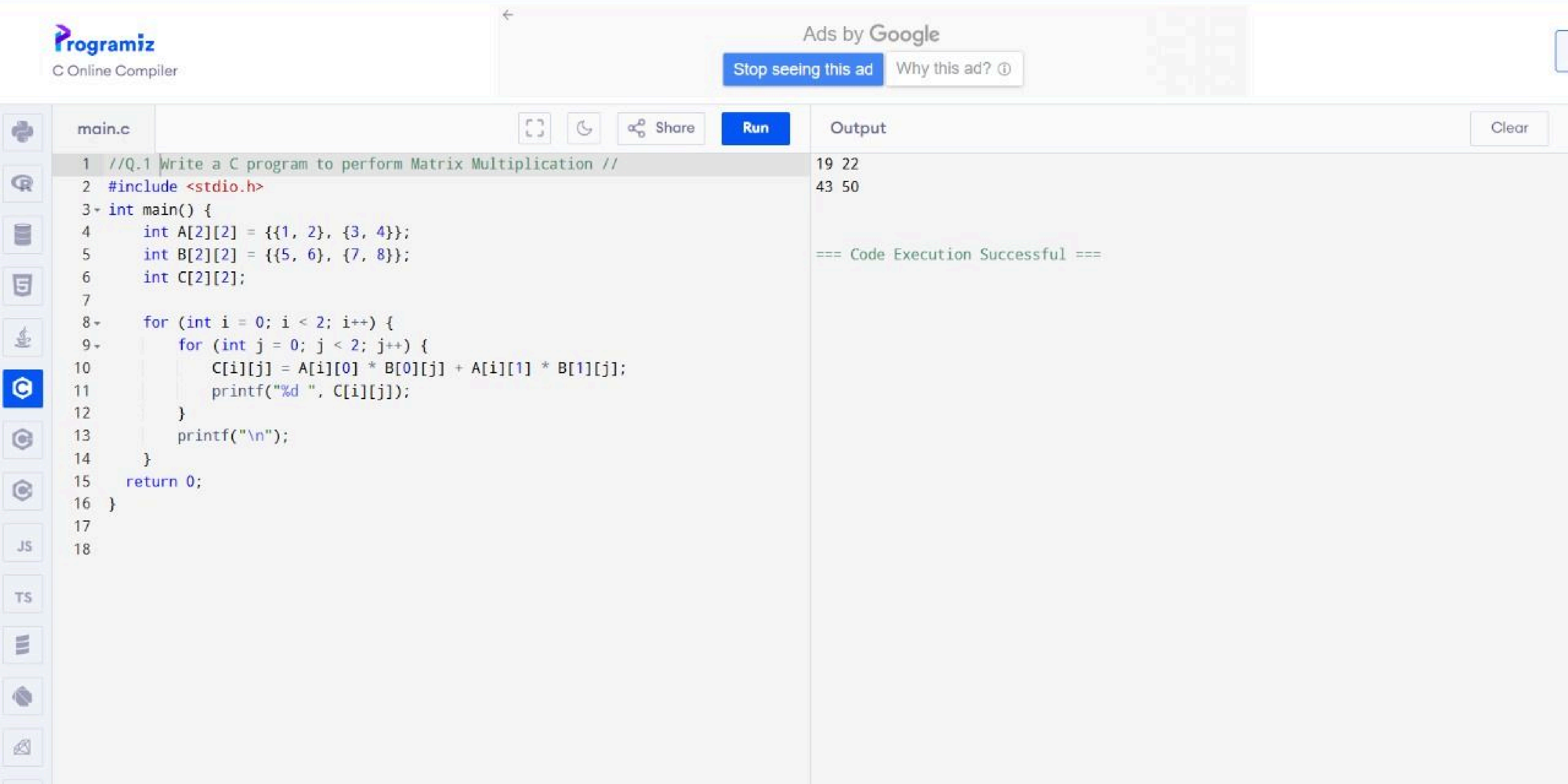
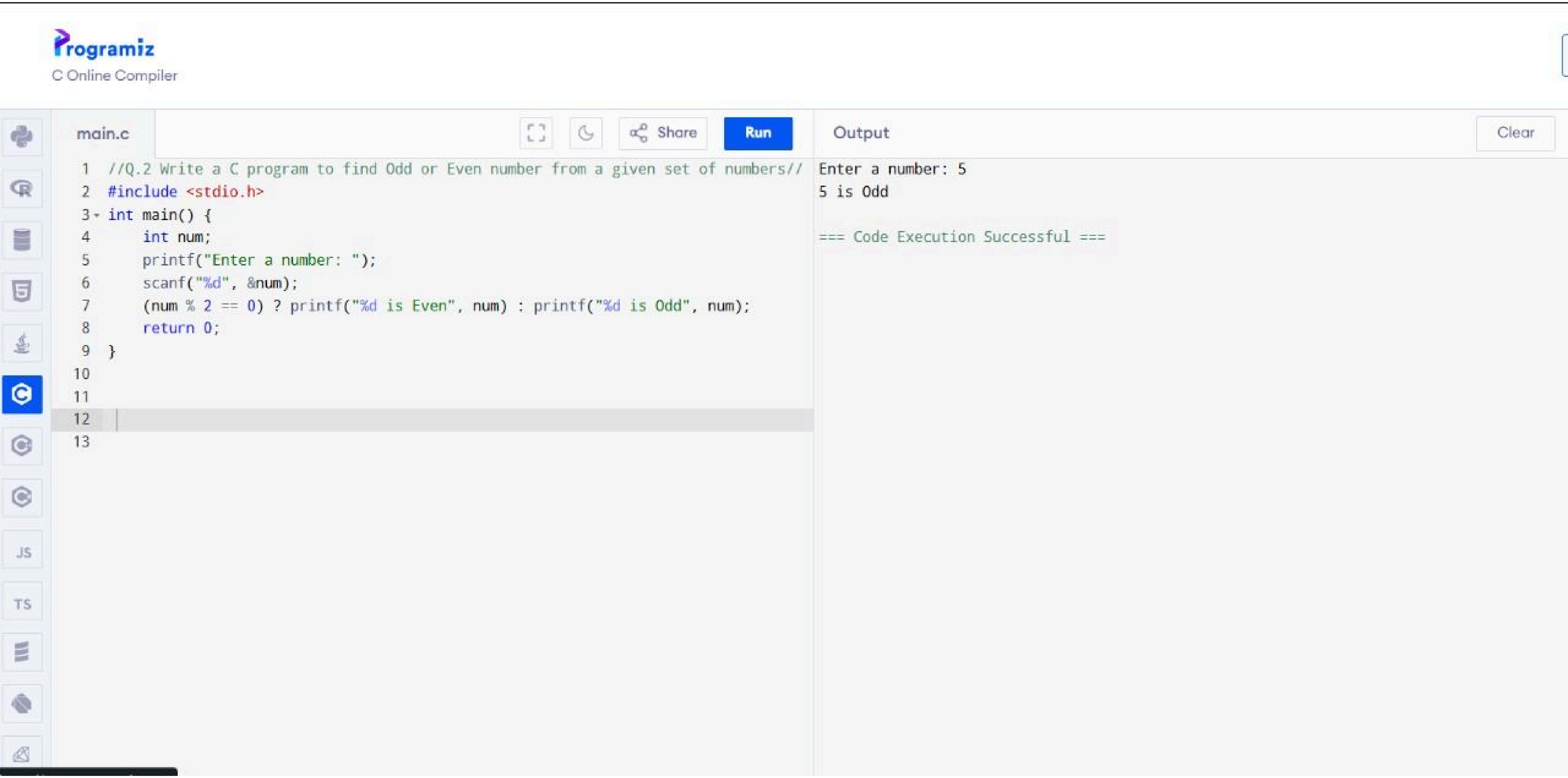


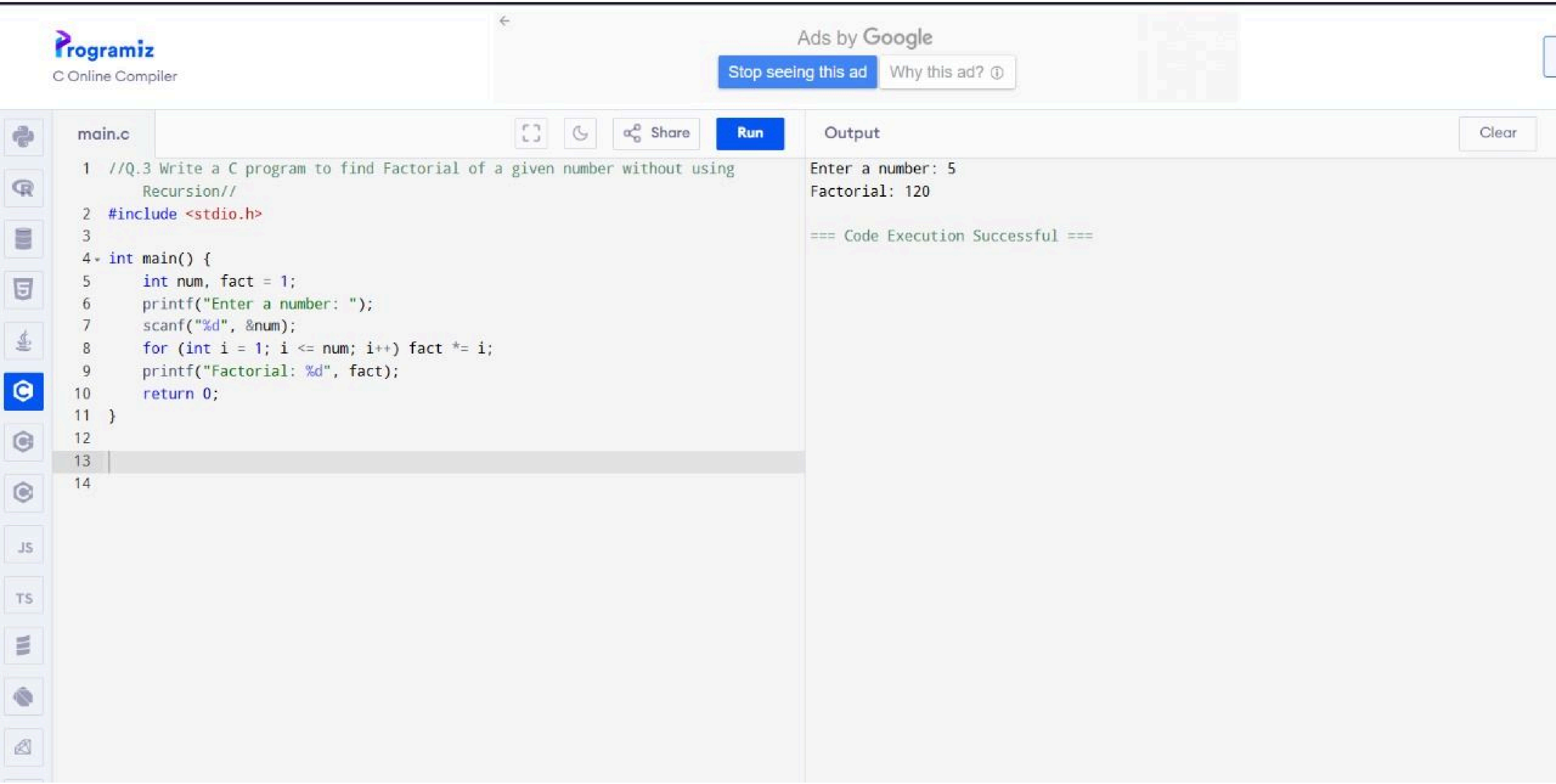
```c
//Q.1 Write a C program to perform Matrix Multiplication //
#include <stdio.h>
int main() {
    int A[2][2] = {{1, 2}, {3, 4}};
    int B[2][2] = {{5, 6}, {7, 8}};
    int C[2][2];

    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            C[i][j] = A[i][0] * B[0][j] + A[i][1] * B[1][j];
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
  return 0;
}


```

```
19 22
43 50


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



```c
//Q.2 Write a C program to find Odd or Even number from a given set of numbers//
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    (num % 2 == 0) ? printf("%d is Even", num) : printf("%d is Odd", num);
    return 0;
}
```

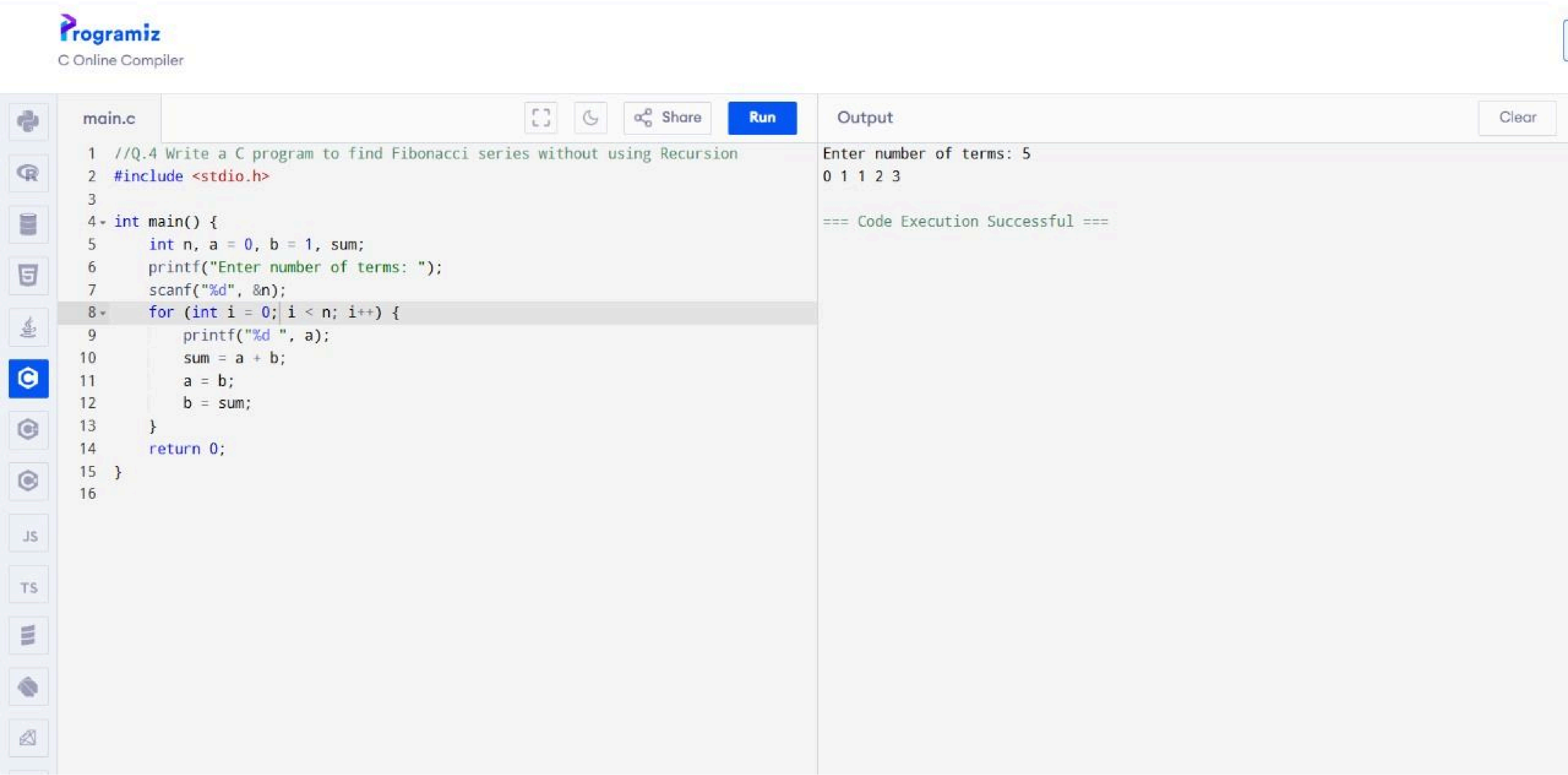
```
Enter a number: 5
5 is Odd

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



Ads by Google
Stop seeing this ad Why this ad? ⓘ

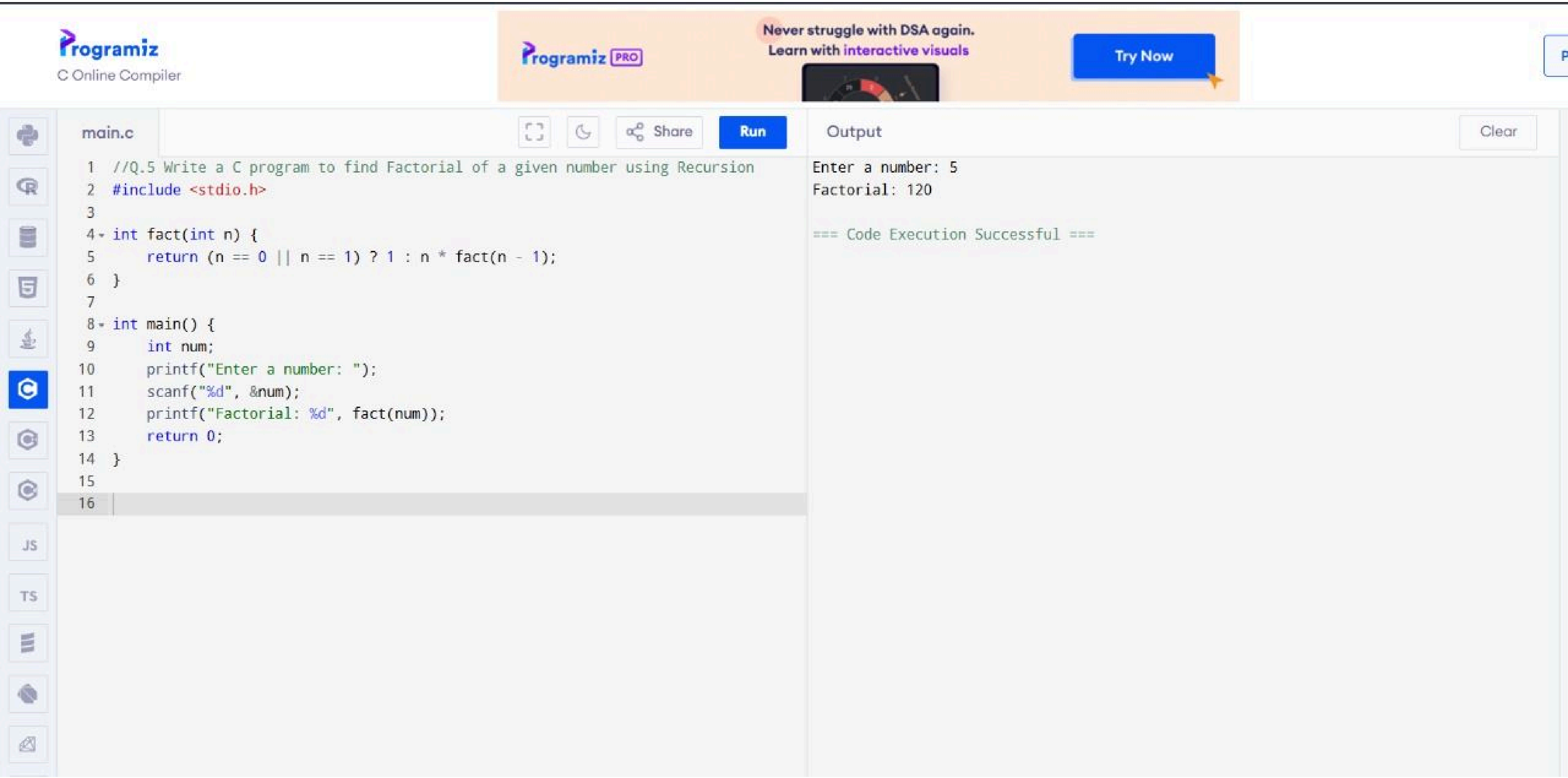
```c
//Q.3 Write a C program to find Factorial of a given number without using
    Recursion//
#include <stdio.h>

int main() {
    int num, fact = 1;
    printf("Enter a number: ");
    scanf("%d", &num);
    for (int i = 1; i <= num; i++) fact *= i;
    printf("Factorial: %d", fact);
    return 0;
}


```

```
Enter a number: 5
Factorial: 120

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



```c
//Q.4 Write a C program to find Fibonacci series without using Recursion
#include <stdio.h>

int main() {
    int n, a = 0, b = 1, sum;
    printf("Enter number of terms: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        printf("%d ", a);
        sum = a + b;
        a = b;
        b = sum;
    }
    return 0;
}
```

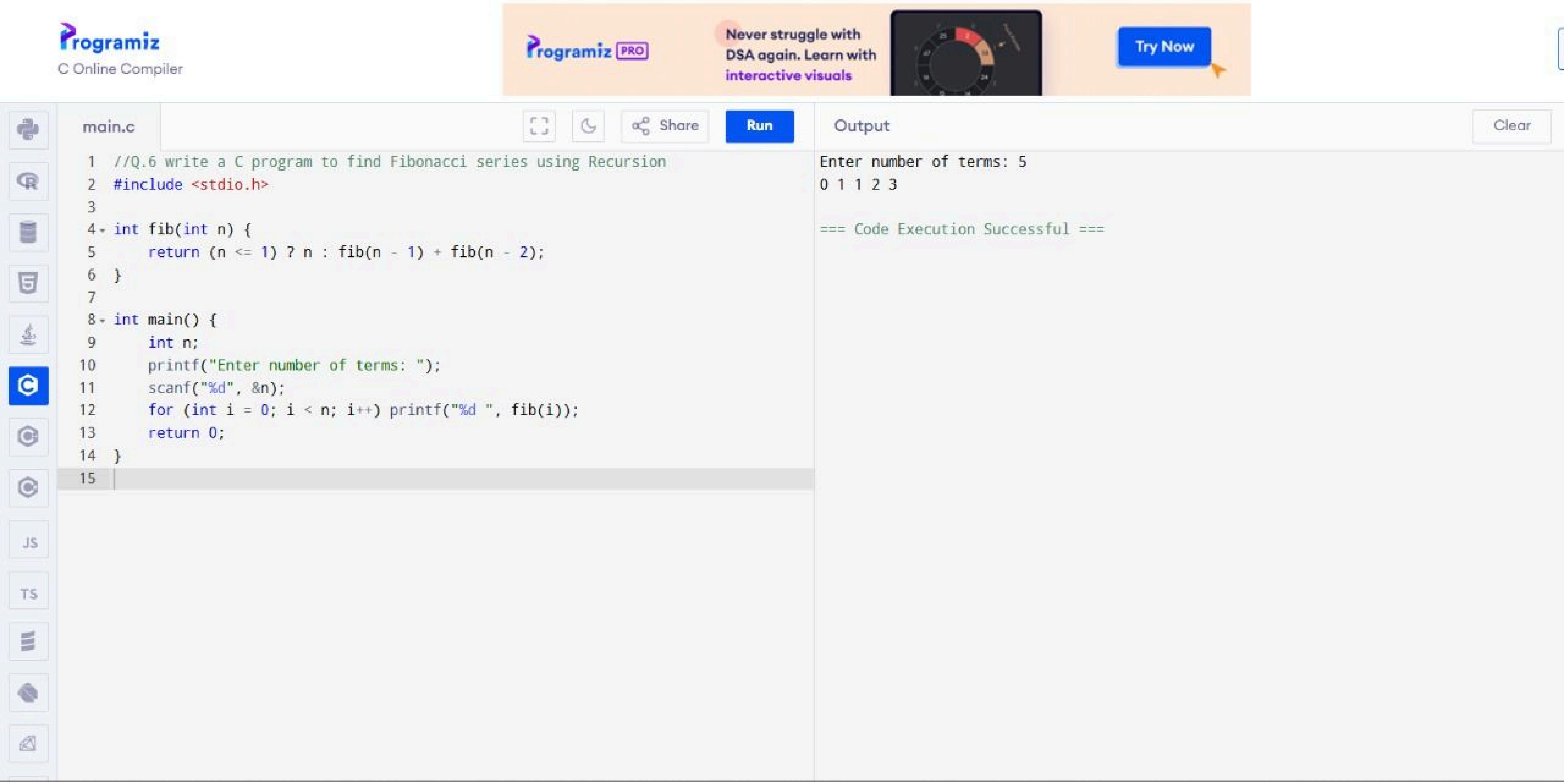
```
Enter number of terms: 5
0 1 1 2 3

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



Programiz PRO
Never struggle with DSA again.
Learn with interactive visuals
Try Now

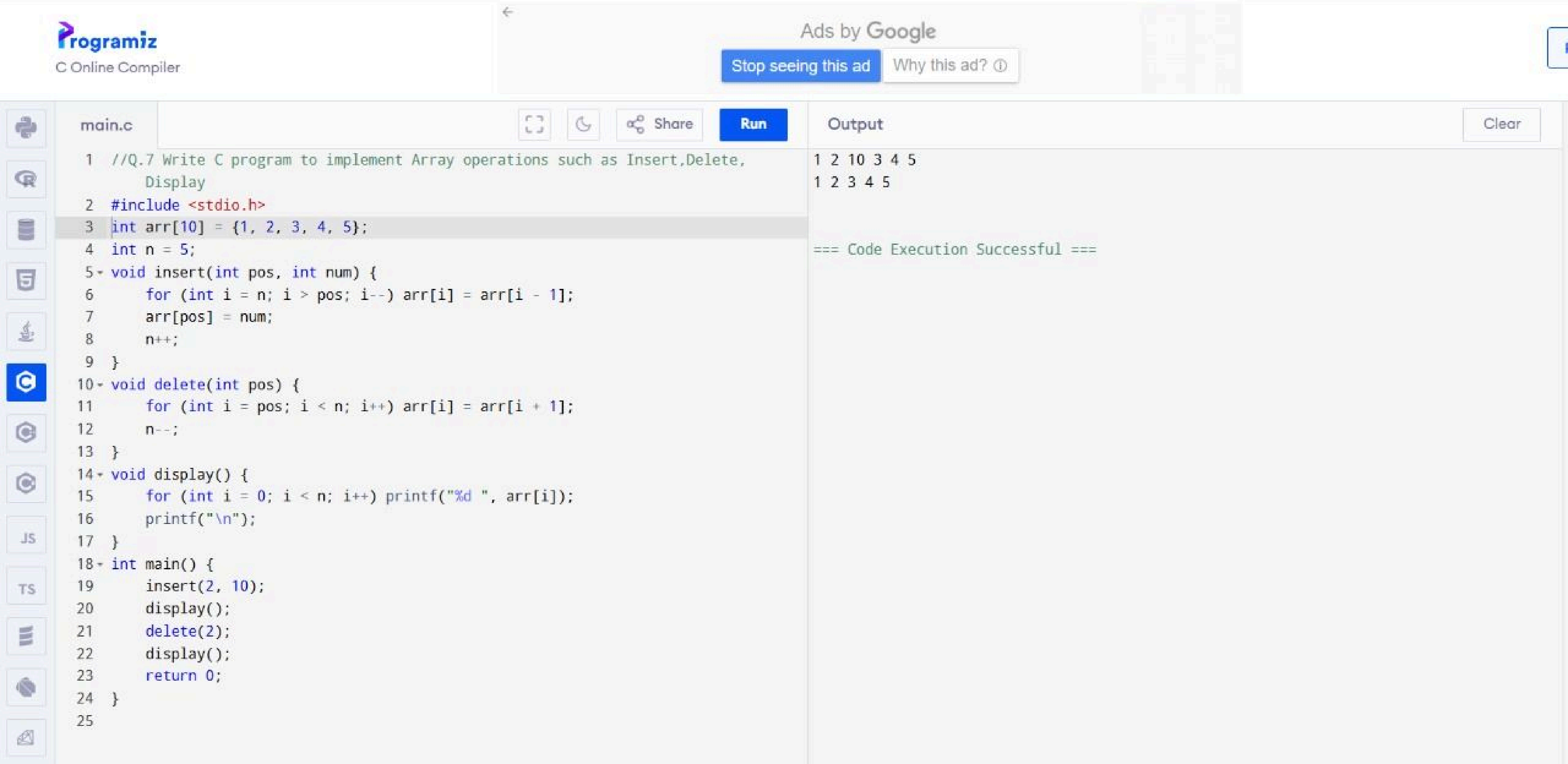
```
//Q.5 Write a C program to find Factorial of a given number using Recursion
#include <stdio.h>

int fact(int n) {
    return (n == 0 || n == 1) ? 1 : n * fact(n - 1);
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Factorial: %d", fact(num));
    return 0;
}

```

```
Enter a number: 5
Factorial: 120

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



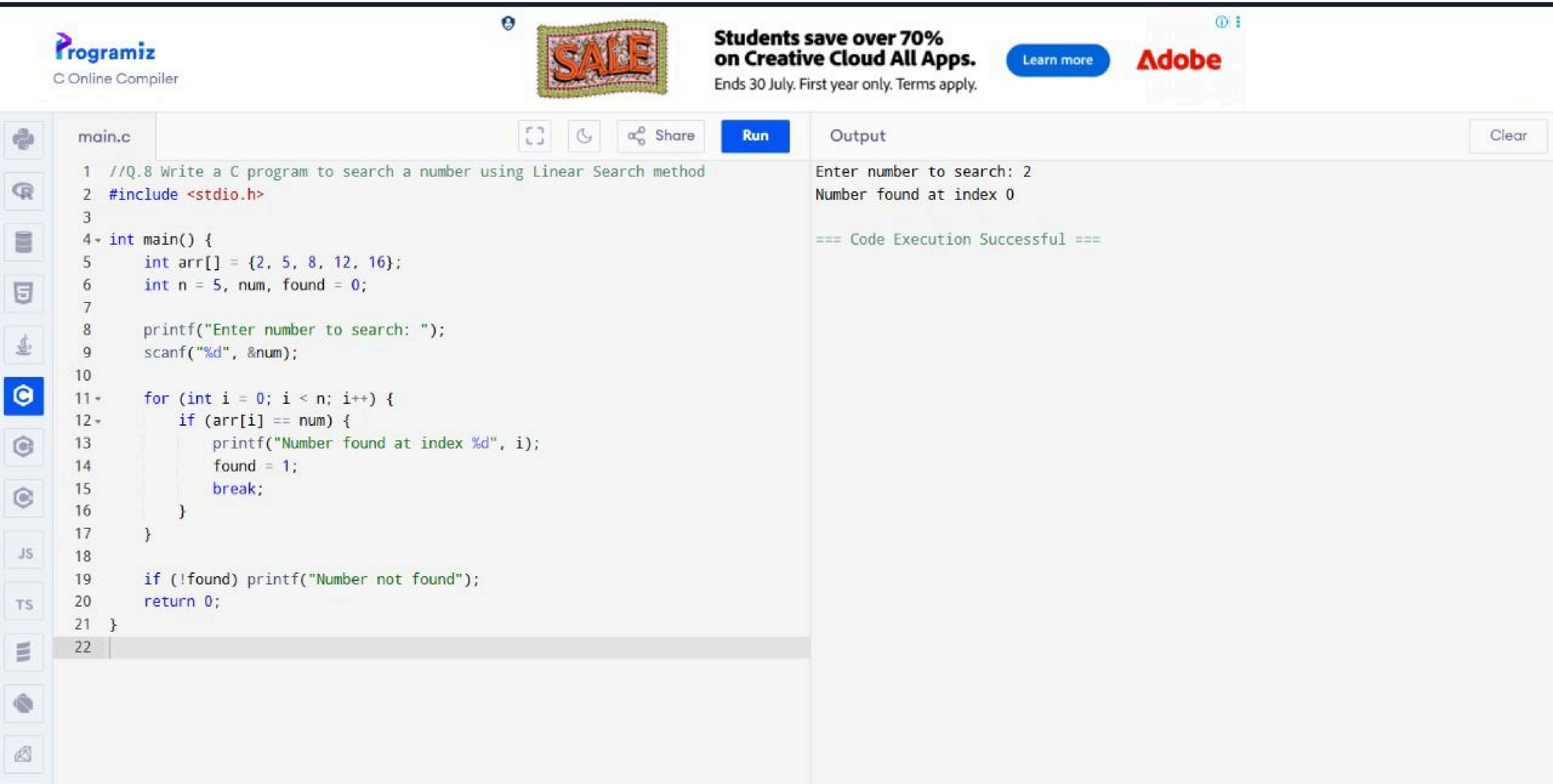
```c
//Q.6 write a C program to find Fibonacci series using Recursion
#include <stdio.h>

int fib(int n) {
    return (n <= 1) ? n : fib(n - 1) + fib(n - 2);
}

int main() {
    int n;
    printf("Enter number of terms: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++) printf("%d ", fib(i));
    return 0;
}
```

```
Enter number of terms: 5
0 1 1 2 3

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



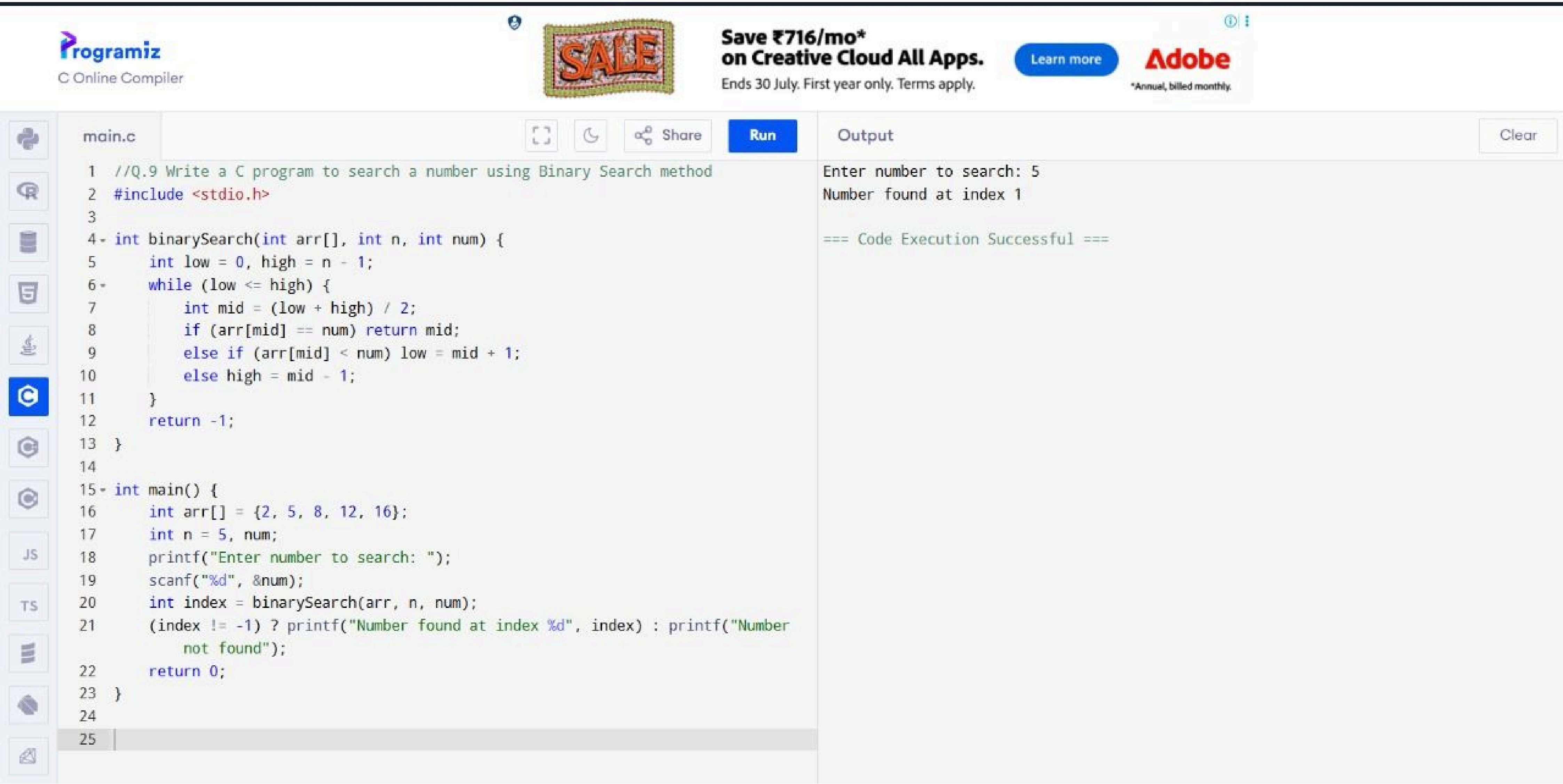
```c
//Q.7 Write C program to implement Array operations such as Insert,Delete, Display
#include <stdio.h>
int arr[10] = {1, 2, 3, 4, 5};
int n = 5;
void insert(int pos, int num) {
    for (int i = n; i > pos; i--) arr[i] = arr[i - 1];
    arr[pos] = num;
    n++;
}
void delete(int pos) {
    for (int i = pos; i < n; i++) arr[i] = arr[i + 1];
    n--;
}
void display() {
    for (int i = 0; i < n; i++) printf("%d ", arr[i]);
    printf("\n");
}
int main() {
    insert(2, 10);
    display();
    delete(2);
    display();
    return 0;
}
```

```
1 2 10 3 4 5
1 2 3 4 5


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


Programiz
C Online Compiler
Students save over 70% on Creative Cloud All Apps.
Ends 30 July. First year only. Terms apply.
Learn more
Adobe
main.c
Share
Run
Output
Clear

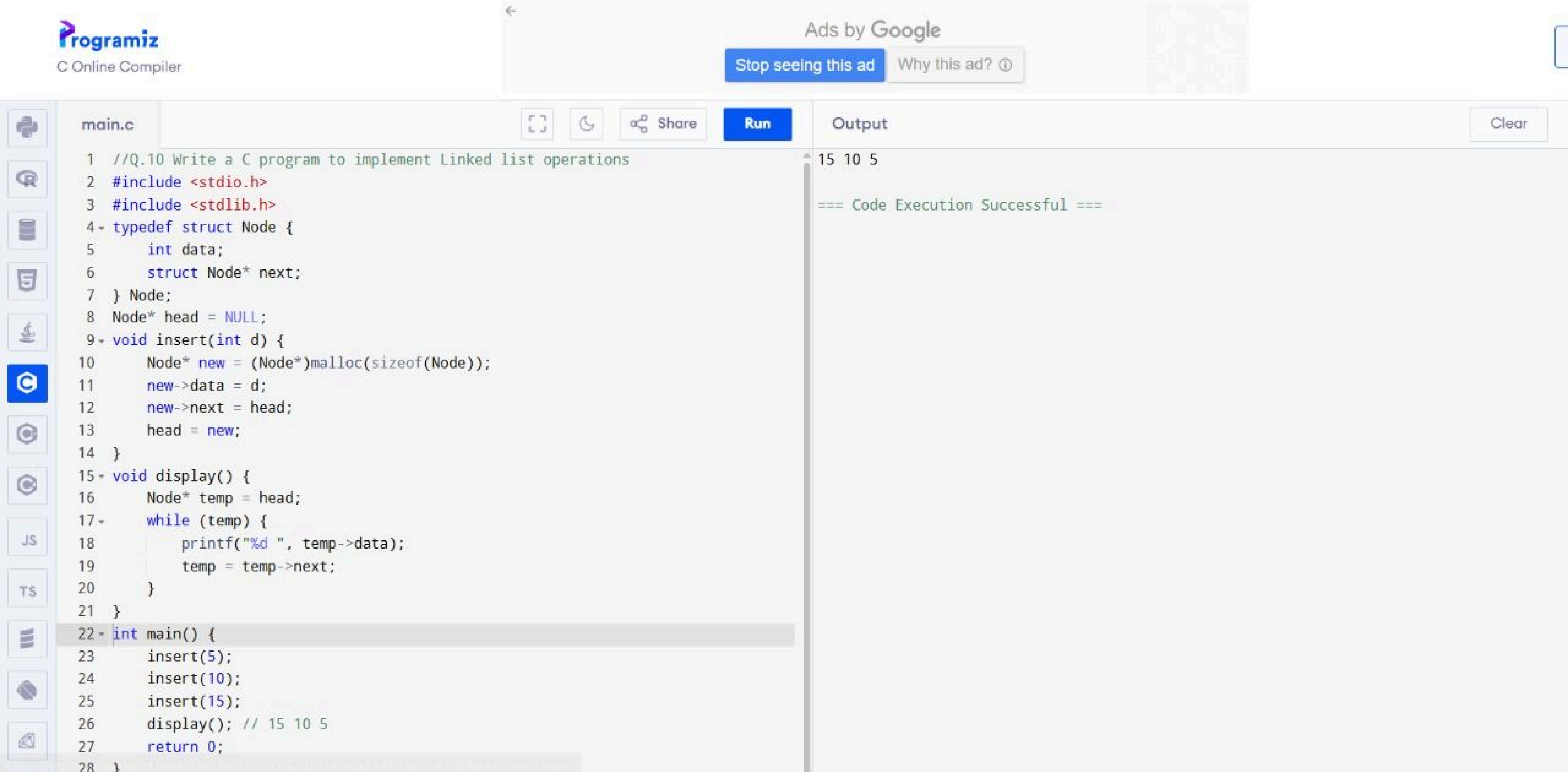
```c
//Q.8 Write a C program to search a number using Linear Search method
#include <stdio.h>

int main() {
    int arr[] = {2, 5, 8, 12, 16};
    int n = 5, num, found = 0;

    printf("Enter number to search: ");
    scanf("%d", &num);

    for (int i = 0; i < n; i++) {
        if (arr[i] == num) {
            printf("Number found at index %d", i);
            found = 1;
            break;
        }
    }

    if (!found) printf("Number not found");
    return 0;
}
```

```
Enter number to search: 2
Number found at index 0

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



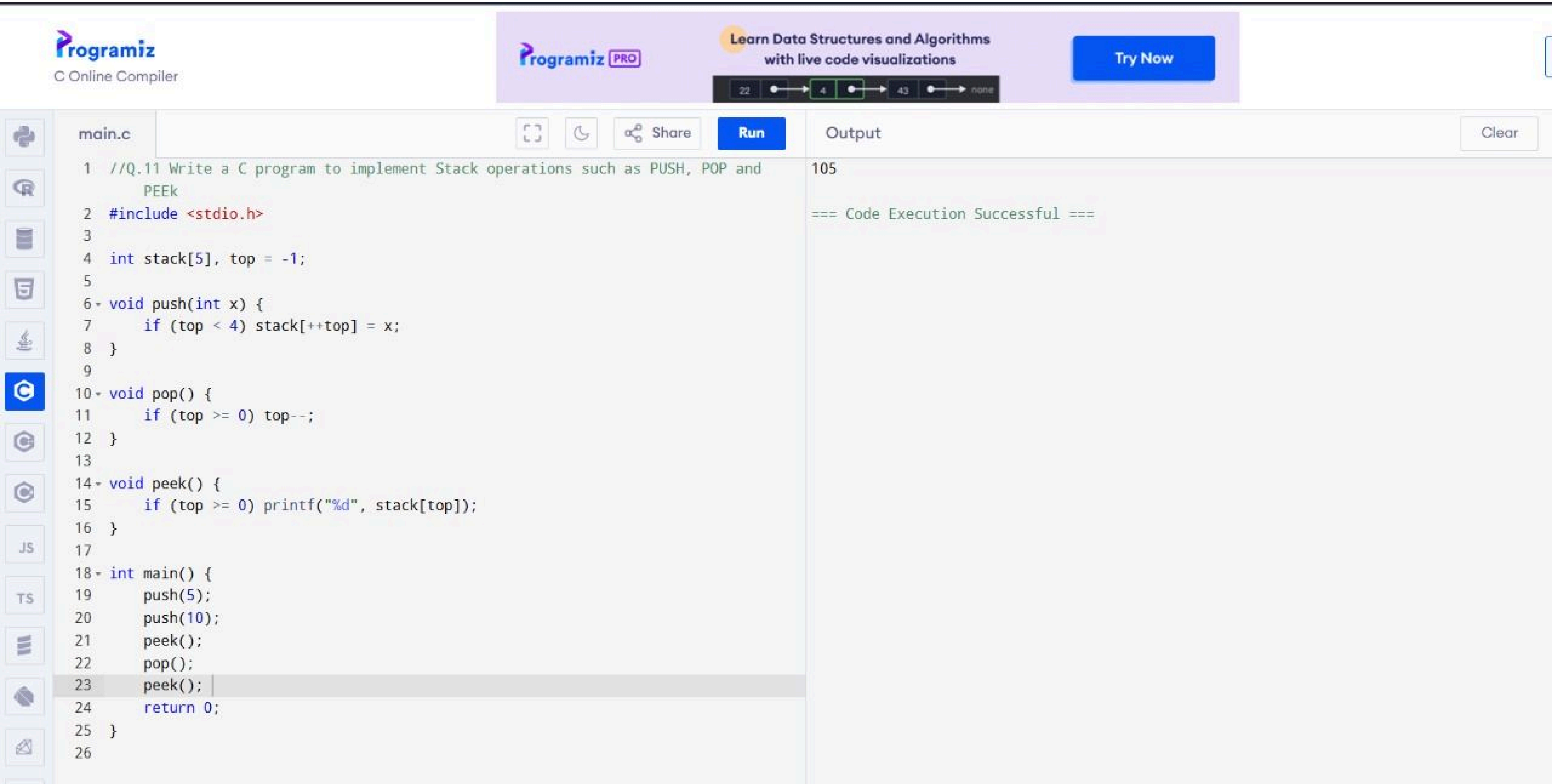
```c
//Q.9 Write a C program to search a number using Binary Search method
#include <stdio.h>

int binarySearch(int arr[], int n, int num) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == num) return mid;
        else if (arr[mid] < num) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}

int main() {
    int arr[] = {2, 5, 8, 12, 16};
    int n = 5, num;
    printf("Enter number to search: ");
    scanf("%d", &num);
    int index = binarySearch(arr, n, num);
    (index != -1) ? printf("Number found at index %d", index) : printf("Number not found");
    return 0;
}

```

```
Enter number to search: 5
Number found at index 1

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



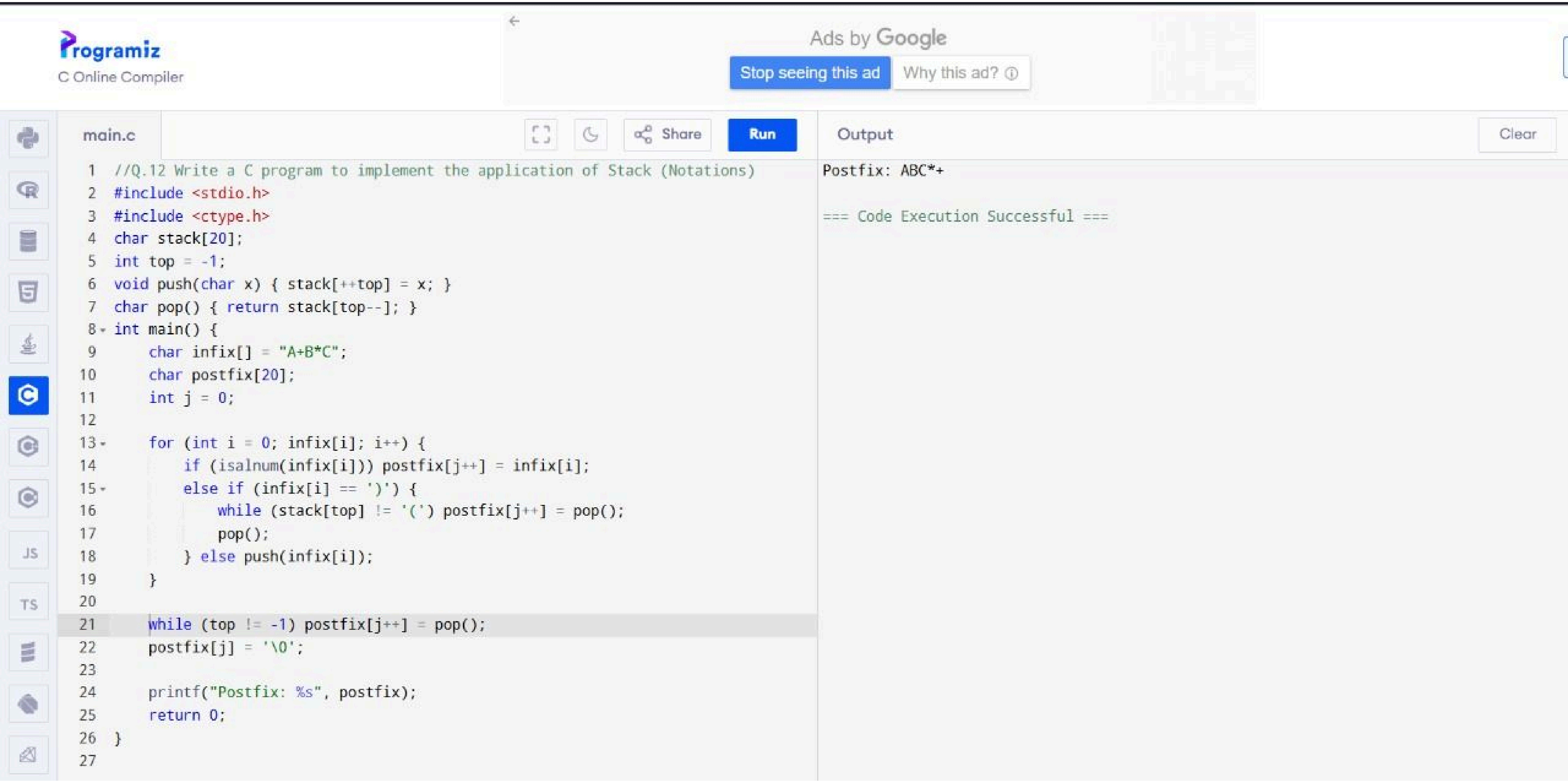
```c
//Q.10 Write a C program to implement Linked list operations
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* head = NULL;
void insert(int d) {
    Node* new = (Node*)malloc(sizeof(Node));
    new->data = d;
    new->next = head;
    head = new;
}
void display() {
    Node* temp = head;
    while (temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}
int main() {
    insert(5);
    insert(10);
    insert(15);
    display(); // 15 10 5
    return 0;
}
```

```
15 10 5

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



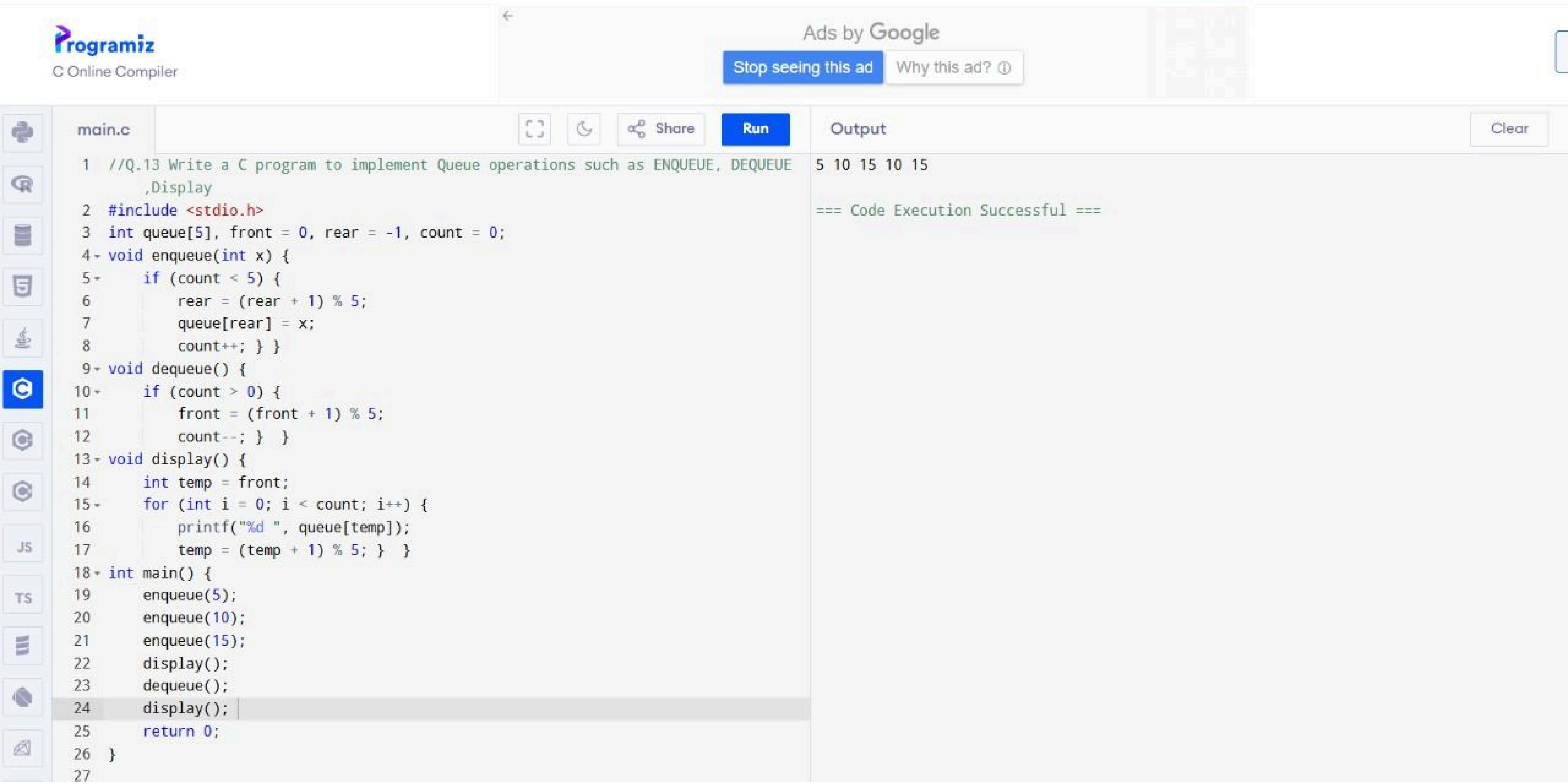
```c
//Q.11 Write a C program to implement Stack operations such as PUSH, POP and PEEk
#include <stdio.h>

int stack[5], top = -1;

void push(int x) {
    if (top < 4) stack[++top] = x;
}

void pop() {
    if (top >= 0) top--;
}

void peek() {
    if (top >= 0) printf("%d", stack[top]);
}

int main() {
    push(5);
    push(10);
    peek();
    pop();
    peek();
    return 0;
}
```

```
105

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



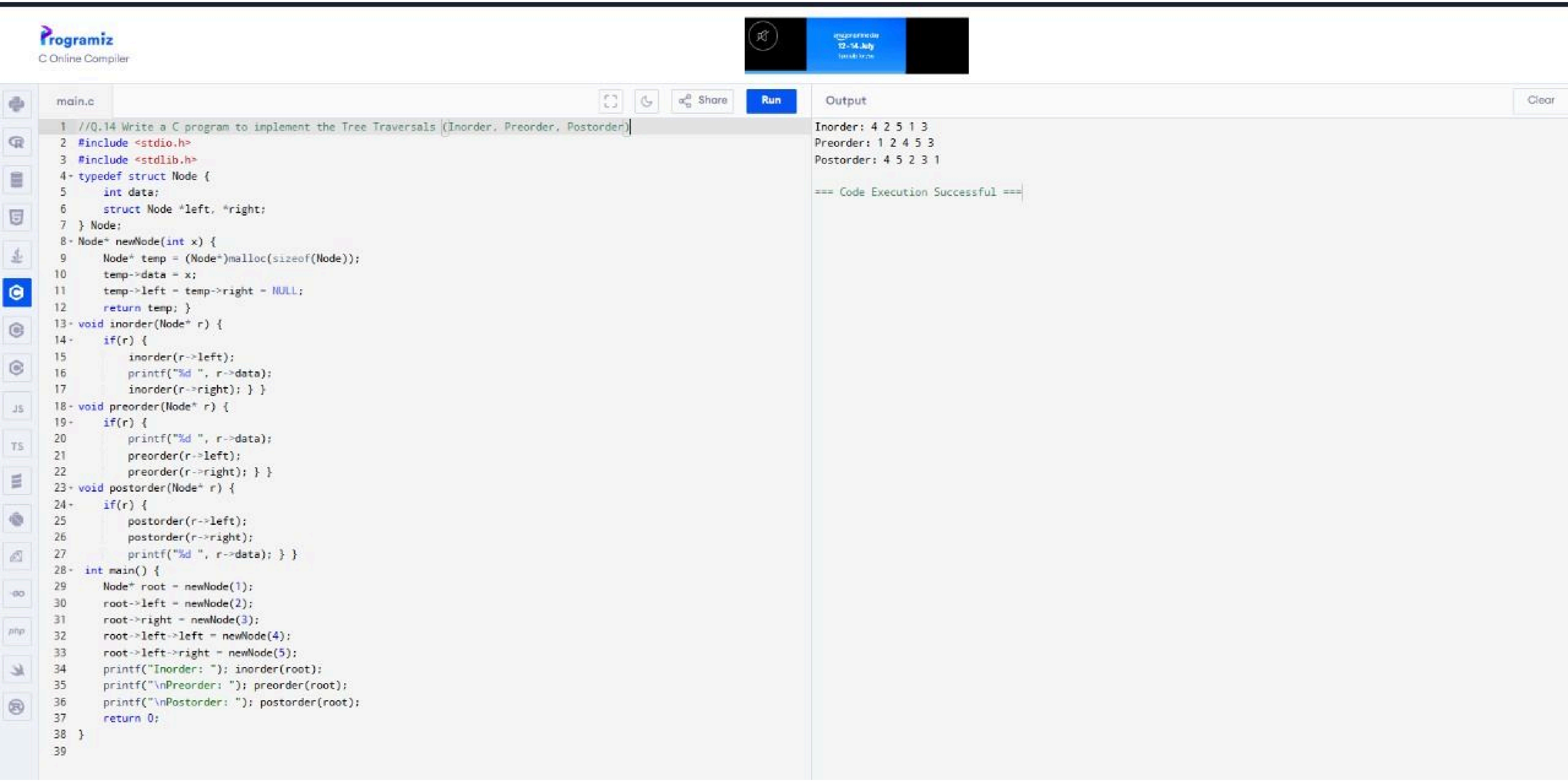
```c
//Q.12 Write a C program to implement the application of Stack (Notations)
#include <stdio.h>
#include <ctype.h>
char stack[20];
int top = -1;
void push(char x) { stack[++top] = x; }
char pop() { return stack[top--]; }
int main() {
    char infix[] = "A+B*C";
    char postfix[20];
    int j = 0;

    for (int i = 0; infix[i]; i++) {
        if (isalnum(infix[i])) postfix[j++] = infix[i];
        else if (infix[i] == ')') {
            while (stack[top] != '(') postfix[j++] = pop();
            pop();
        } else push(infix[i]);
    }

    while (top != -1) postfix[j++] = pop();
    postfix[j] = '\0';

    printf("Postfix: %s", postfix);
    return 0;
}
```

```
Postfix: ABC*+

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



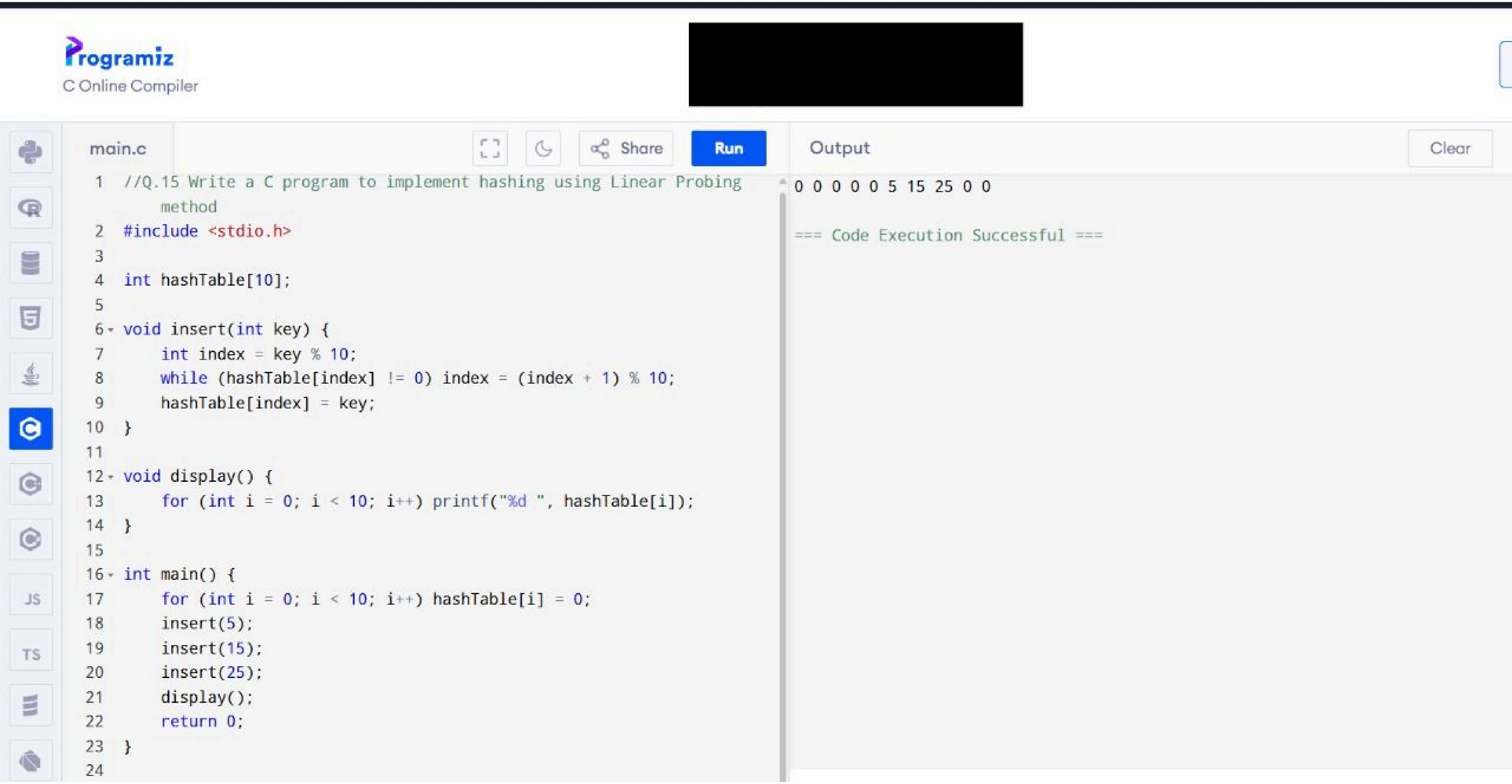
```c
//Q.13 Write a C program to implement Queue operations such as ENQUEUE, DEQUEUE ,Display
#include <stdio.h>
int queue[5], front = 0, rear = -1, count = 0;
void enqueue(int x) {
    if (count < 5) {
        rear = (rear + 1) % 5;
        queue[rear] = x;
        count++; } }
void dequeue() {
    if (count > 0) {
        front = (front + 1) % 5;
        count--; }  }
void display() {
    int temp = front;
    for (int i = 0; i < count; i++) {
        printf("%d ", queue[temp]);
        temp = (temp + 1) % 5; }  }
int main() {
    enqueue(5);
    enqueue(10);
    enqueue(15);
    display();
    dequeue();
    display();
    return 0;
}

```

```
5 10 15 10 15

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



```c
//Q.14 Write a C program to implement the Tree Traversals (Inorder, Preorder, Postorder)
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node *left, *right;
} Node;
Node* newNode(int x) {
    Node* temp = (Node*)malloc(sizeof(Node));
    temp->data = x;
    temp->left = temp->right = NULL;
    return temp; }
void inorder(Node* r) {
    if(r) {
        inorder(r->left);
        printf("%d ", r->data);
        inorder(r->right); } }
void preorder(Node* r) {
    if(r) {
        printf("%d ", r->data);
        preorder(r->left);
        preorder(r->right); } }
void postorder(Node* r) {
    if(r) {
        postorder(r->left);
        postorder(r->right);
        printf("%d ", r->data); } }
 int main() {
    Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    printf("Inorder: "); inorder(root);
    printf("\nPreorder: "); preorder(root);
    printf("\nPostorder: "); postorder(root);
    return 0;
}
```

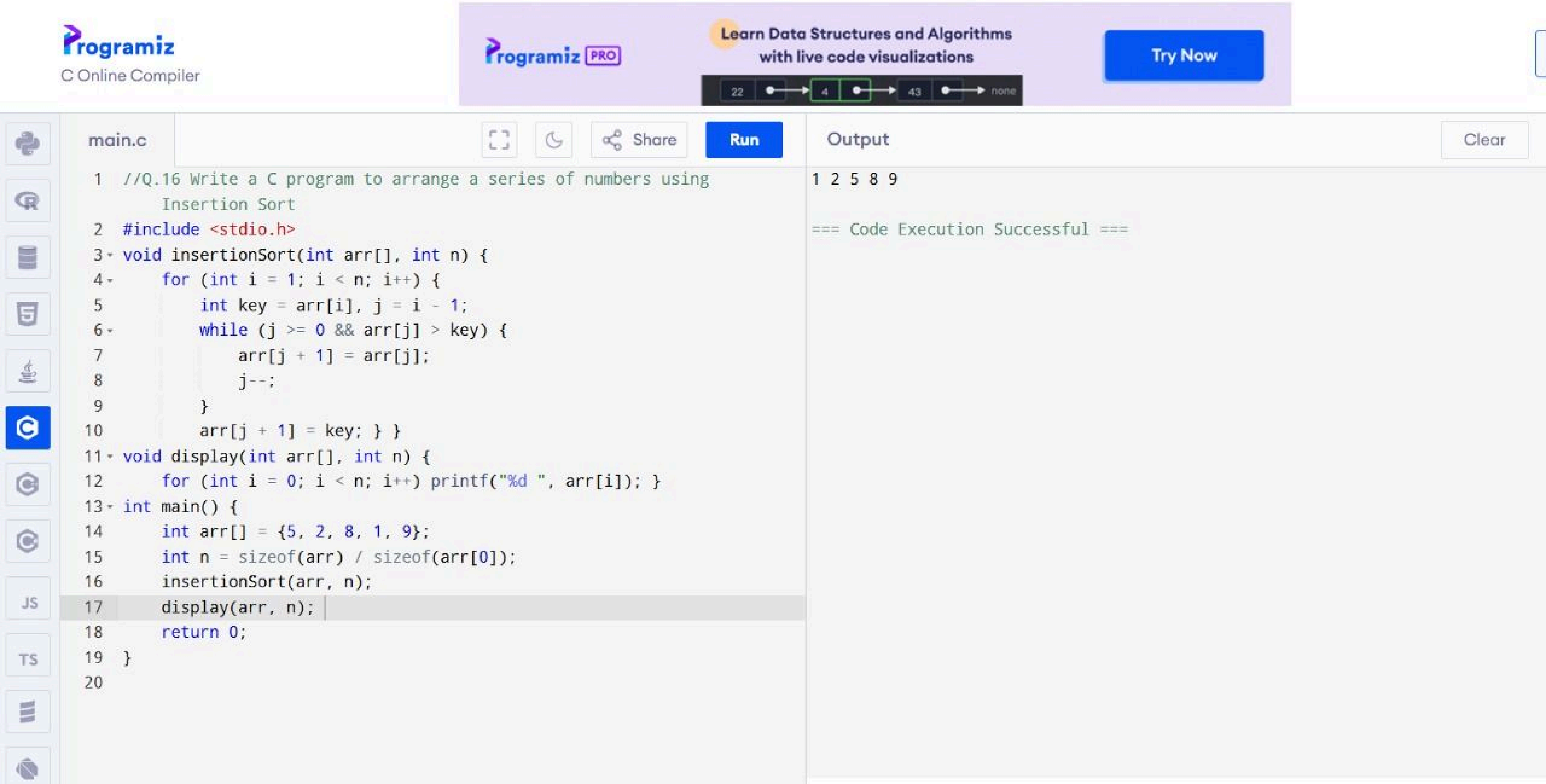
```
Inorder: 4 2 5 1 3
Preorder: 1 2 4 5 3
Postorder: 4 5 2 3 1

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



[redacted]

```c
//Q.15 Write a C program to implement hashing using Linear Probing method
#include <stdio.h>

int hashTable[10];

void insert(int key) {
    int index = key % 10;
    while (hashTable[index] != 0) index = (index + 1) % 10;
    hashTable[index] = key;
}

void display() {
    for (int i = 0; i < 10; i++) printf("%d ", hashTable[i]);
}

int main() {
    for (int i = 0; i < 10; i++) hashTable[i] = 0;
    insert(5);
    insert(15);
    insert(25);
    display();
    return 0;
}
```

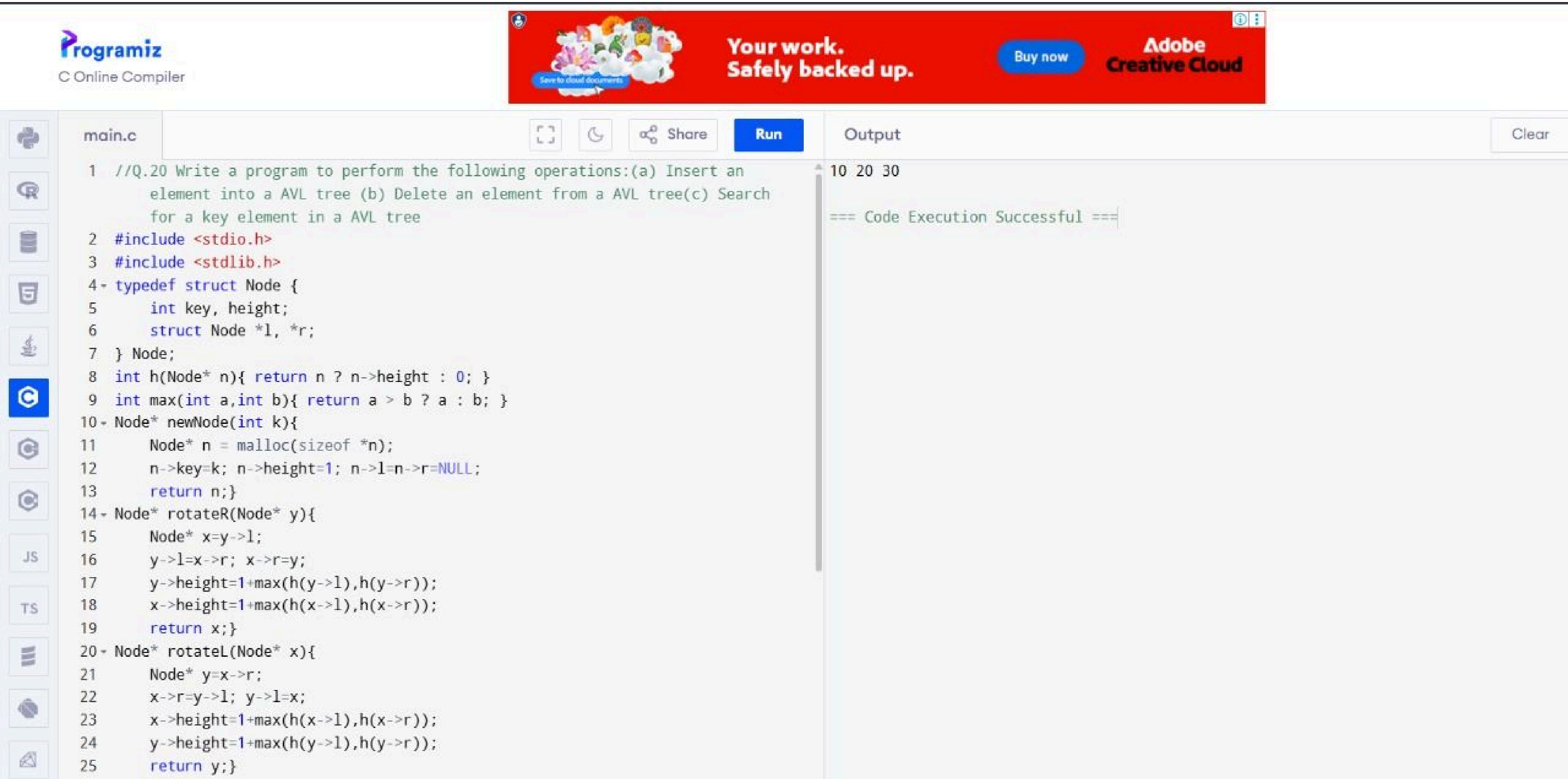
```
0 0 0 0 0 5 15 25 0 0

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



Programiz PRO
Learn Data Structures and Algorithms with live code visualizations
Try Now

```c
//Q.16 Write a C program to arrange a series of numbers using
    Insertion Sort
#include <stdio.h>
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i], j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key; } }
void display(int arr[], int n) {
    for (int i = 0; i < n; i++) printf("%d ", arr[i]); }
int main() {
    int arr[] = {5, 2, 8, 1, 9};
    int n = sizeof(arr) / sizeof(arr[0]);
    insertionSort(arr, n);
    display(arr, n);
    return 0;
}
```

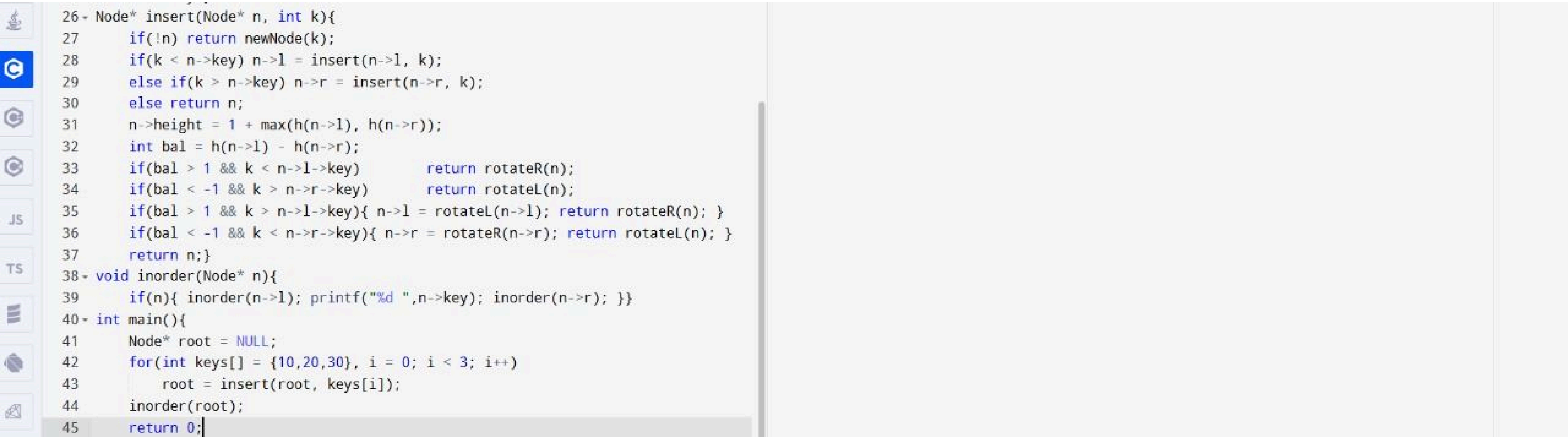
```
1 2 5 8 9

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



Your work. Safely backed up. Buy now Adobe Creative Cloud

```c
//Q.20 Write a program to perform the following operations:(a) Insert an element into a AVL tree (b) Delete an element from a AVL tree(c) Search for a key element in a AVL tree
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int key, height;
    struct Node *l, *r;
} Node;
int h(Node* n){ return n ? n->height : 0; }
int max(int a,int b){ return a > b ? a : b; }
Node* newNode(int k){
    Node* n = malloc(sizeof *n);
    n->key=k; n->height=1; n->l=n->r=NULL;
    return n;}
Node* rotateR(Node* y){
    Node* x=y->l;
    y->l=x->r; x->r=y;
    y->height=1+max(h(y->l),h(y->r));
    x->height=1+max(h(x->l),h(x->r));
    return x;}
Node* rotateL(Node* x){
    Node* y=x->r;
    x->r=y->l; y->l=x;
    x->height=1+max(h(x->l),h(x->r));
    y->height=1+max(h(y->l),h(y->r));
    return y;}
```

```
10 20 30

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



```c
Node* insert(Node* n, int k){
    if(!n) return newNode(k);
    if(k < n->key) n->l = insert(n->l, k);
    else if(k > n->key) n->r = insert(n->r, k);
    else return n;
    n->height = 1 + max(h(n->l), h(n->r));
    int bal = h(n->l) - h(n->r);
    if(bal > 1 && k < n->l->key)        return rotateR(n);
    if(bal < -1 && k > n->r->key)       return rotateL(n);
    if(bal > 1 && k > n->l->key){ n->l = rotateL(n->l); return rotateR(n); }
    if(bal < -1 && k < n->r->key){ n->r = rotateR(n->r); return rotateL(n); }
    return n;}
void inorder(Node* n){
    if(n){ inorder(n->l); printf("%d ",n->key); inorder(n->r); }}
int main(){
    Node* root = NULL;
    for(int keys[] = {10,20,30}, i = 0; i < 3; i++)
        root = insert(root, keys[i]);
    inorder(root);
    return 0;
```