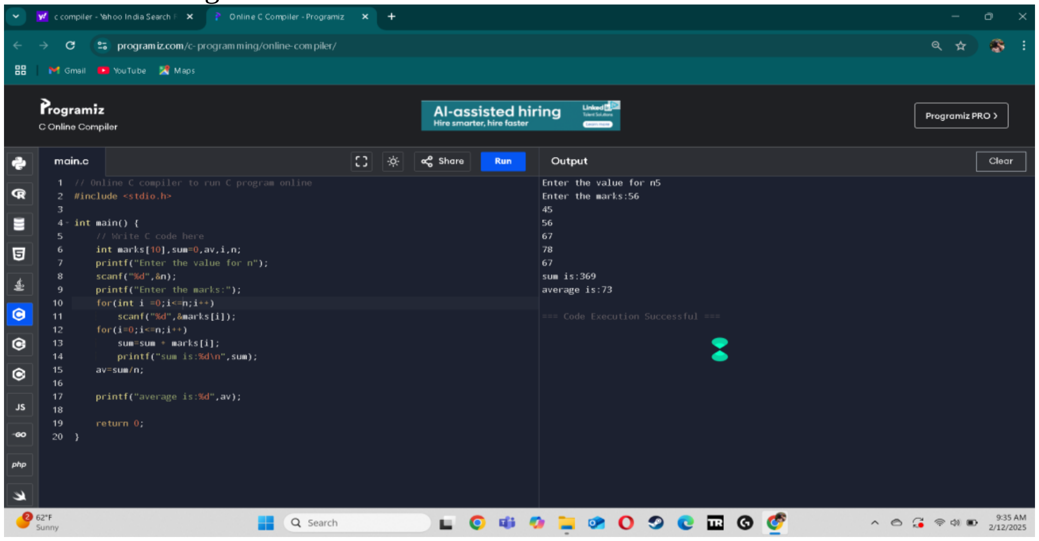
DS lab

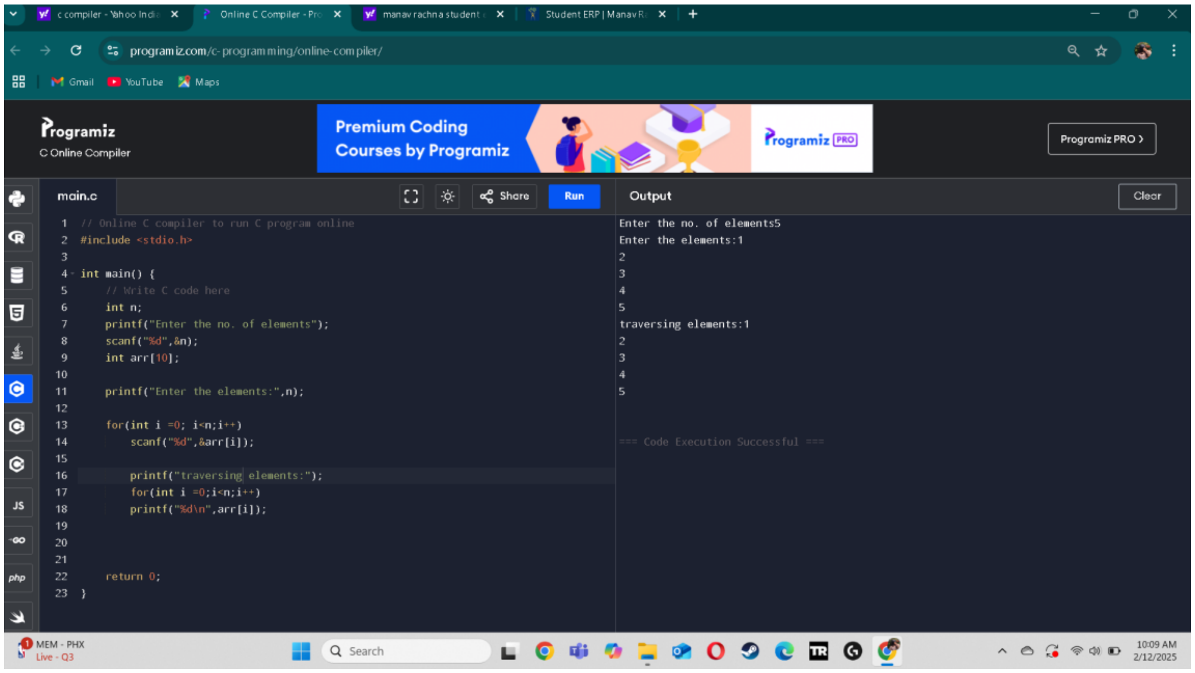
1.sum and average



```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    // Write C code here
    int marks[10],sum=0,av,i,n;
    printf("Enter the value for n");
    scanf("%d",&n);
    printf("Enter the marks:");
    for(int i =0;i<=n;i++)
        scanf("%d",&marks[i]);
    for(i=0;i<=n;i++)
        sum=sum + marks[i];
        printf("sum is:%d\n",sum);
    av=sum/n;

    printf("average is:%d",av);

    return 0;
}
```

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

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

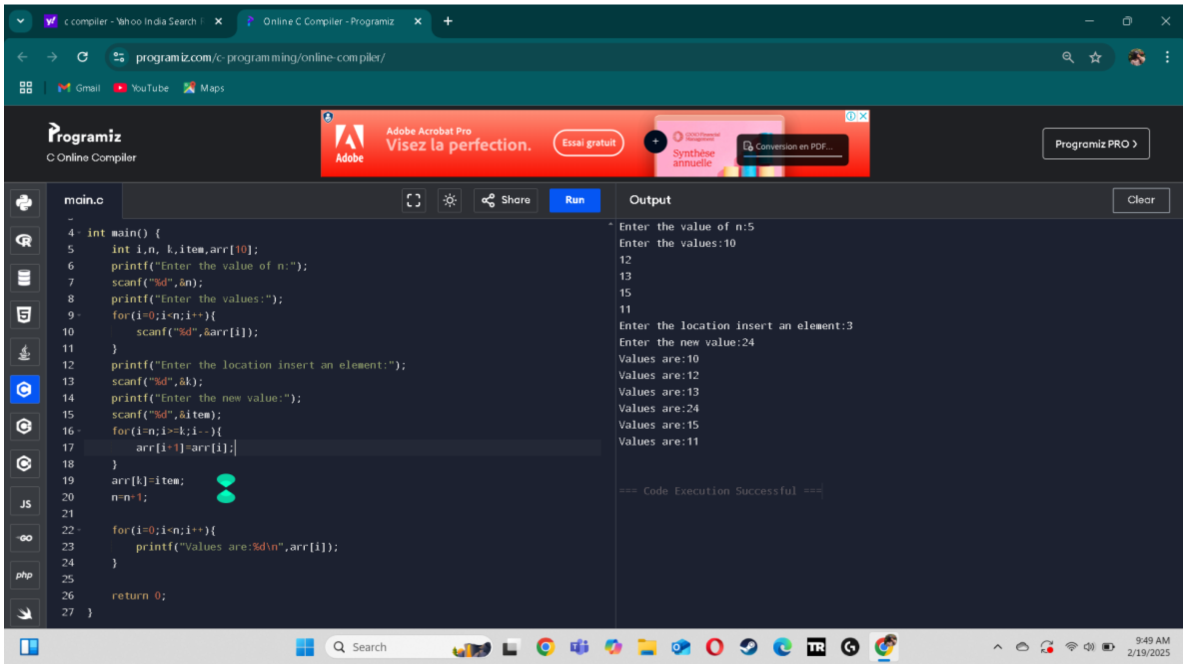
2.travesing in array



```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    // Write C code here
    int n;
    printf("Enter the no. of elements");
    scanf("%d",&n);
    int arr[10];

    printf("Enter the elements:",n);

    for(int i =0; i<n;i++)
        scanf("%d",&arr[i]);

        printf("traversing elements:");
        for(int i =0;i<n;i++)
        printf("%d\n",arr[i]);


    return 0;
}
```

```
Enter the no. of elements5
Enter the elements:1
2
3
4
5
traversing elements:1
2
3
4
5


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

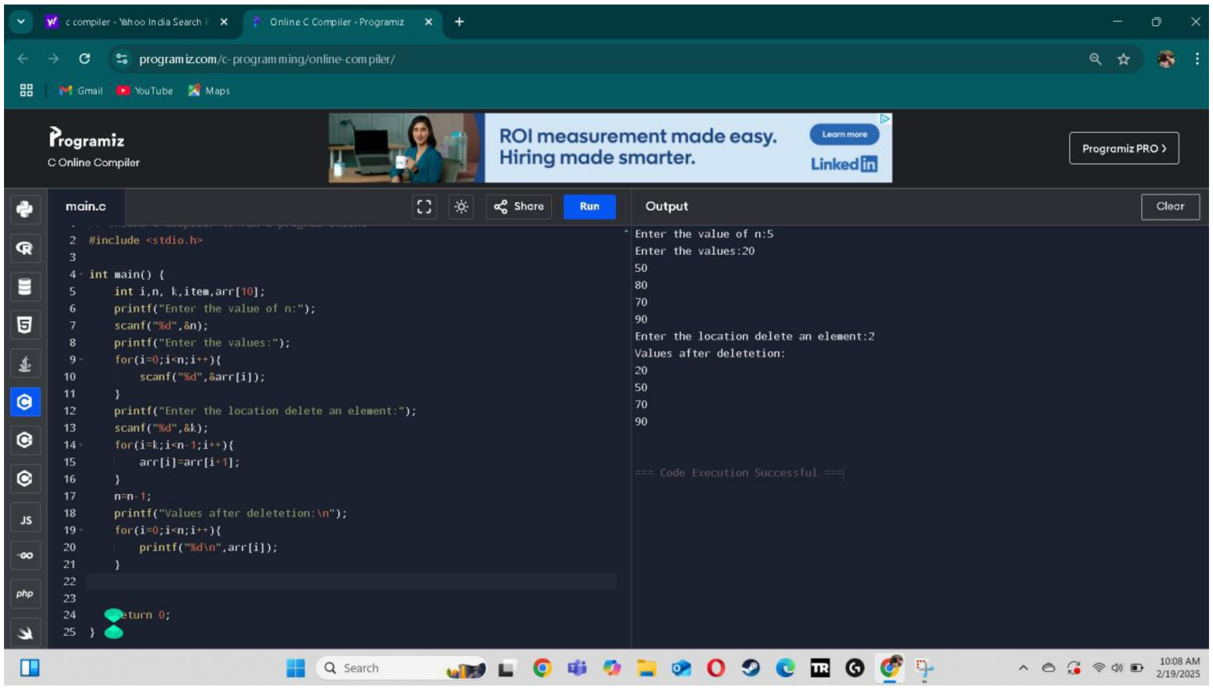
3.Insertion in array



```c
int main() {
    int i,n, k,item,arr[10];
    printf("Enter the value of n:");
    scanf("%d",&n);
    printf("Enter the values:");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    printf("Enter the location insert an element:");
    scanf("%d",&k);
    printf("Enter the new value:");
    scanf("%d",&item);
    for(i=n;i>=k;i--){
        arr[i+1]=arr[i];
    }
    arr[k]=item;
    n=n+1;

    for(i=0;i<n;i++){
        printf("Values are:%d\n",arr[i]);
    }

    return 0;
}
```

```
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 ===
```

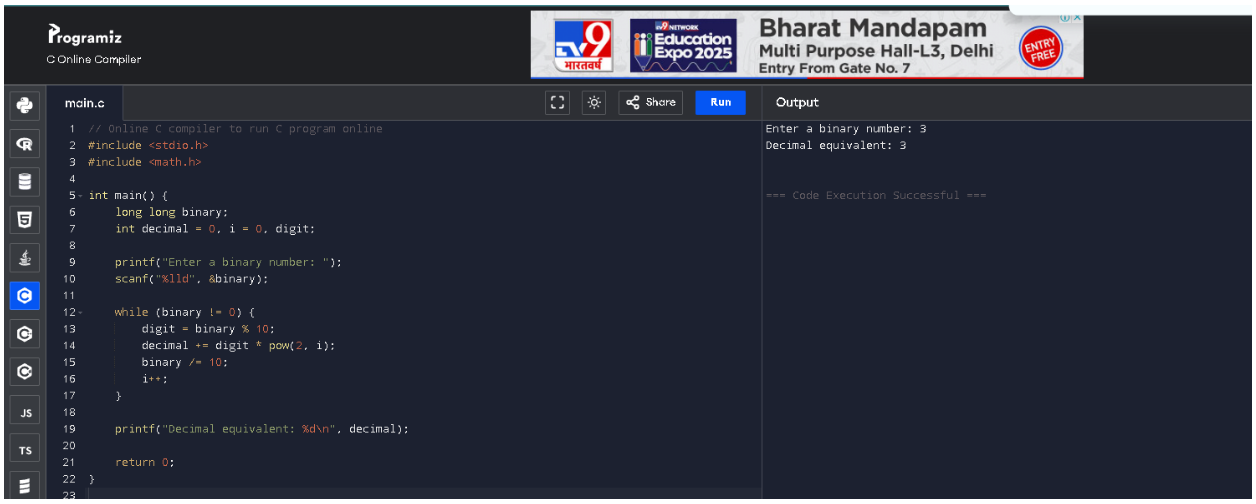
4.Deletion in array



```c
#include <stdio.h>

int main() {
    int i,n, k,item,arr[10];
    printf("Enter the value of n:");
    scanf("%d",&n);
    printf("Enter the values:");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    printf("Enter the location delete an element:");
    scanf("%d",&k);
    for(i=k;i<n-1;i++){
        arr[i]=arr[i+1];
    }
    n=n-1;
    printf("Values after deletetion:\n");
    for(i=0;i<n;i++){
        printf("%d\n",arr[i]);
    }


    return 0;
}
```

```
Enter the value of n:5
Enter the values:20
50
80
70
90
Enter the location delete an element:2
Values after deletetion:
20
50
70
90


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

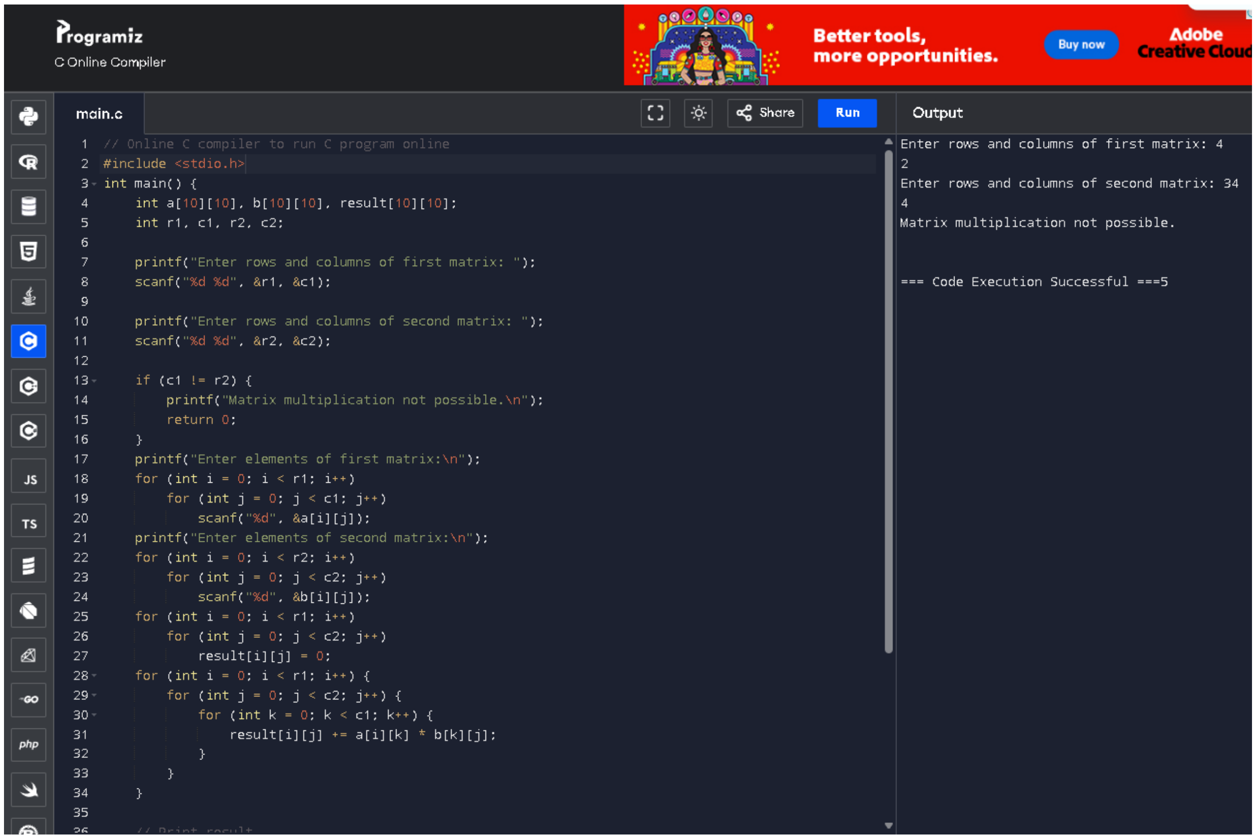
5.Binay to decimal



```c
// Online C compiler to run C program online
#include <stdio.h>
#include <math.h>

int main() {
    long long binary;
    int decimal = 0, i = 0, digit;

    printf("Enter a binary number: ");
    scanf("%lld", &binary);

    while (binary != 0) {
        digit = binary % 10;
        decimal += digit * pow(2, i);
        binary /= 10;
        i++;
    }

    printf("Decimal equivalent: %d\n", decimal);

    return 0;
}
```

```
Enter a binary number: 3
Decimal equivalent: 3


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

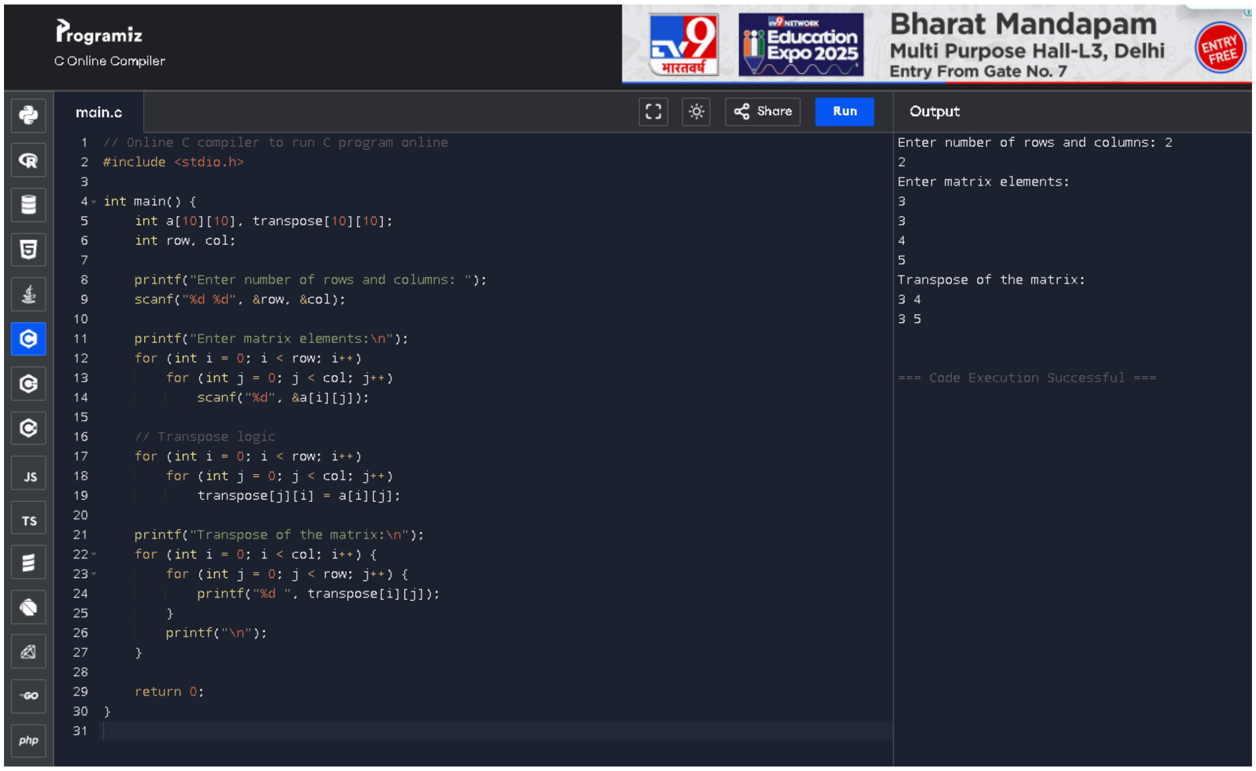
6. 2D multiplication in array



```c
// Online C compiler to run C program online
#include <stdio.h>
int main() {
    int a[10][10], b[10][10], result[10][10];
    int r1, c1, r2, c2;

    printf("Enter rows and columns of first matrix: ");
    scanf("%d %d", &r1, &c1);

    printf("Enter rows and columns of second matrix: ");
    scanf("%d %d", &r2, &c2);

    if (c1 != r2) {
        printf("Matrix multiplication not possible.\n");
        return 0;
    }
    printf("Enter elements of first matrix:\n");
    for (int i = 0; i < r1; i++)
        for (int j = 0; j < c1; j++)
            scanf("%d", &a[i][j]);
    printf("Enter elements of second matrix:\n");
    for (int i = 0; i < r2; i++)
        for (int j = 0; j < c2; j++)
            scanf("%d", &b[i][j]);
    for (int i = 0; i < r1; i++)
        for (int j = 0; j < c2; j++)
            result[i][j] = 0;
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            for (int k = 0; k < c1; k++) {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }

    // Print result
```

```
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

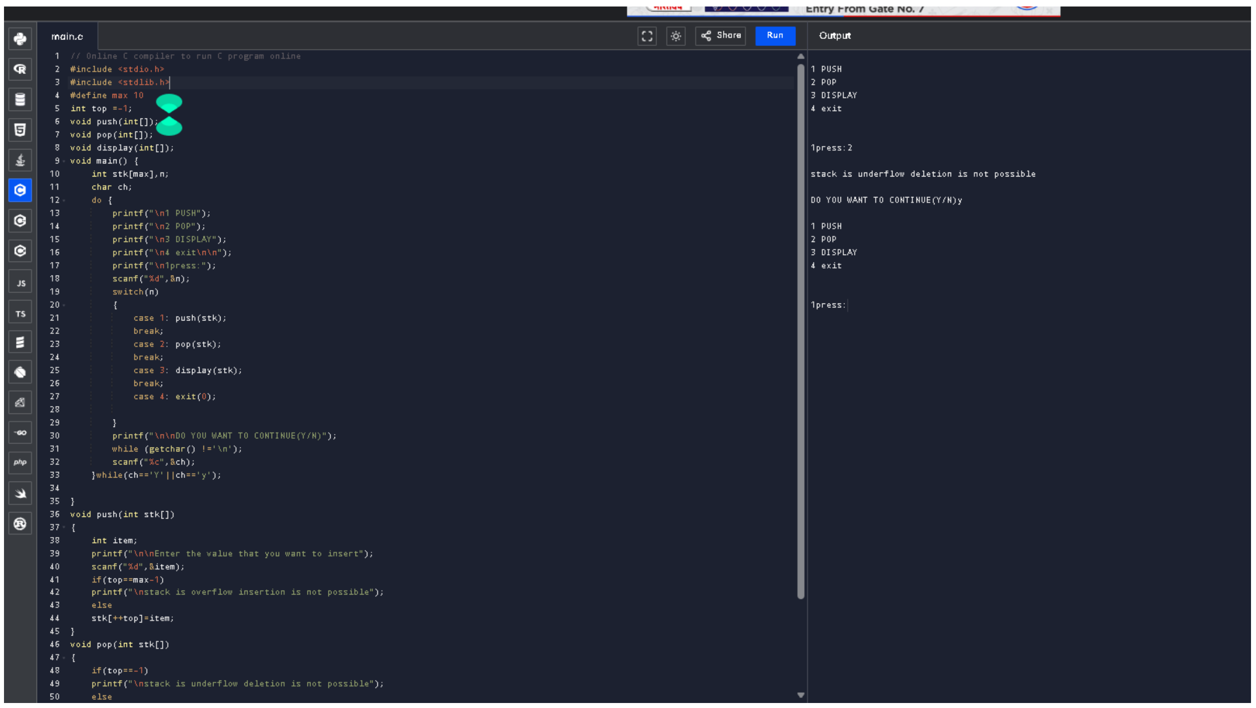
main.c

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int a[10][10], transpose[10][10];
    int row, col;

    printf("Enter number of rows and columns: ");
    scanf("%d %d", &row, &col);

    printf("Enter matrix elements:\n");
    for (int i = 0; i < row; i++)
        for (int j = 0; j < col; j++)
            scanf("%d", &a[i][j]);

    // Transpose logic
    for (int i = 0; i < row; i++)
        for (int j = 0; j < col; j++)
            transpose[j][i] = a[i][j];

    printf("Transpose of the matrix:\n");
    for (int i = 0; i < col; i++) {
        for (int j = 0; j < row; j++) {
            printf("%d ", transpose[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

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



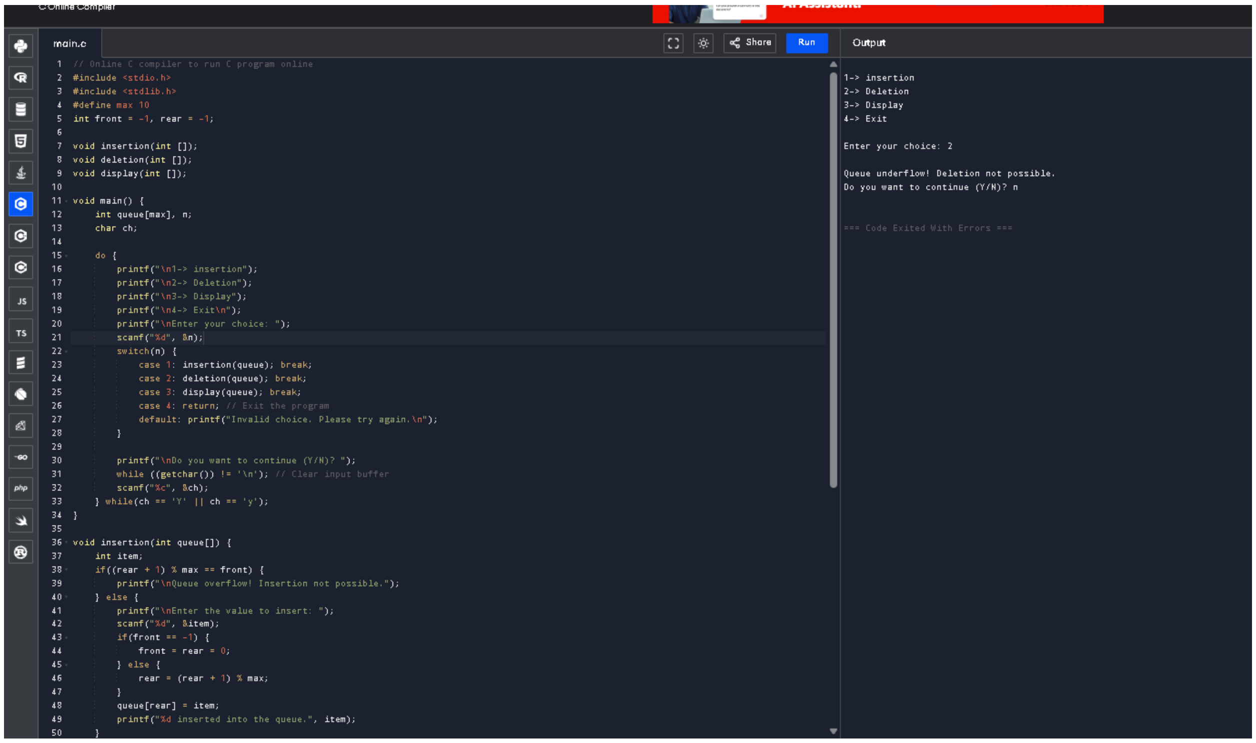
main.c

```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
#define max 10
int top =-1;
void push(int[]);
void pop(int[]);
void display(int[]);
void main() {
    int stk[max],n;
    char ch;
    do {
        printf("\n1 PUSH");
        printf("\n2 POP");
        printf("\n3 DISPLAY");
        printf("\n4 exit\n\n");
        printf("\n1press:");
        scanf("%d",&n);
        switch(n)
        {
            case 1: push(stk);
            break;
            case 2: pop(stk);
            break;
            case 3: display(stk);
            break;
            case 4: exit(0);

        }
        printf("\n\nDO YOU WANT TO CONTINUE(Y/N)");
        while (getchar() !='\n');
        scanf("%c",&ch);
    }while(ch=='Y'||ch=='y');

}
void push(int stk[])
{
    int item;
    printf("\n\nEnter the value that you want to insert");
    scanf("%d",&item);
    if(top==max-1)
    printf("\nstack is overflow insertion is not possible");
    else
    stk[++top]=item;
}
void pop(int stk[])
{
    if(top==-1)
    printf("\nstack is underflow deletion is not possible");
    else
```

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)y

1 PUSH
2 POP
3 DISPLAY
4 exit


1press:
```

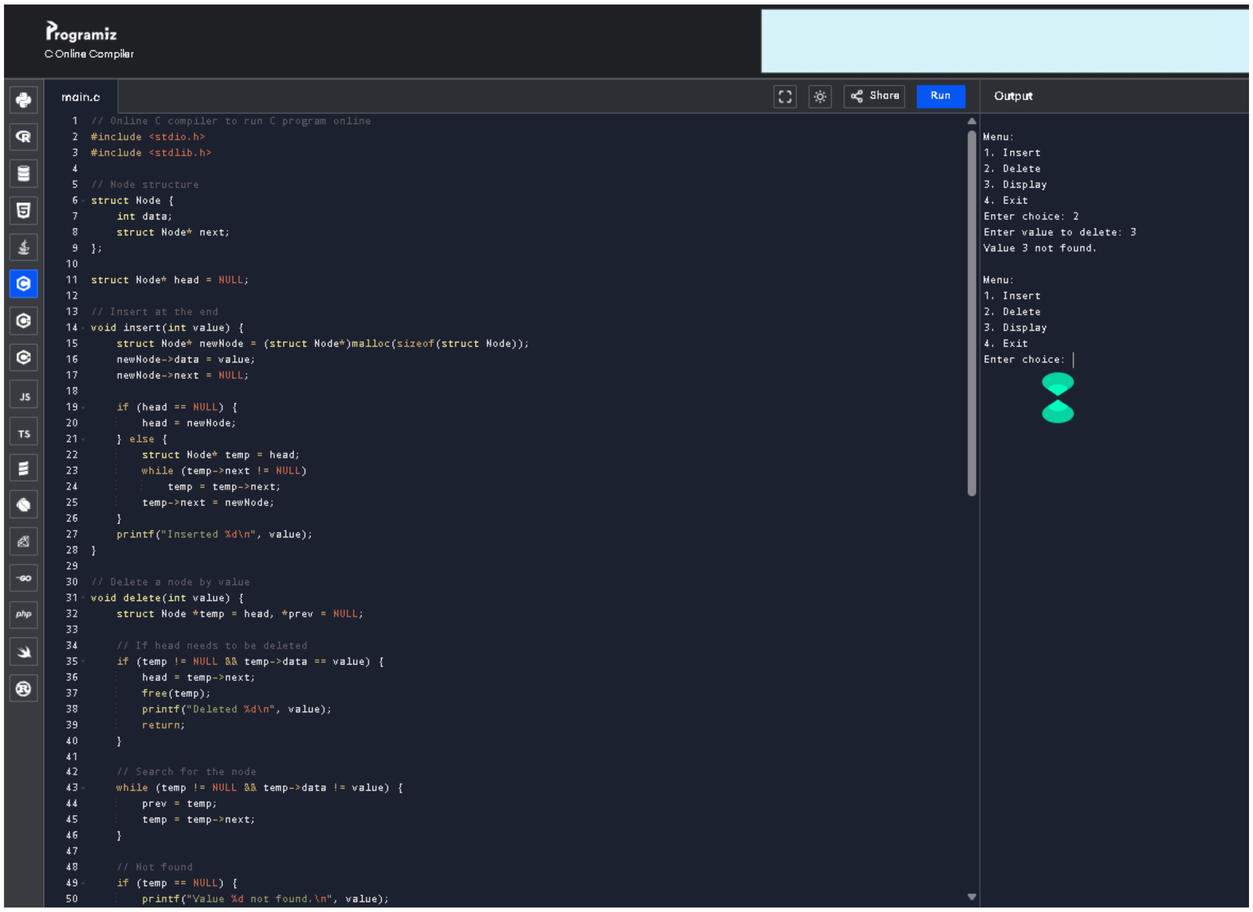
9. Queue operation - insertion deletion display



```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
#define max 10
int front = -1, rear = -1;

void insertion(int []);
void deletion(int []);
void display(int []);

void main() {
    int queue[max], n;
    char ch;

    do {
        printf("\n1-> insertion");
        printf("\n2-> Deletion");
        printf("\n3-> Display");
        printf("\n4-> Exit\n");
        printf("\nEnter your choice: ");
        scanf("%d", &n);
        switch(n) {
            case 1: insertion(queue); break;
            case 2: deletion(queue); break;
            case 3: display(queue); break;
            case 4: return; // Exit the program
            default: printf("Invalid choice. Please try again.\n");
        }

        printf("\nDo you want to continue (Y/N)? ");
        while ((getchar()) != '\n'); // Clear input buffer
        scanf("%c", &ch);
    } while(ch == 'Y' || ch == 'y');
}

void insertion(int queue[]) {
    int item;
    if((rear + 1) % max == front) {
        printf("\nQueue overflow! Insertion not possible.");
    } else {
        printf("\nEnter the value to insert: ");
        scanf("%d", &item);
        if(front == -1) {
            front = rear = 0;
        } else {
            rear = (rear + 1) % max;
        }
        queue[rear] = item;
        printf("%d inserted into the queue.", item);
    }
```

```
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 ===
```

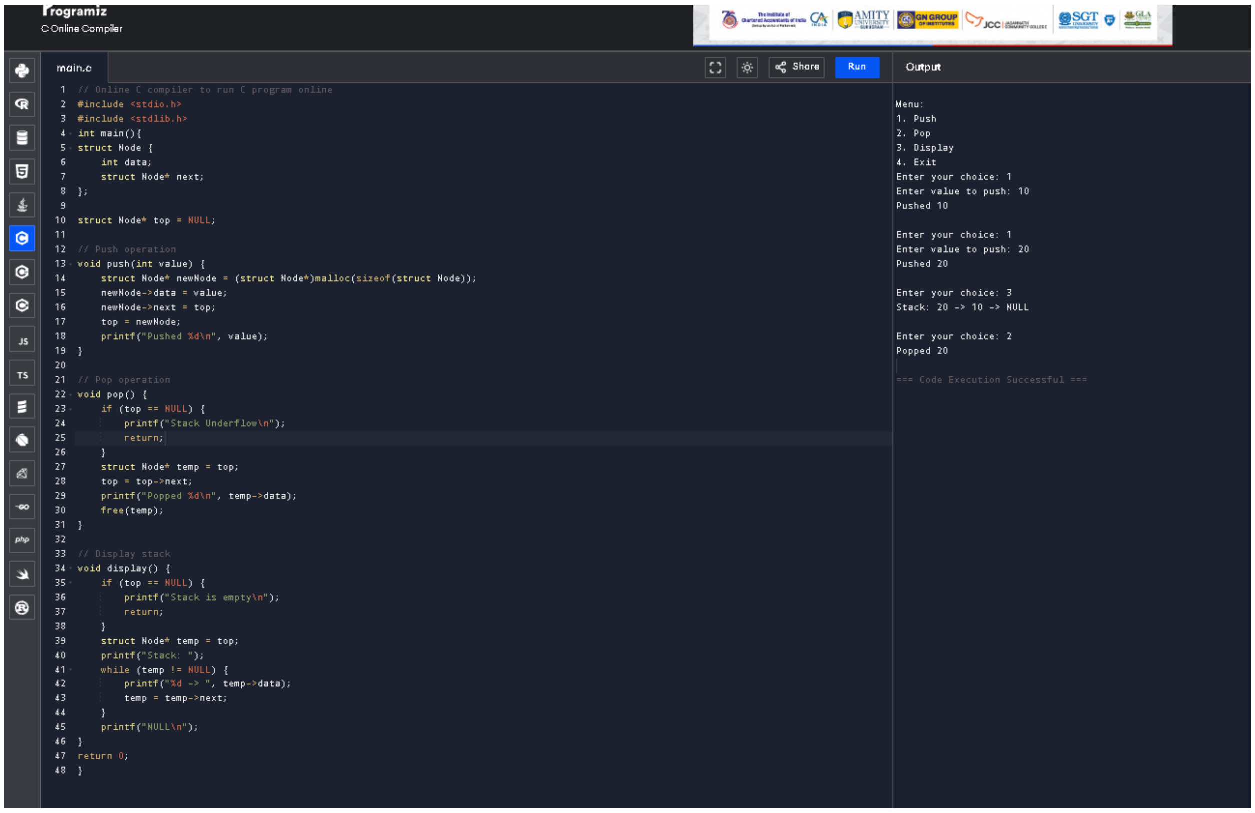
10. Linklist operation- insertion deletion display



```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>

// Node structure
struct Node {
    int data;
    struct Node* next;
};

struct Node* head = NULL;

// Insert at the end
void insert(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
    printf("Inserted %d\n", value);
}

// Delete a node by value
void delete(int value) {
    struct Node *temp = head, *prev = NULL;

    // If head needs to be deleted
    if (temp != NULL && temp->data == value) {
        head = temp->next;
        free(temp);
        printf("Deleted %d\n", value);
        return;
    }

    // Search for the node
    while (temp != NULL && temp->data != value) {
        prev = temp;
        temp = temp->next;
    }

    // Not found
    if (temp == NULL) {
        printf("Value %d not found.\n", value);
```

```
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:
```

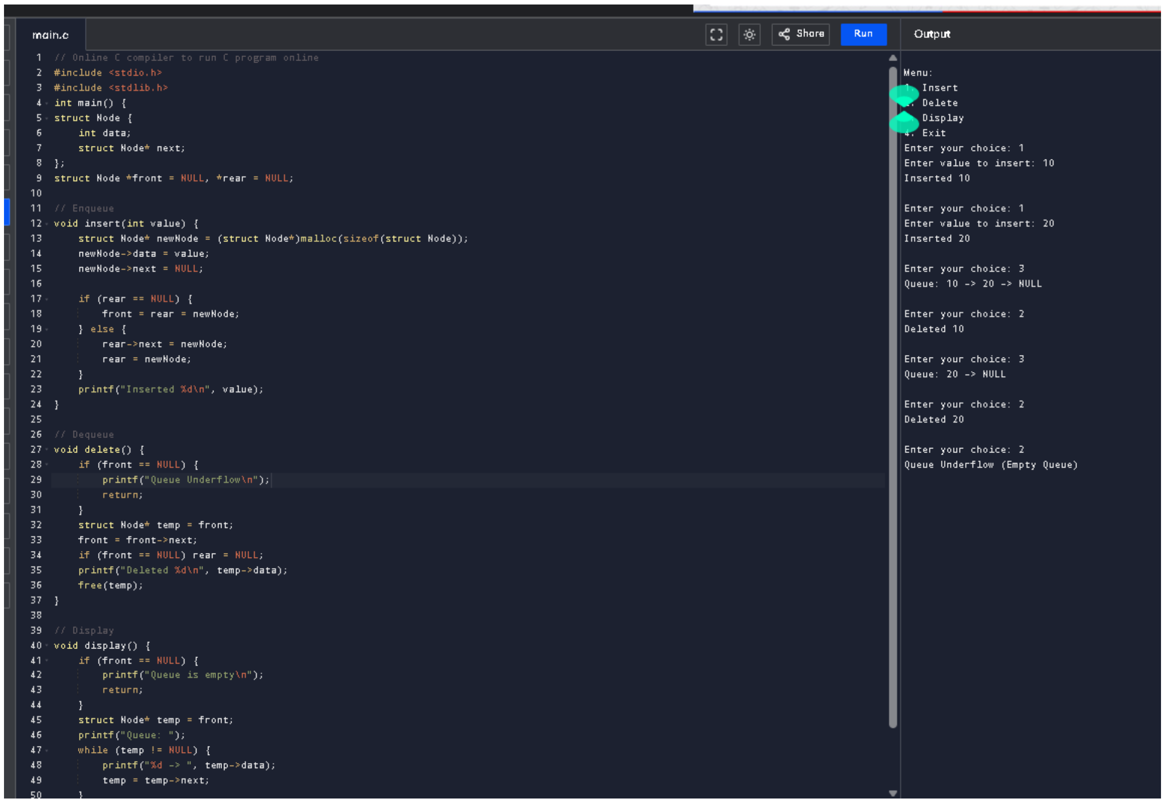
11. Stack through linklist - push pop display



```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
int main(){
struct Node {
    int data;
    struct Node* next;
};

struct Node* top = NULL;

// Push operation
void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
    printf("Pushed %d\n", value);
}

// Pop operation
void pop() {
    if (top == NULL) {
        printf("Stack Underflow\n");
        return;
    }
    struct Node* temp = top;
    top = top->next;
    printf("Popped %d\n", temp->data);
    free(temp);
}

// Display stack
void display() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    struct Node* temp = top;
    printf("Stack: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
return 0;
}
```

```
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

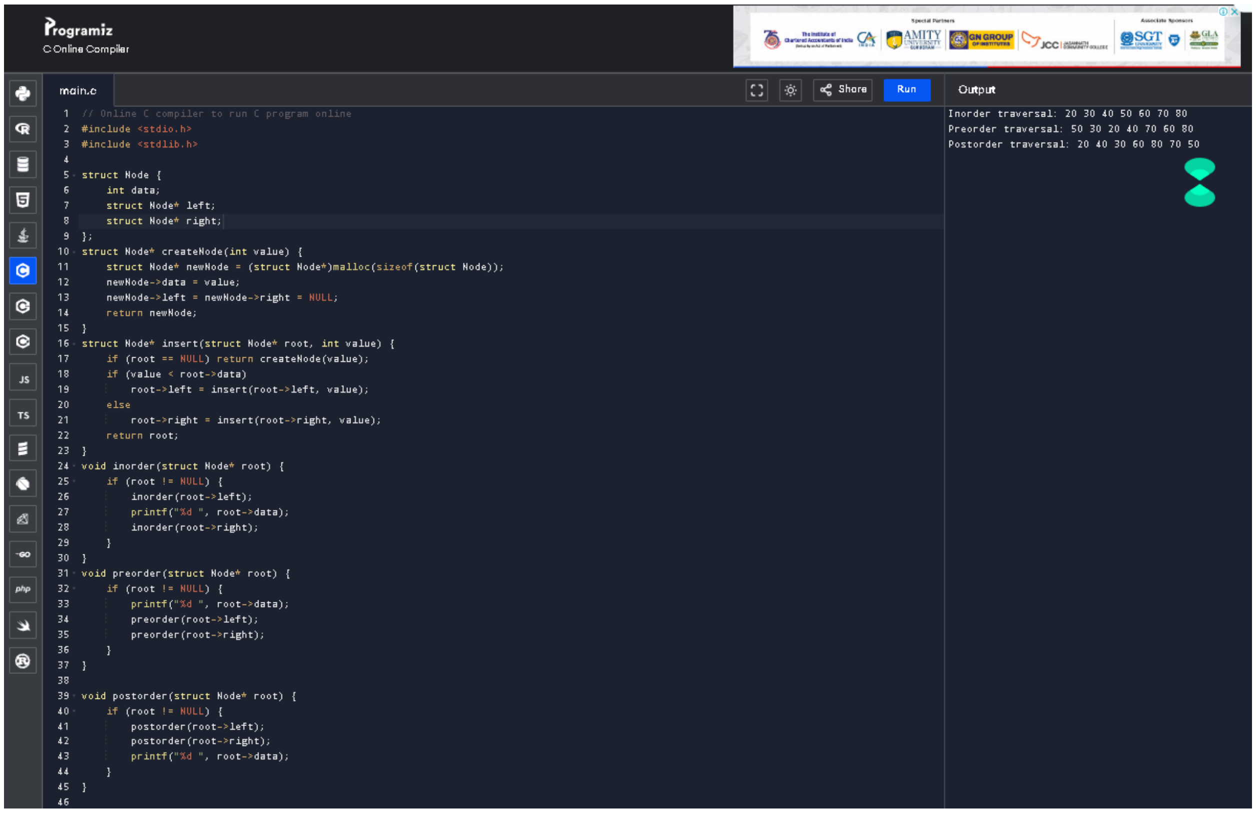


```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
int main() {
struct Node {
    int data;
    struct Node* next;
};
struct Node *front = NULL, *rear = NULL;

// Enqueue
void insert(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    printf("Inserted %d\n", value);
}

// Dequeue
void delete() {
    if (front == NULL) {
        printf("Queue Underflow\n");
        return;
    }
    struct Node* temp = front;
    front = front->next;
    if (front == NULL) rear = NULL;
    printf("Deleted %d\n", temp->data);
    free(temp);
}

// Display
void display() {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
    struct Node* temp = front;
    printf("Queue: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
```

```
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.

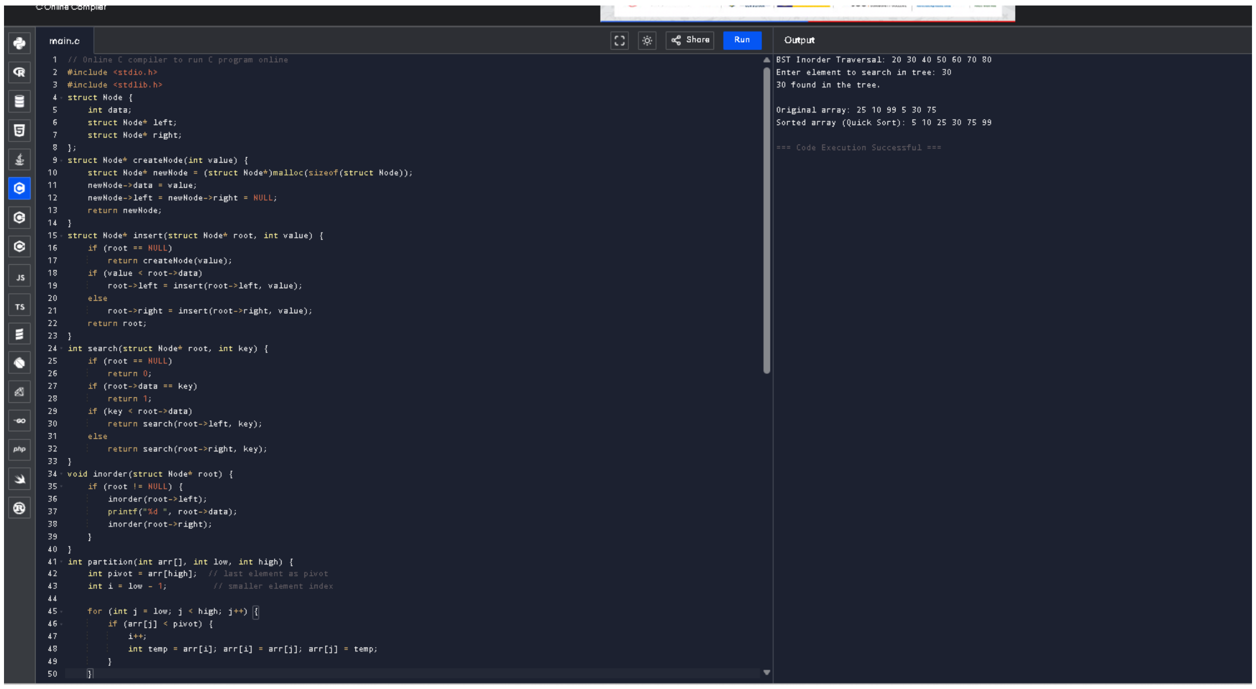
13.Tree traversal



```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}
struct Node* insert(struct Node* root, int value) {
    if (root == NULL) return createNode(value);
    if (value < root->data)
        root->left = insert(root->left, value);
    else
        root->right = insert(root->right, value);
    return root;
}
void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
void preorder(struct Node* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void postorder(struct Node* root) {
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}

```

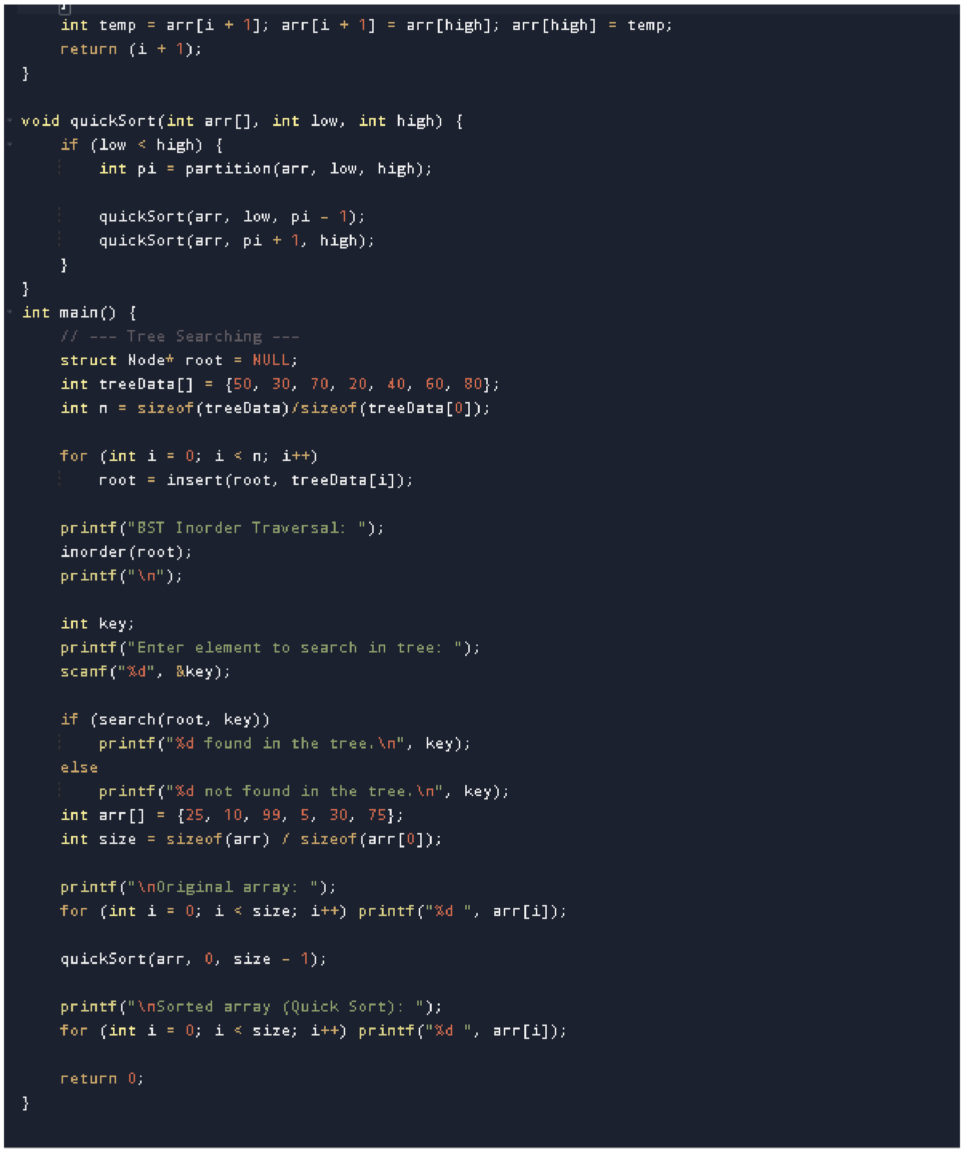
```
Inorder traversal: 20 30 40 50 60 70 80
Preorder traversal: 50 30 20 40 70 60 80
Postorder traversal: 20 40 30 60 80 70 50
```

14. Tree Searching - Quick sort



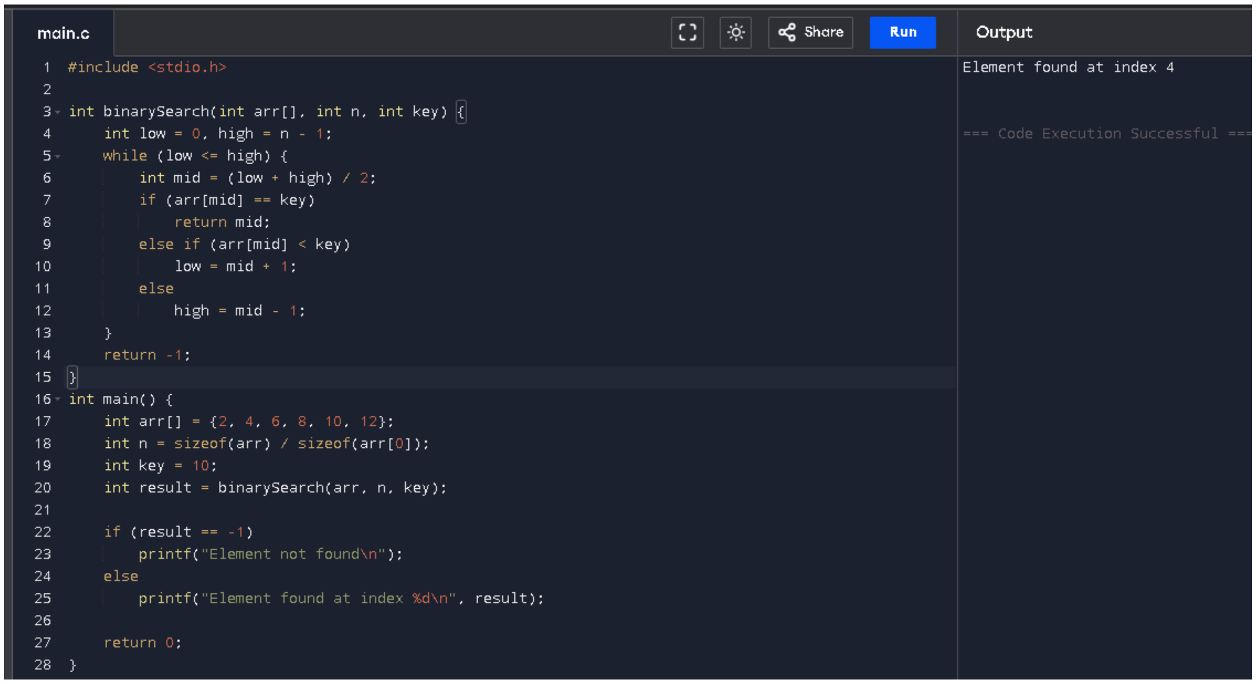
```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}
struct Node* insert(struct Node* root, int value) {
    if (root == NULL)
        return createNode(value);
    if (value < root->data)
        root->left = insert(root->left, value);
    else
        root->right = insert(root->right, value);
    return root;
}
int search(struct Node* root, int key) {
    if (root == NULL)
        return 0;
    if (root->data == key)
        return 1;
    if (key < root->data)
        return search(root->left, key);
    else
        return search(root->right, key);
}
void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];  // last element as pivot
    int i = low - 1;         // smaller element index

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
        }
    }
```

```
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 ===
```



```c
    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;
}
```

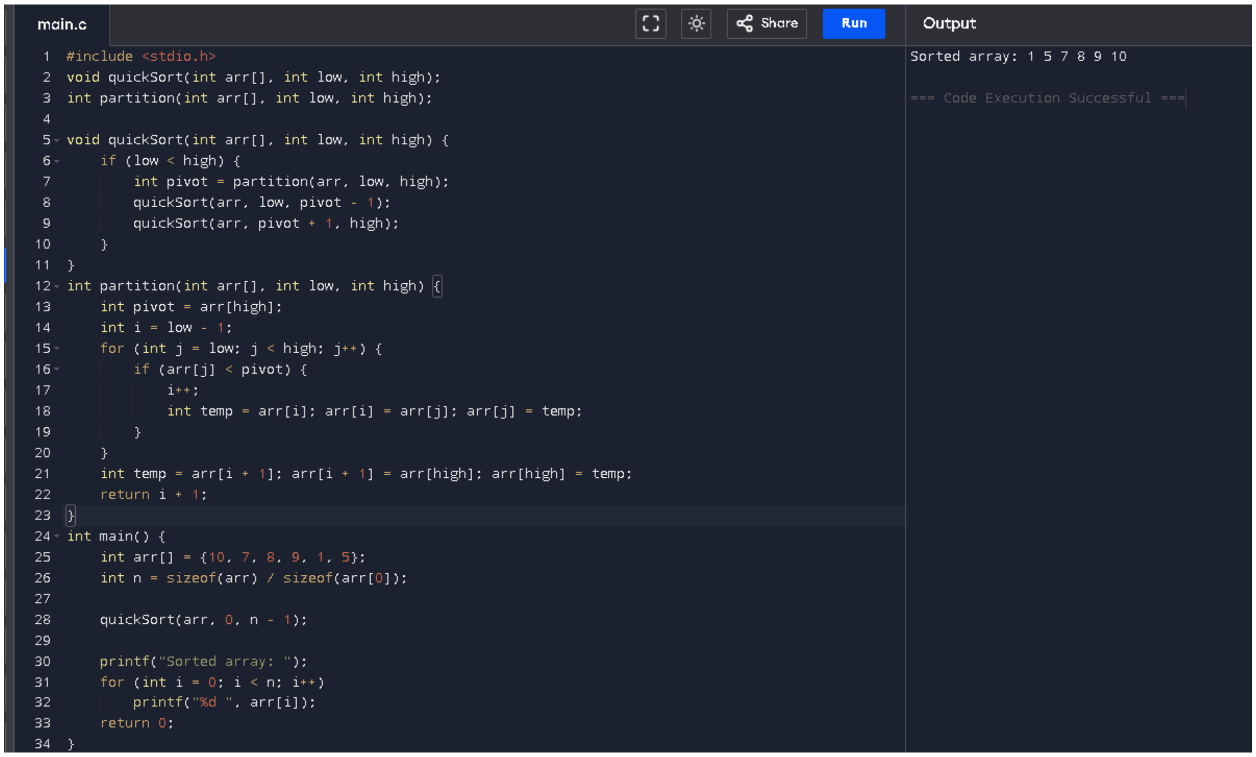
15. Binary Search method


main.c Share Run Output

```c
#include <stdio.h>

int binarySearch(int arr[], int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}
int main() {
    int arr[] = {2, 4, 6, 8, 10, 12};
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 10;
    int result = binarySearch(arr, n, key);

    if (result == -1)
        printf("Element not found\n");
    else
        printf("Element found at index %d\n", result);

    return 0;
}
```

```
Element found at index 4

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

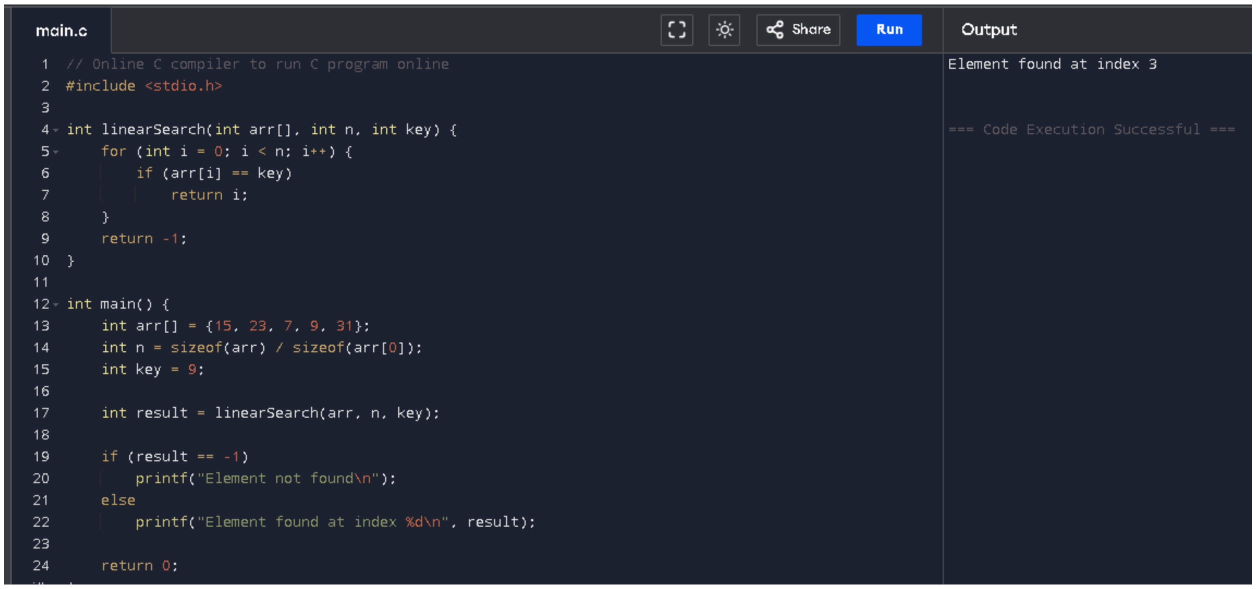
16. Sorting using quick sort


main.c Share Run Output

```c
#include <stdio.h>
void quickSort(int arr[], int low, int high);
int partition(int arr[], int low, int high);

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivot = partition(arr, low, high);
        quickSort(arr, low, pivot - 1);
        quickSort(arr, pivot + 1, high);
    }
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
        }
    }
    int temp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = temp;
    return i + 1;
}
int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);

    quickSort(arr, 0, n - 1);

    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

```
Sorted array: 1 5 7 8 9 10

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

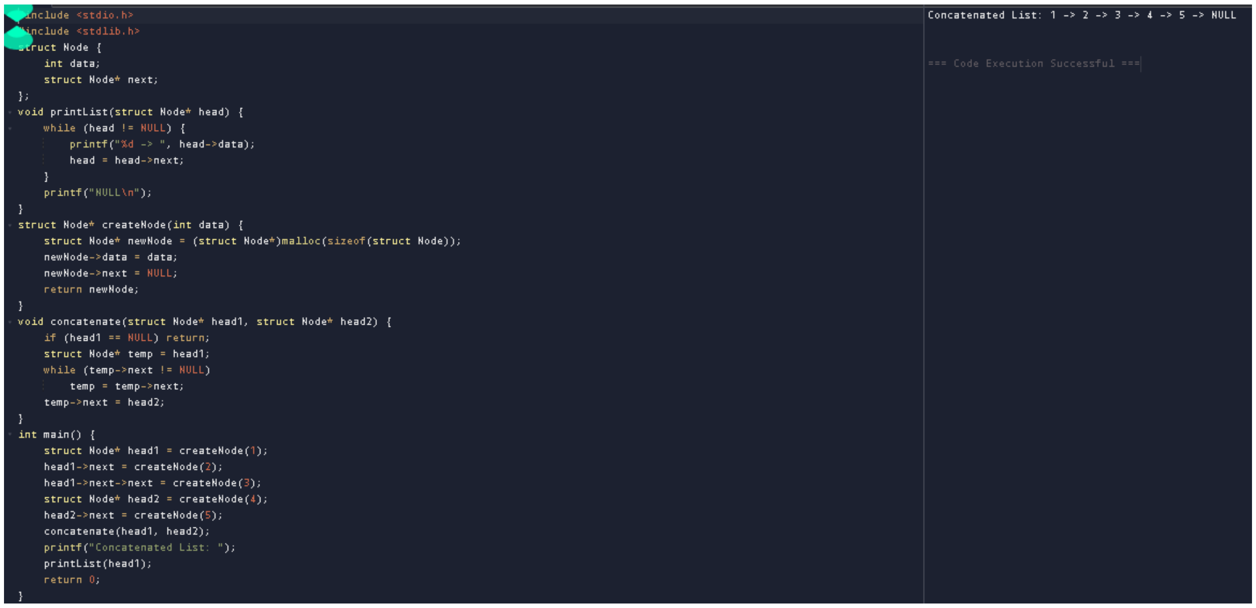
17. Linear search method



```c
// Online C compiler to run C program online
#include <stdio.h>

int linearSearch(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == key)
            return i;
    }
    return -1;
}

int main() {
    int arr[] = {15, 23, 7, 9, 31};
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 9;

    int result = linearSearch(arr, n, key);

    if (result == -1)
        printf("Element not found\n");
    else
        printf("Element found at index %d\n", result);

    return 0;
```

```
Element found at index 3


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

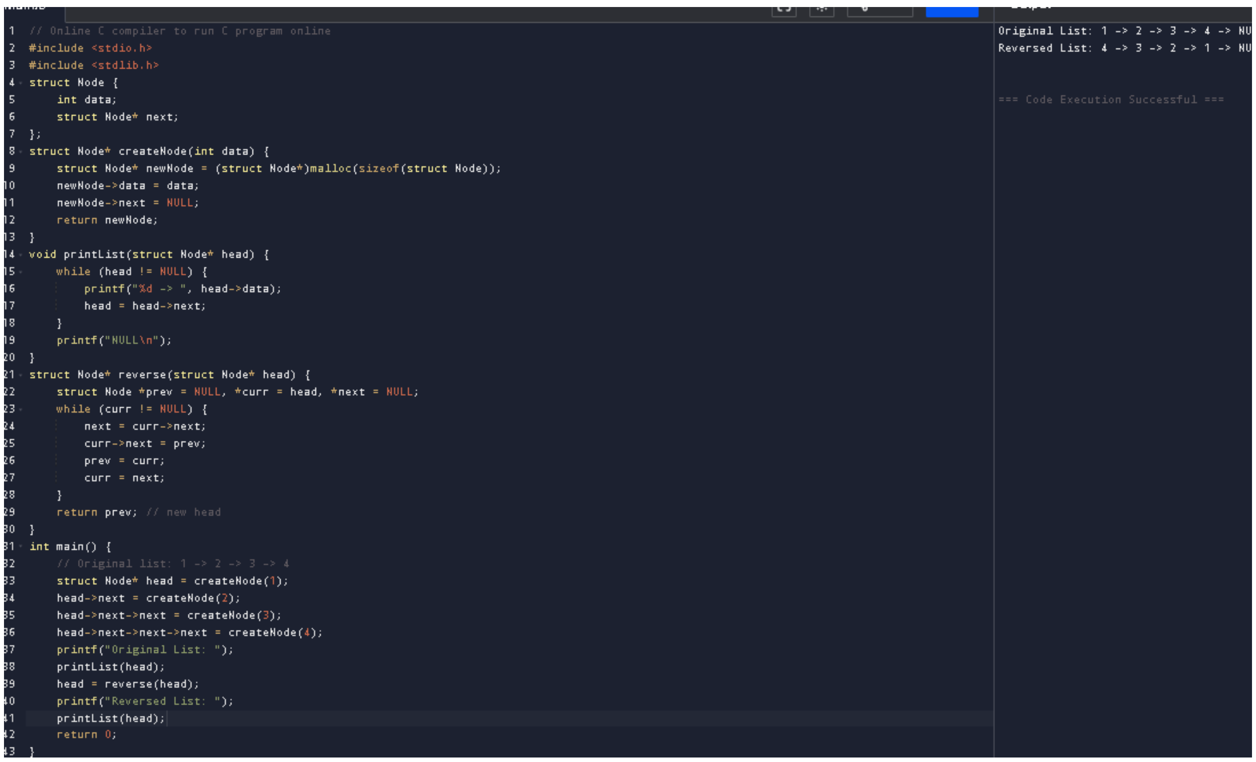
18. Concatenation of 2 linked lists



```c
#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);
    printf("Concatenated List: ");
    printList(head1);
    return 0;
}
```

```
Concatenated List: 1 -> 2 -> 3 -> 4 -> 5 -> NULL

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

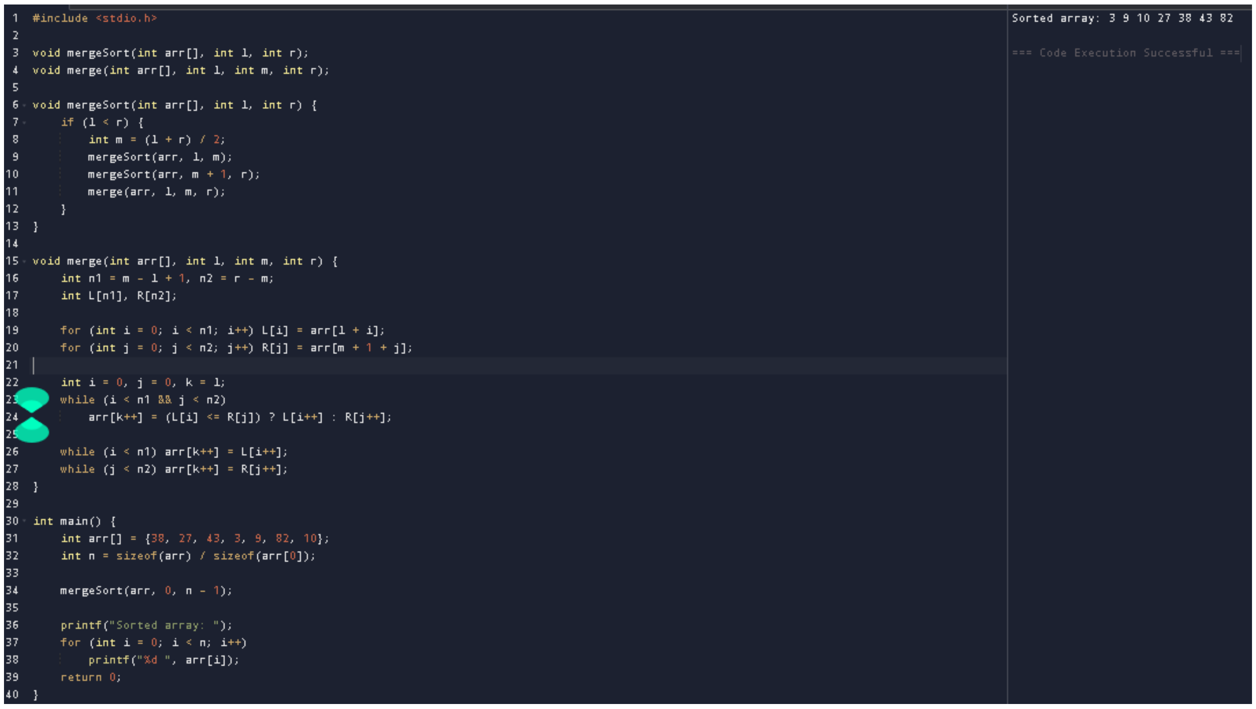
19. Reverse of a linked list



```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void printList(struct Node* head) {
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}
struct Node* reverse(struct Node* head) {
    struct Node *prev = NULL, *curr = head, *next = NULL;
    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev; // new head
}
int main() {
    // Original list: 1 -> 2 -> 3 -> 4
    struct Node* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);
    head->next->next->next = createNode(4);
    printf("Original List: ");
    printList(head);
    head = reverse(head);
    printf("Reversed List: ");
    printList(head);
    return 0;
}
```

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


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

20. Sorting using merge sort



```c
#include <stdio.h>

void mergeSort(int arr[], int l, int r);
void merge(int arr[], int l, int m, int r);

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = (l + r) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1, n2 = r - m;
    int L[n1], R[n2];

    for (int i = 0; i < n1; i++) L[i] = arr[l + i];
    for (int j = 0; j < n2; j++) R[j] = arr[m + 1 + j];

    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2)
        arr[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];

    while (i < n1) arr[k++] = L[i++];
    while (j < n2) arr[k++] = R[j++];
}

int main() {
    int arr[] = {38, 27, 43, 3, 9, 82, 10};
    int n = sizeof(arr) / sizeof(arr[0]);

    mergeSort(arr, 0, n - 1);

    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

```
Sorted array: 3 9 10 27 38 43 82

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