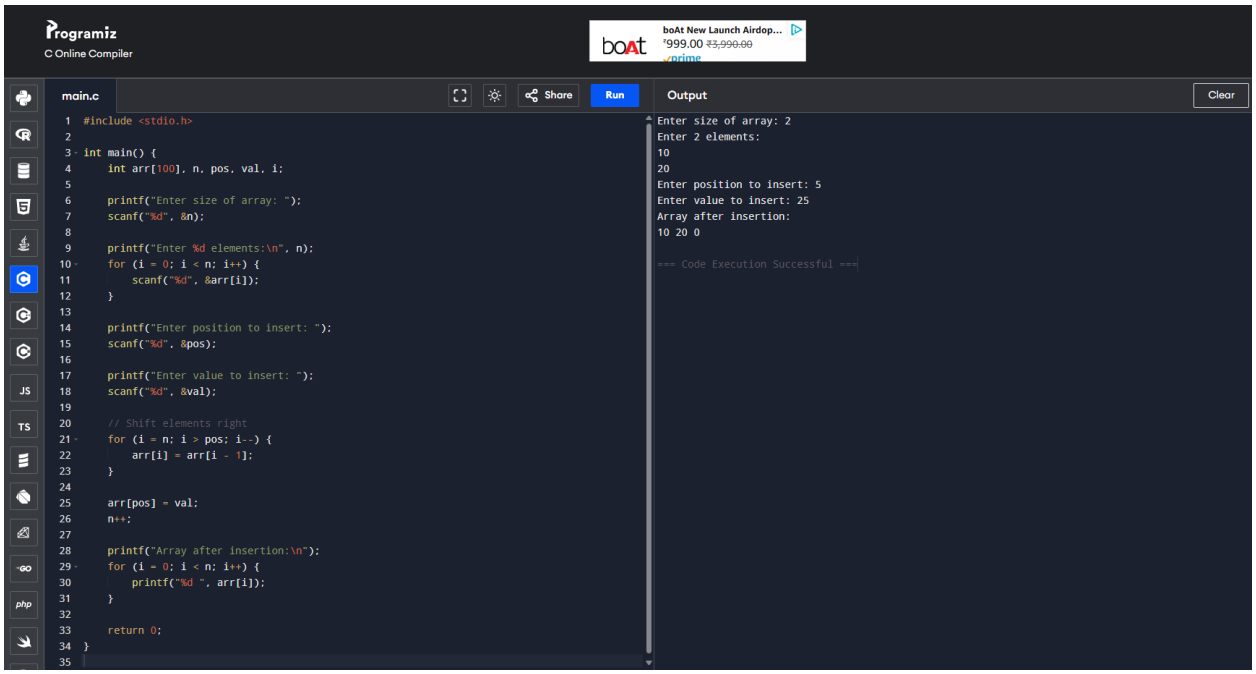
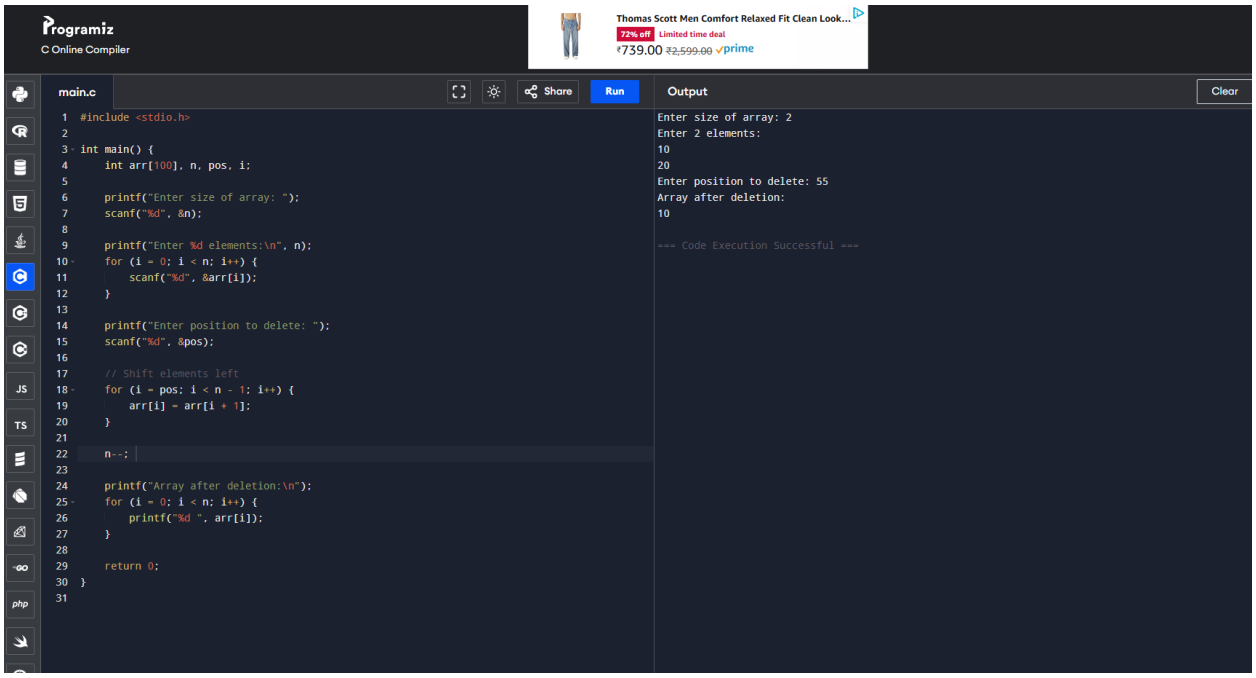
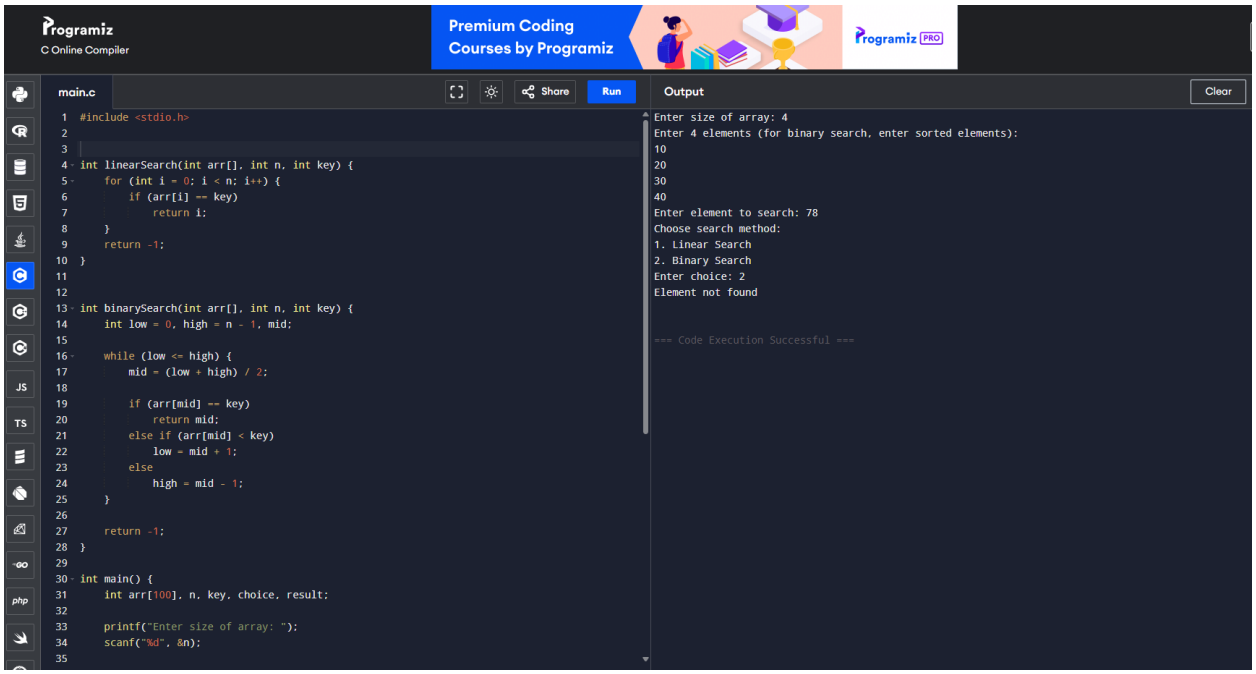
1. Write a program in c to implement insertion in 1-D arrays.

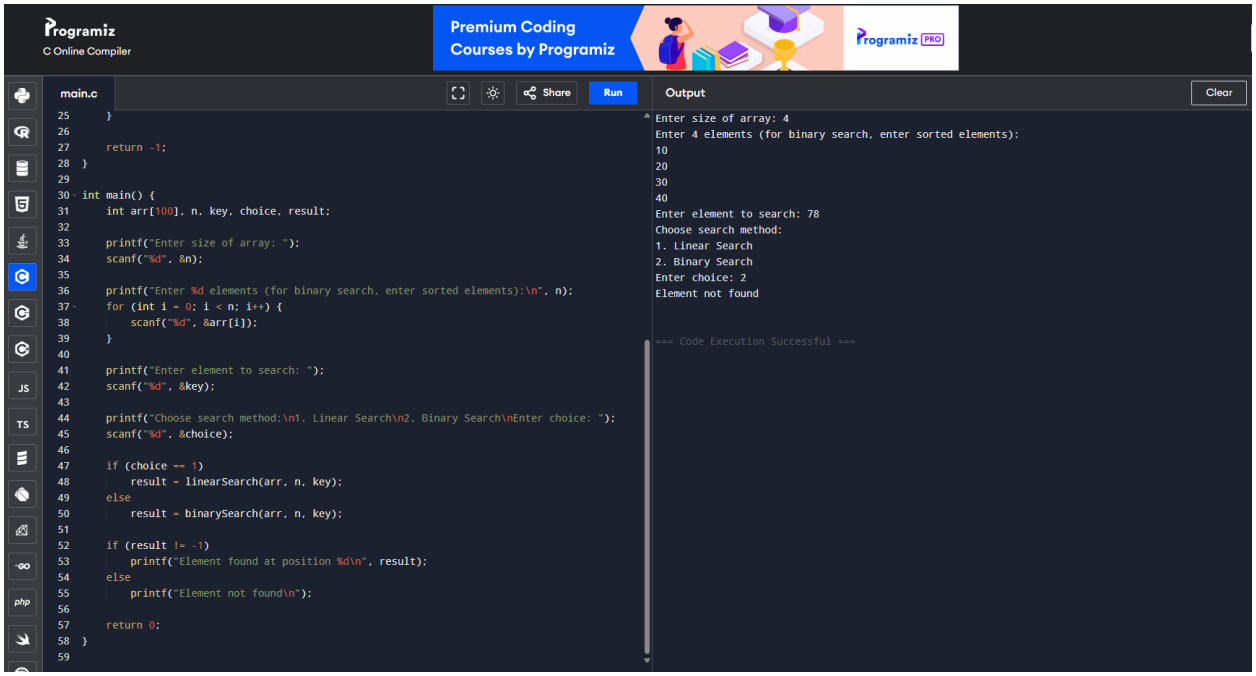


2. Write a program in c to implement deletion in 1-D arrays.

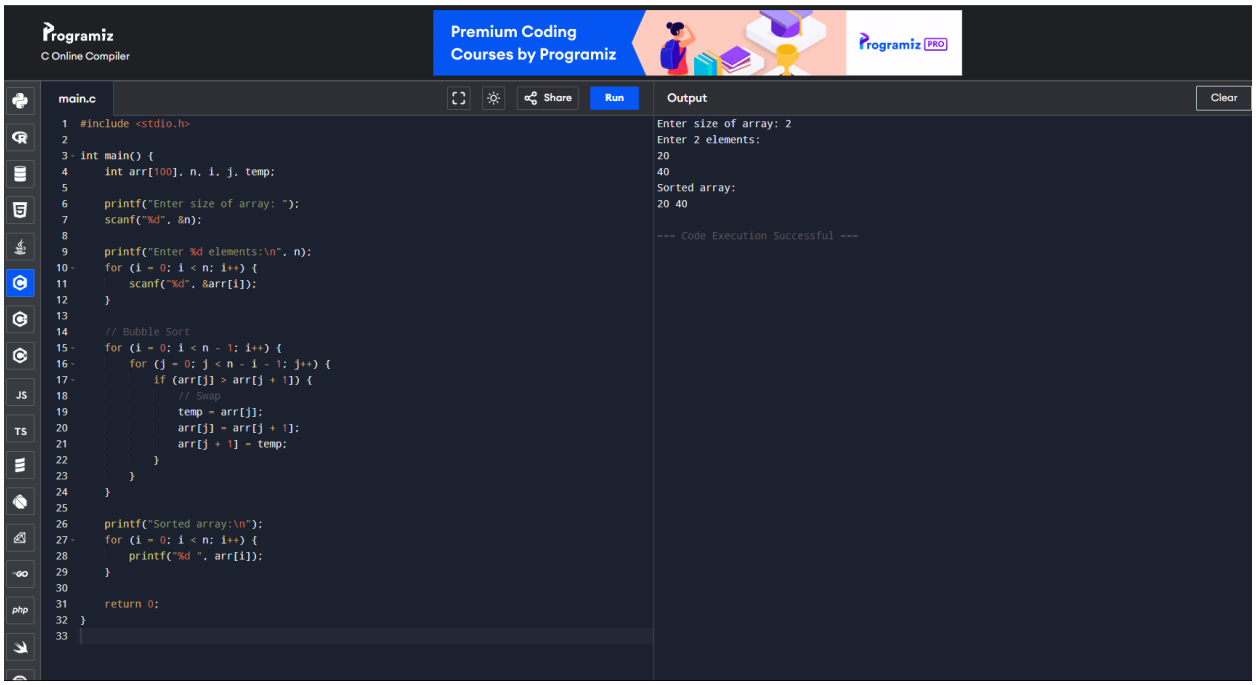


3. Write a program in c to implement linear and binary searching in 1-d arrays.

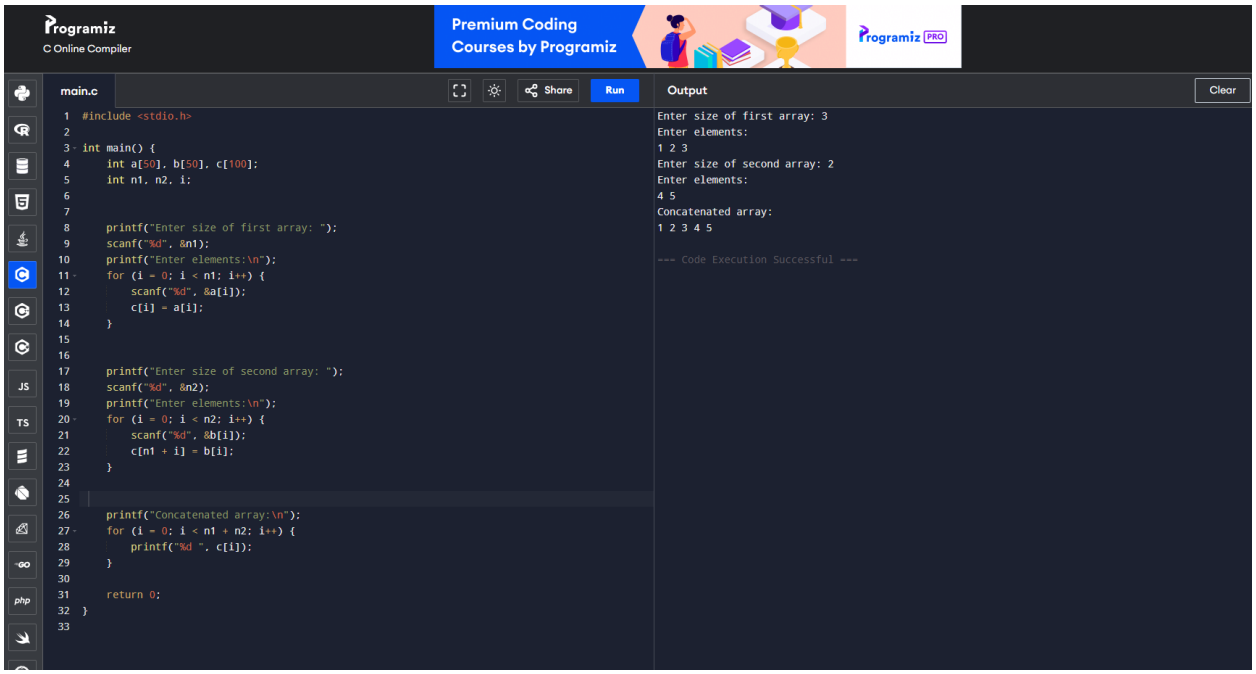




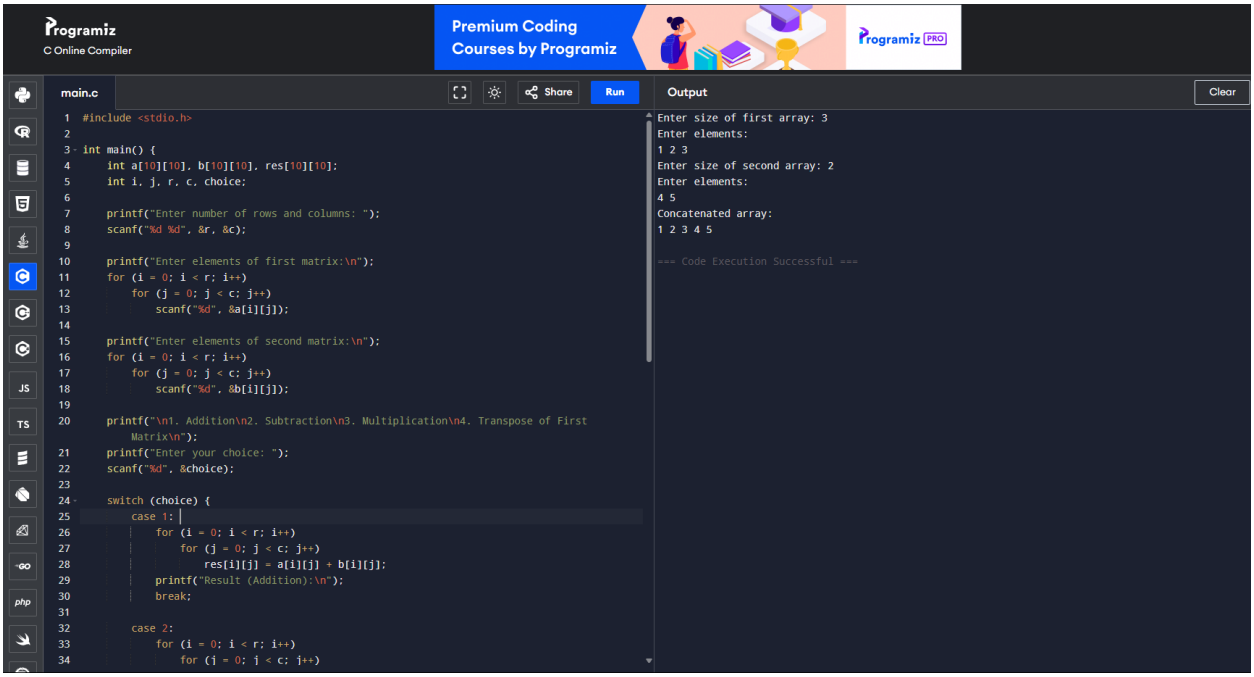
4. Write a program in c to implement sorting in 1-D arrays.

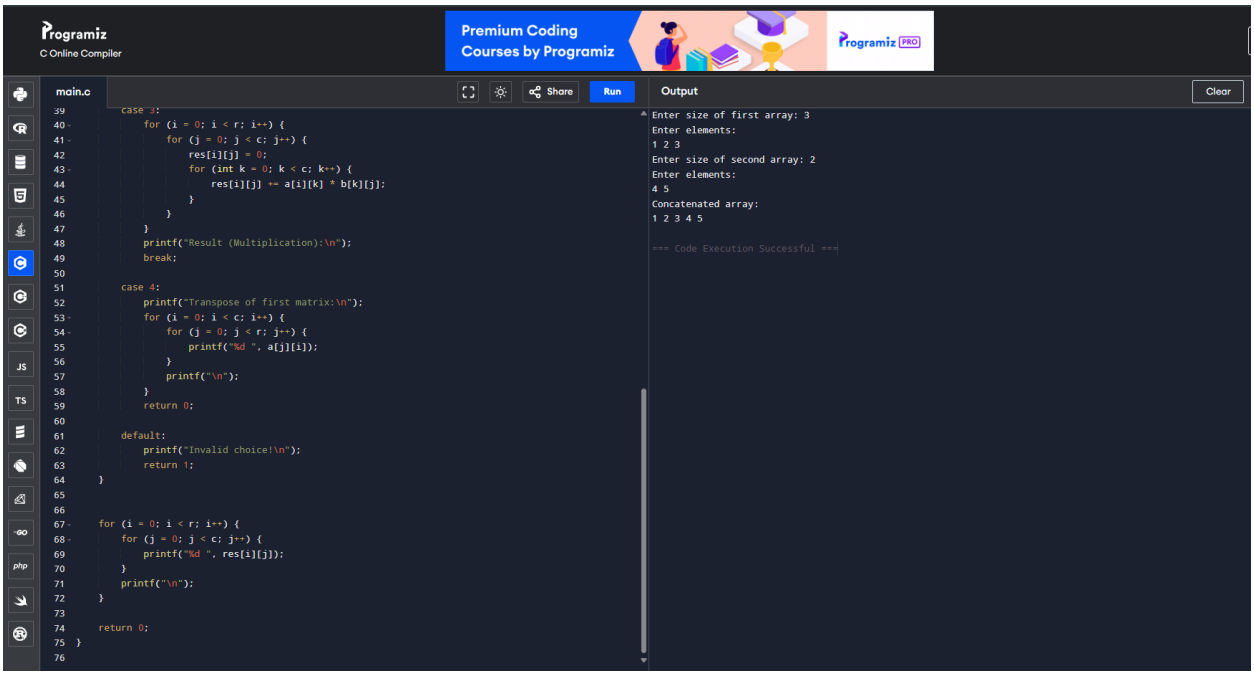


5. Write a program in c to concatenate two arrays.

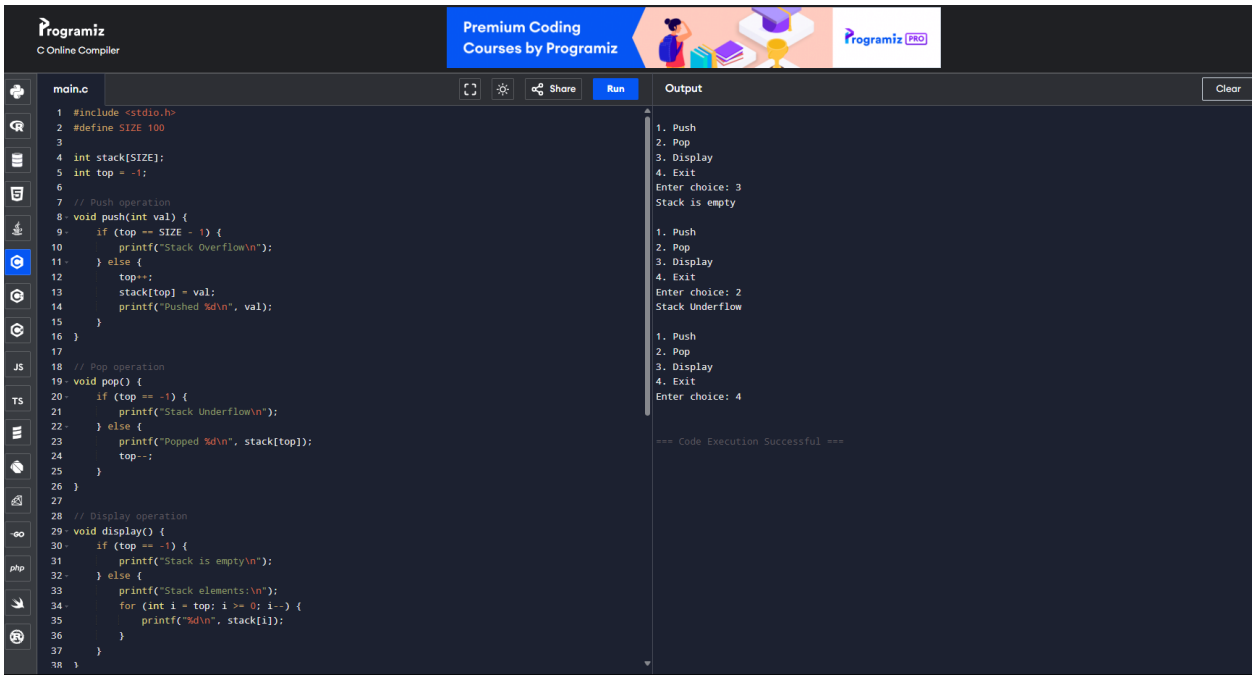


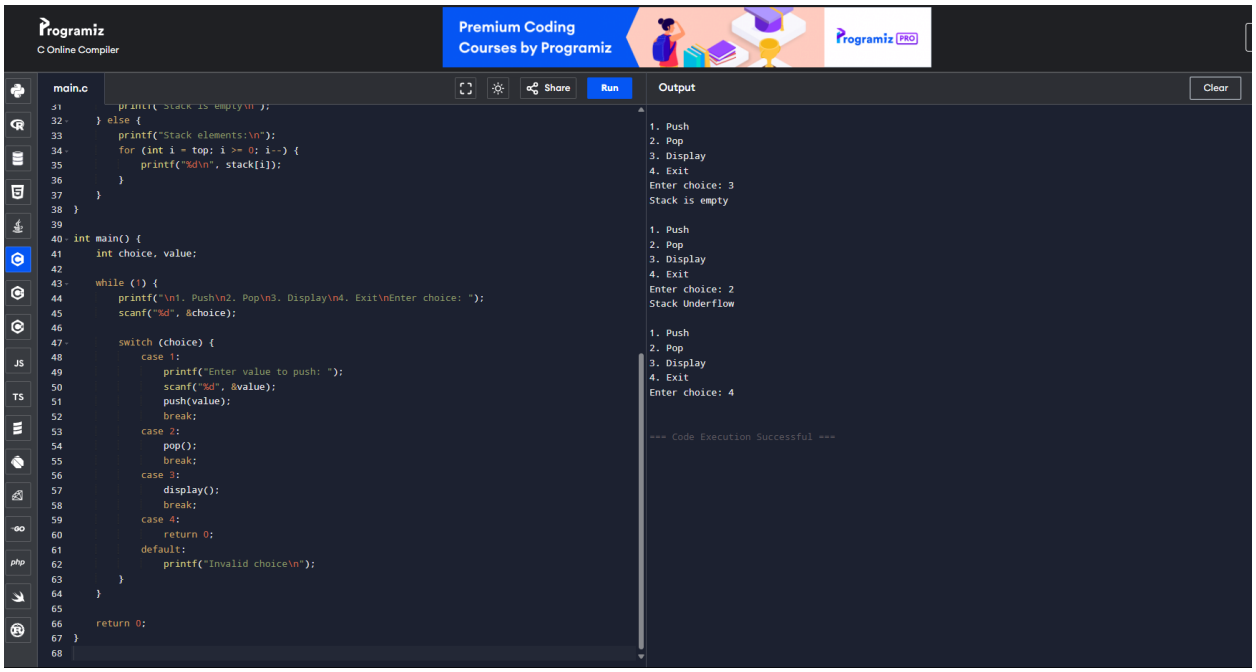
6. Write a program in C to implement the following Operations on 2-D Array (addition; subtraction; multiplication; transpose)



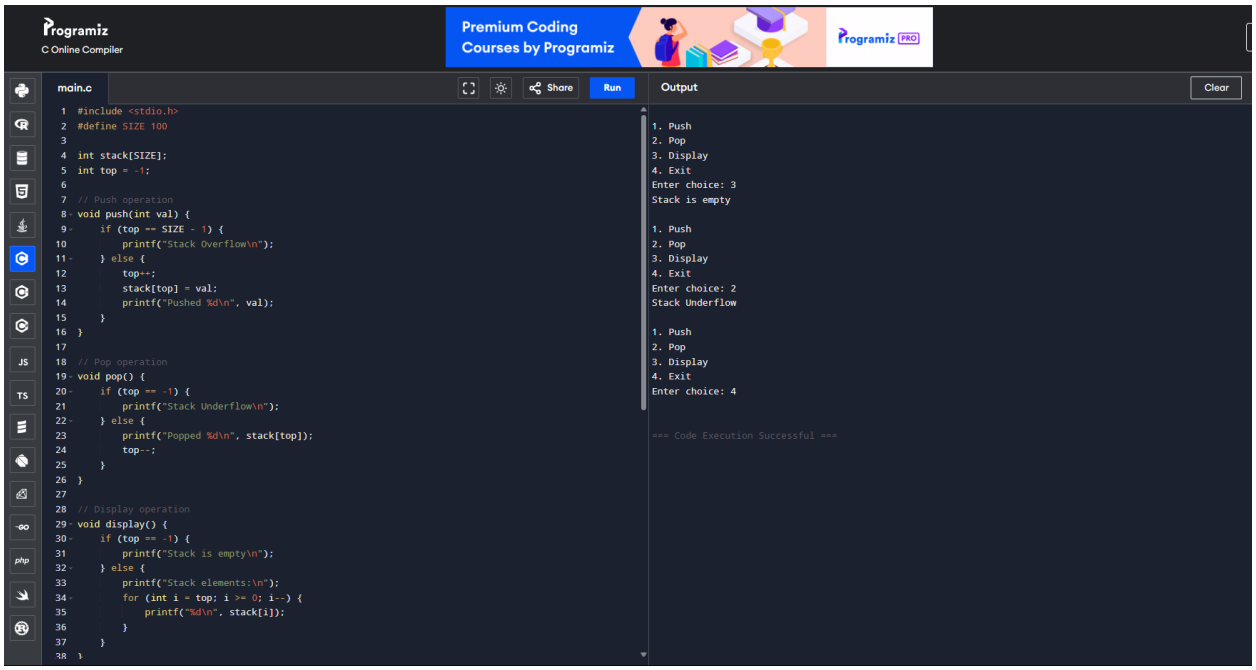


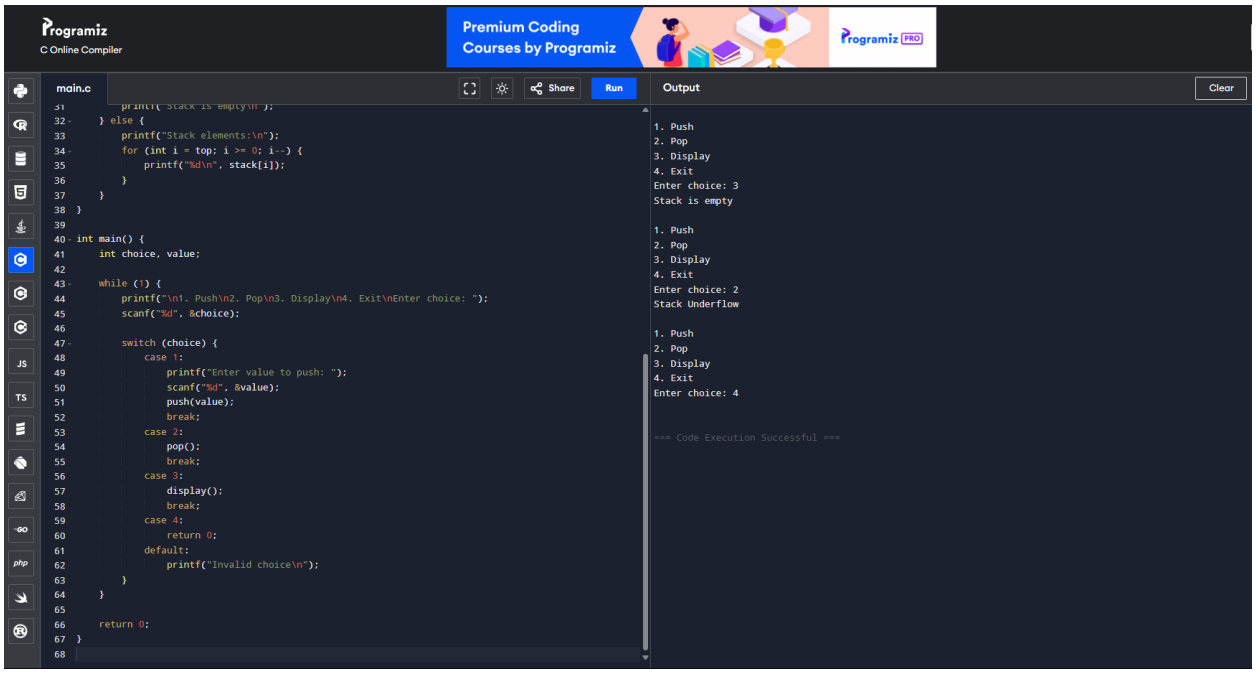
7. Write a program in C to implement operations on Stack using arrays.



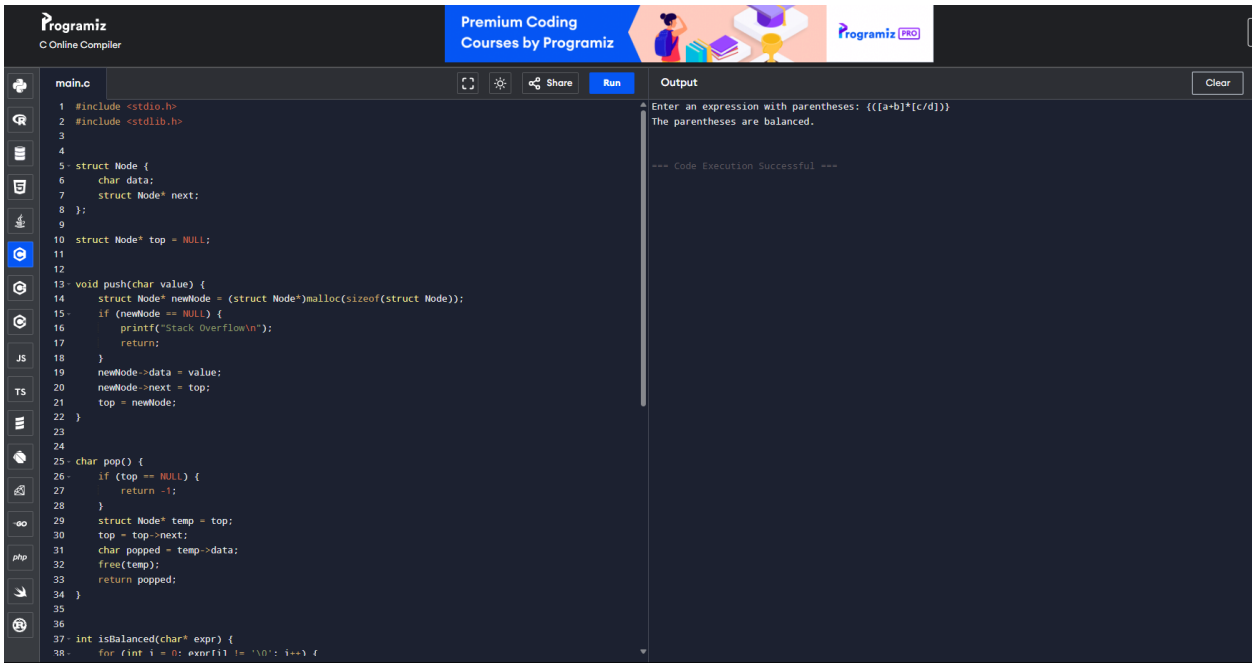


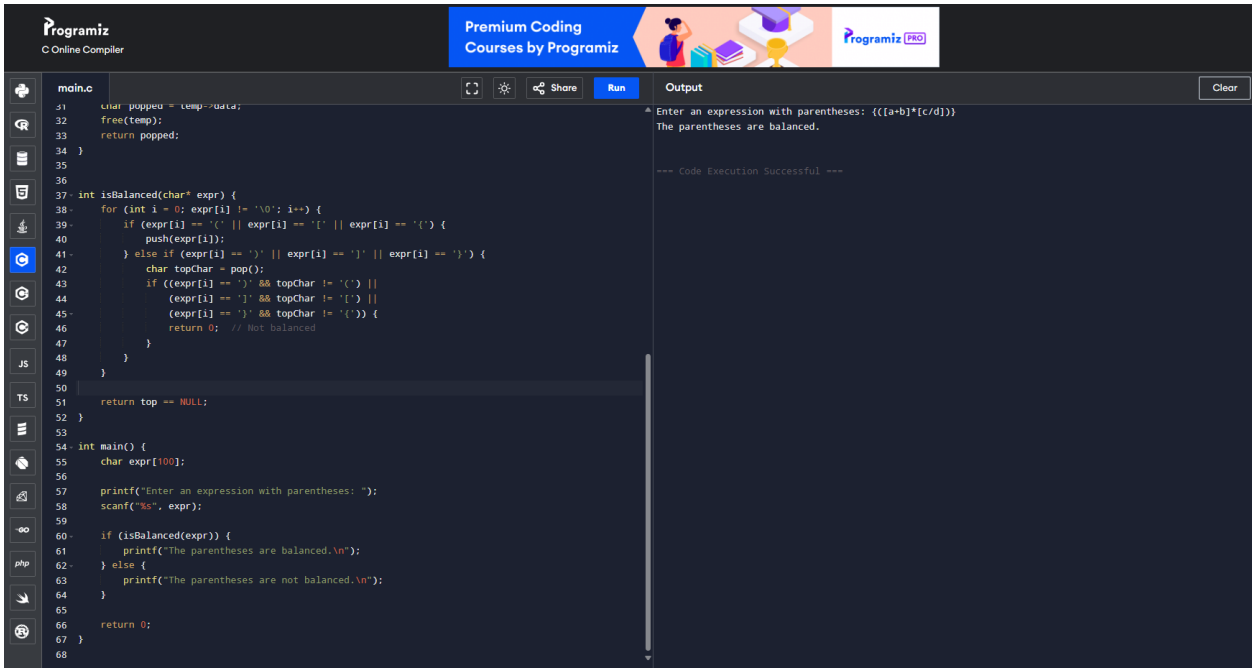
8. Write a program in c to implement operations on stack using linked list.



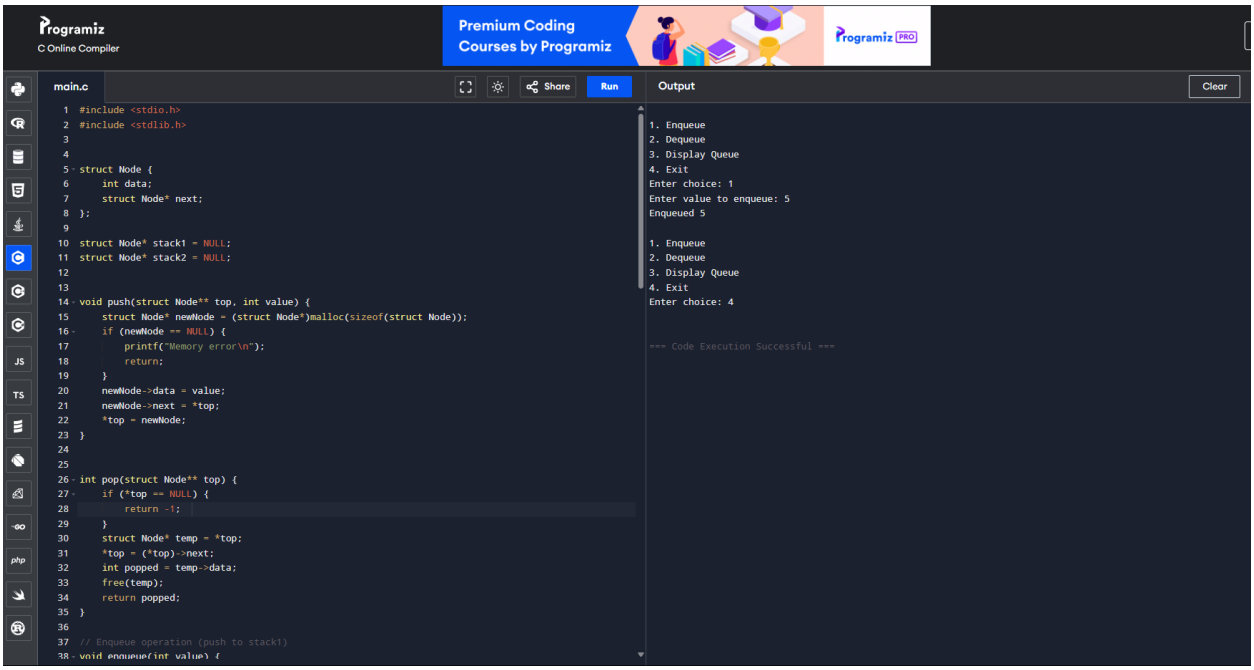


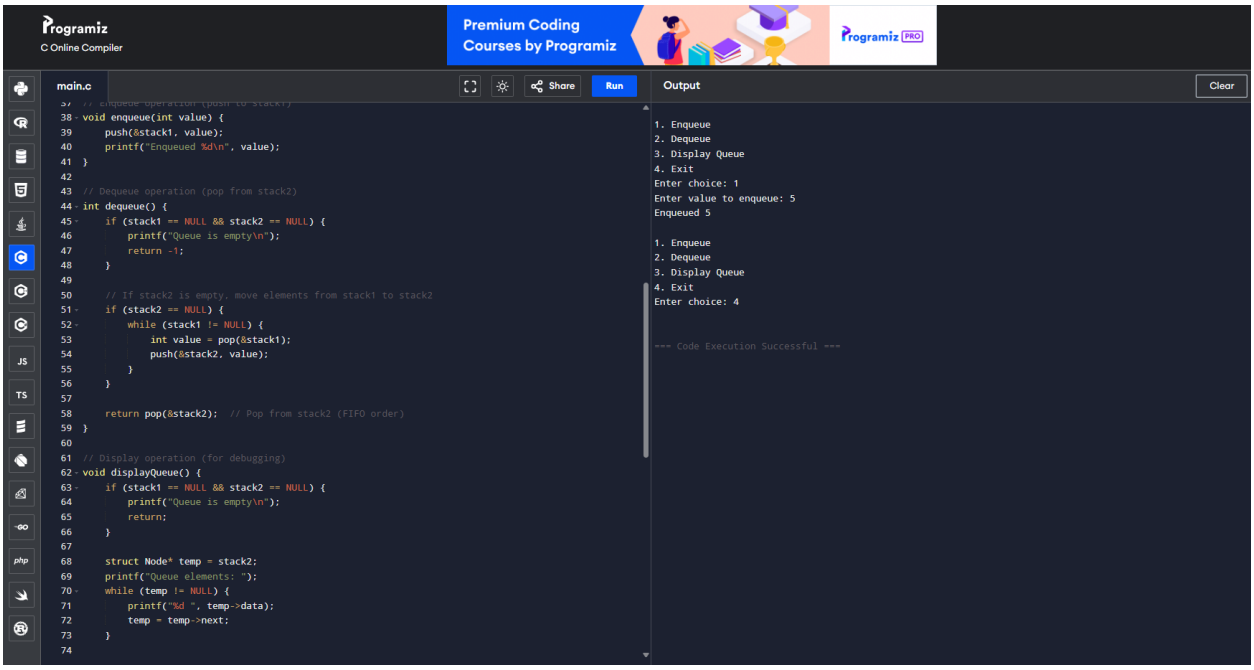
9. Write a program in c to implement applications of stack.

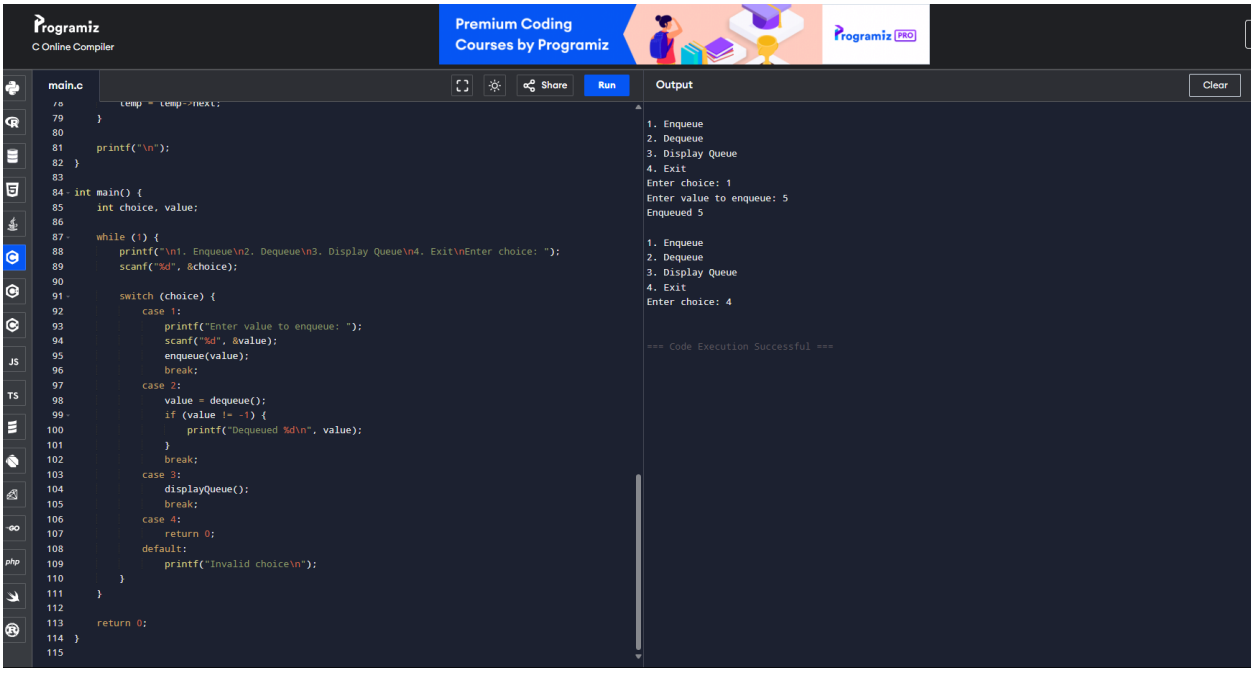




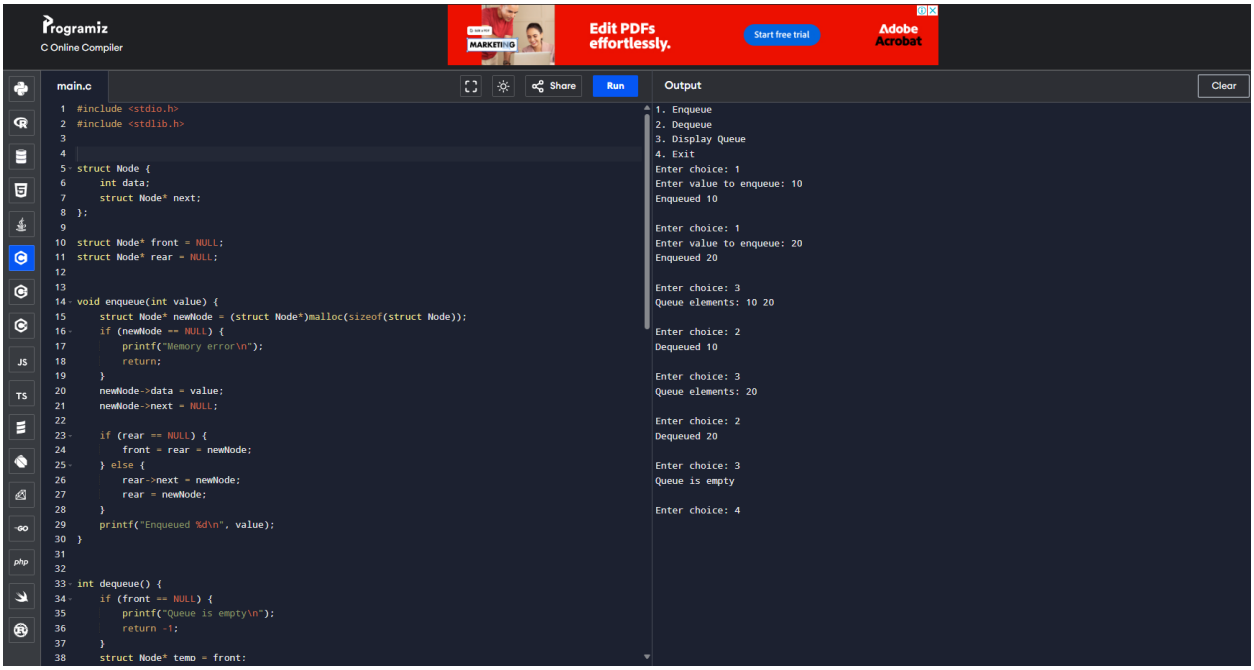
10. Write a program in c to implement operation on queue using stack.





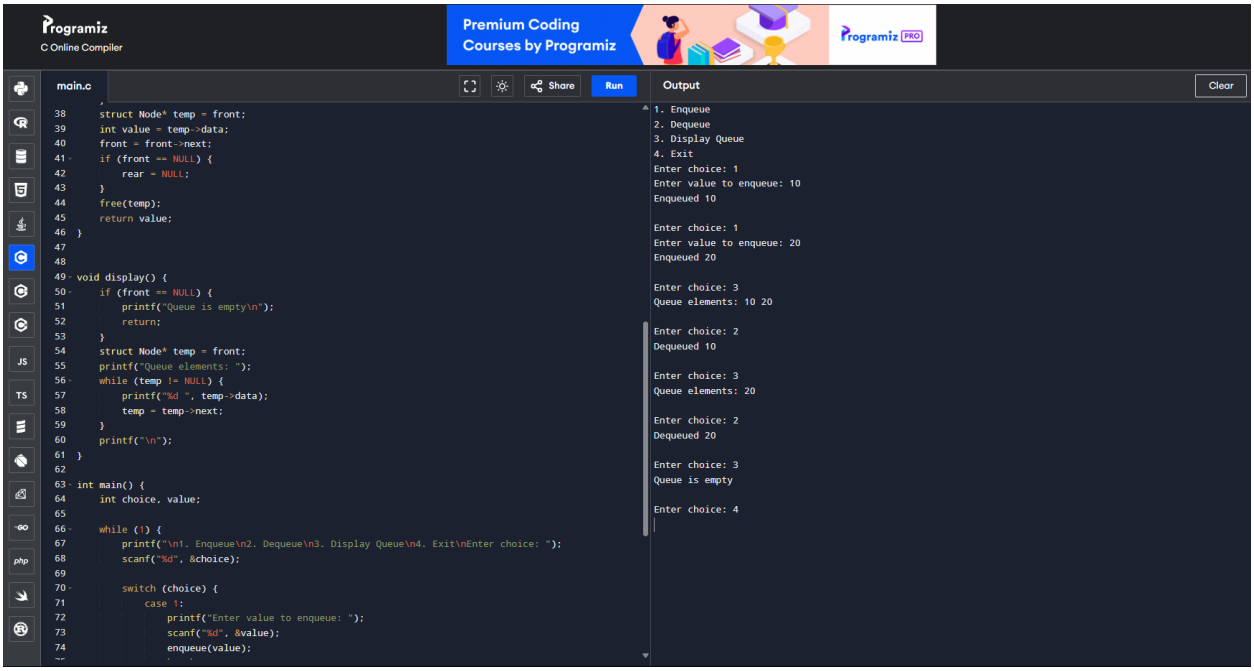

Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
}
printf("\n");
}
int main() {
int choice, value;
while (1) {
printf("\n1. Enqueue\n2. Dequeue\n3. Display Queue\n4. Exit\nEnter choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter value to enqueue: ");
scanf("%d", &value);
enqueue(value);
break;
case 2:
value = dequeue();
if (value != -1) {
printf("Dequeued %d\n", value);
}
break;
case 3:
displayQueue();
break;
case 4:
return 0;
default:
printf("Invalid choice\n");
}
}
return 0;
}
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 5
Enqueued 5
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 4
=== Code Execution Successful ===

11.Write a program in C to implement operations on queue using linked list simple code



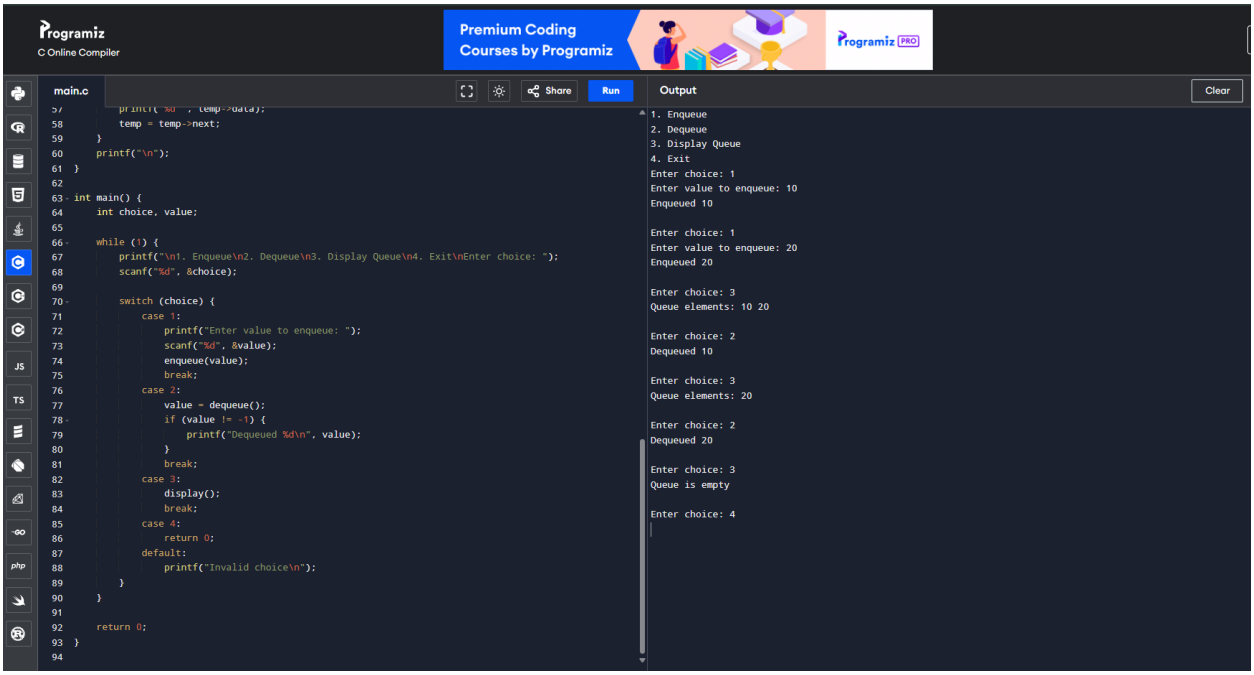
```c
#include <stdio.h>
#include <stdlib.h>


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

struct Node* front = NULL;
struct Node* rear = NULL;


void enqueue(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory error\n");
        return;
    }
    newNode->data = value;
    newNode->next = NULL;

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


int dequeue() {
    if (front == NULL) {
        printf("Queue is empty\n");
        return -1;
    }
    struct Node* temp = front;
```

```
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 20

Enter choice: 2
Dequeued 20

Enter choice: 3
Queue is empty

Enter choice: 4
```

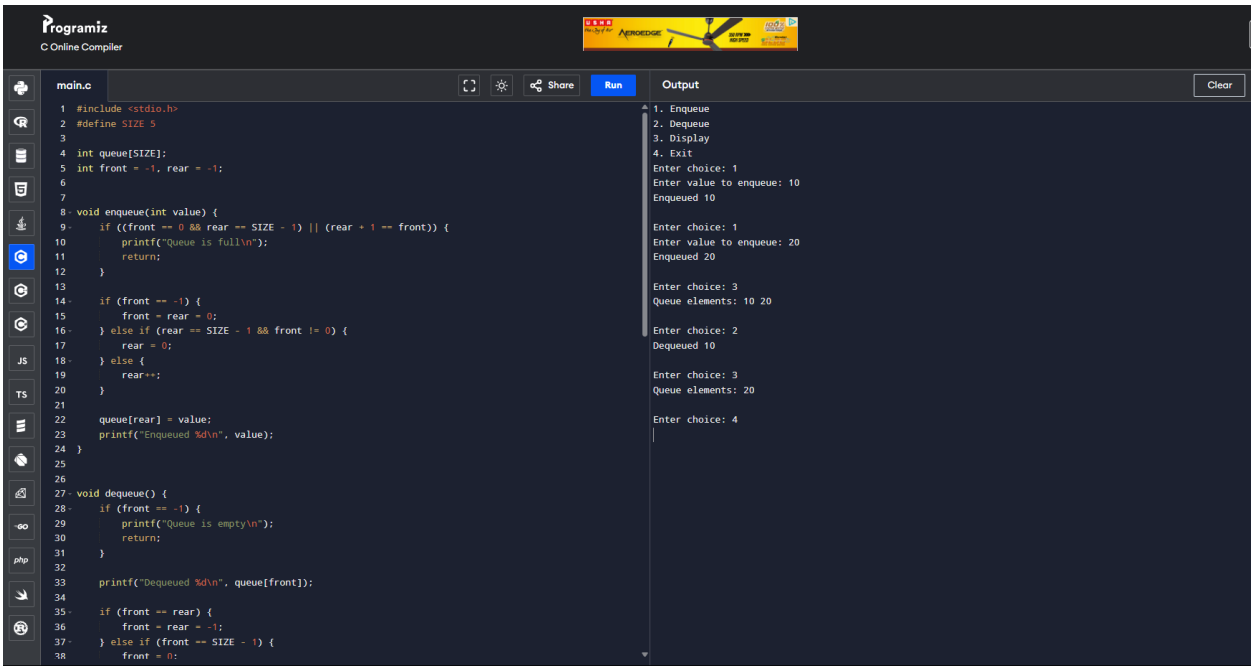


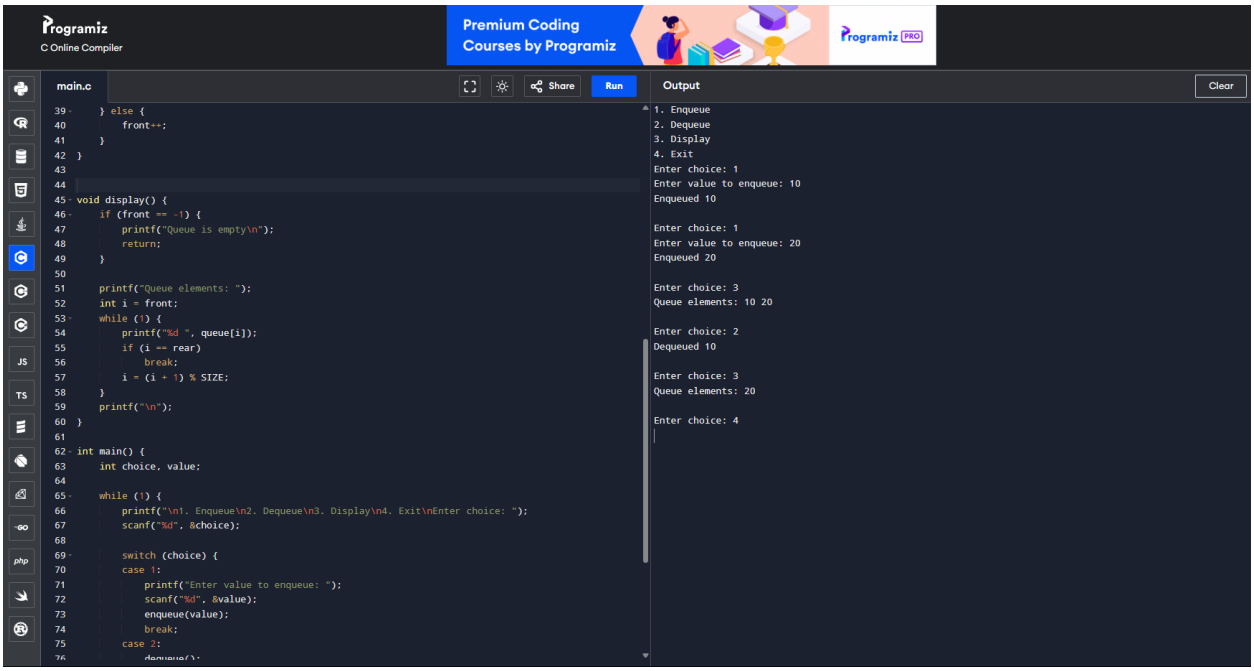
```c
    struct Node* temp = front;
    int value = temp->data;
    front = front->next;
    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
    return value;
}


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

int main() {
    int choice, value;

    while (1) {
        printf("\n1. Enqueue\n2. Dequeue\n3. Display Queue\n4. Exit\nEnter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
```

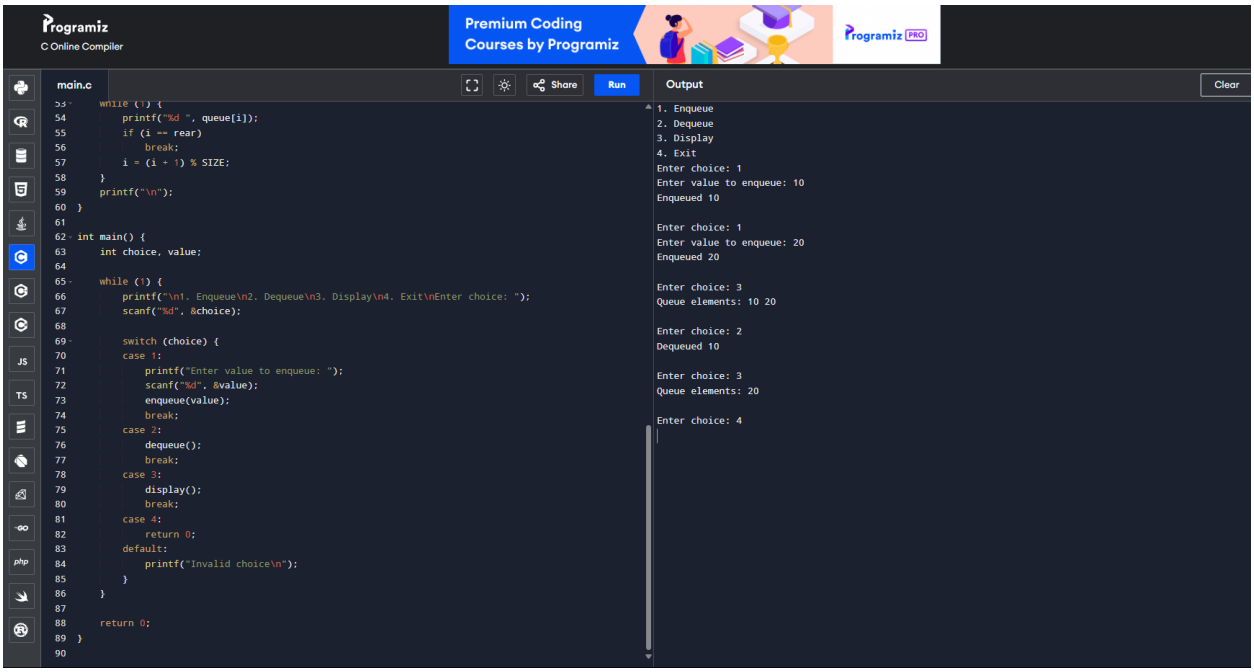
```
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 20

Enter choice: 2
Dequeued 20

Enter choice: 3
Queue is empty

Enter choice: 4
```


Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
int main() {
int choice, value;
while (1) {
printf("\n1. Enqueue\n2. Dequeue\n3. Display Queue\n4. Exit\nEnter choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter value to enqueue: ");
scanf("%d", &value);
enqueue(value);
break;
case 2:
value = dequeue();
if (value != -1) {
printf("Dequeued %d\n", value);
}
break;
case 3:
display();
break;
case 4:
return 0;
default:
printf("Invalid choice\n");
}
}
return 0;
}
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10
Enter choice: 1
Enter value to enqueue: 20
Enqueued 20
Enter choice: 3
Queue elements: 10 20
Enter choice: 2
Dequeued 10
Enter choice: 3
Queue elements: 20
Enter choice: 2
Dequeued 20
Enter choice: 3
Queue is empty
Enter choice: 4

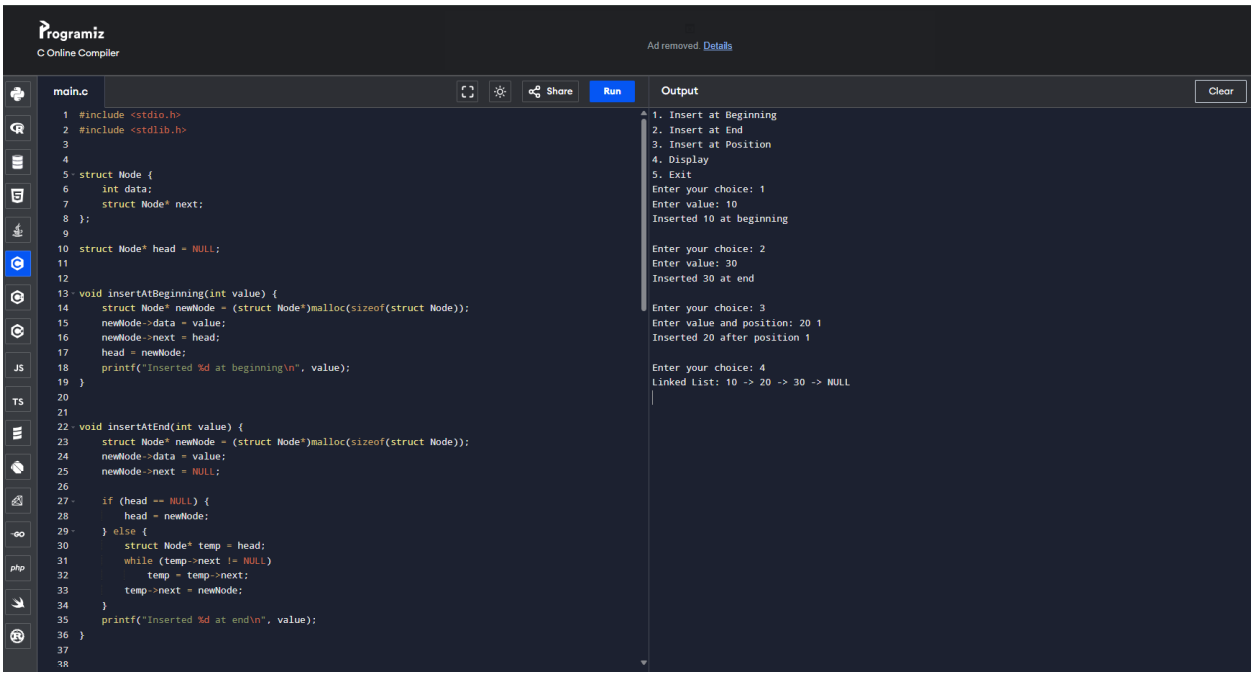
12.Write a program in C to implement operations on circular queue using array






Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
while (1) {
printf("%d ", queue[i]);
if (i == rear)
break;
i = (i + 1) % SIZE;
}
printf("\n");
}
int main() {
int choice, value;
while (1) {
printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nEnter choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter value to enqueue: ");
scanf("%d", &value);
enqueue(value);
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 4:
return 0;
default:
printf("Invalid choice\n");
}
}
return 0;
}
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10
Enter choice: 1
Enter value to enqueue: 20
Enqueued 20
Enter choice: 3
Queue elements: 10 20
Enter choice: 2
Dequeued 10
Enter choice: 3
Queue elements: 20
Enter choice: 4

13.Write a program in C to implement insertion in a linked list(beg; mid; end)



Programiz
C Online Compiler

Ad removed. Details

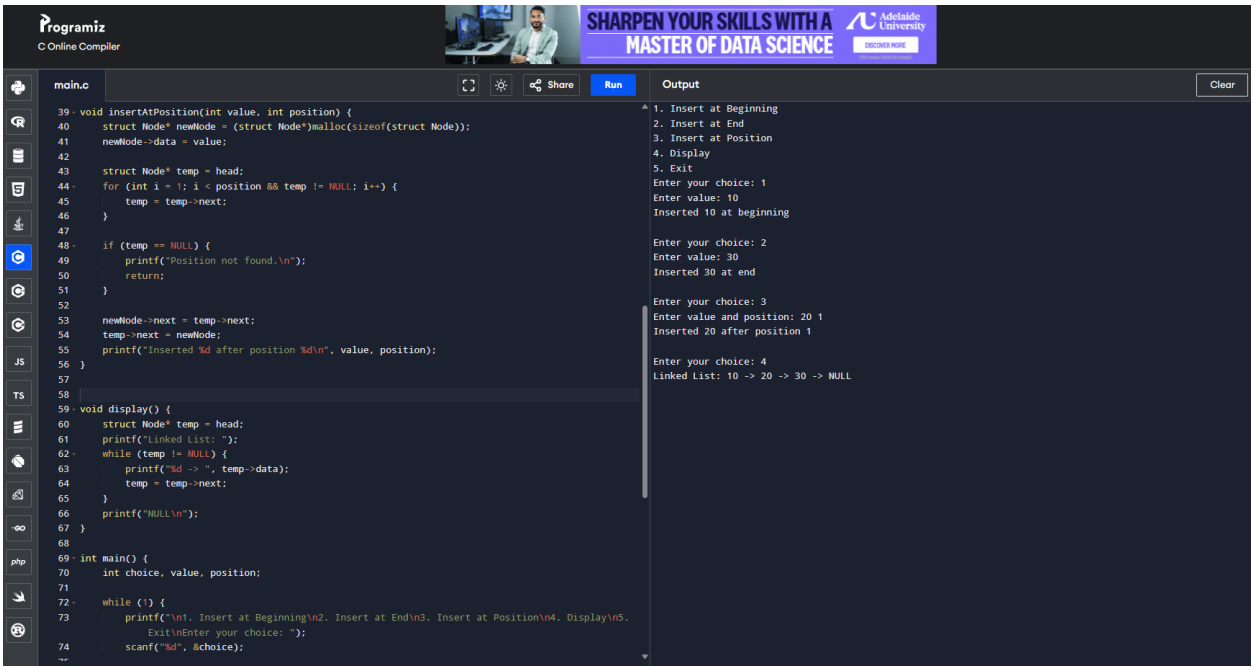
main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>


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

struct Node* head = NULL;


void insertAtBeginning(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
    printf("Inserted %d at beginning\n", value);
}


void insertAtEnd(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 at end\n", value);
}

```

Output Clear

```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> NULL
```



Programiz
C Online Compiler

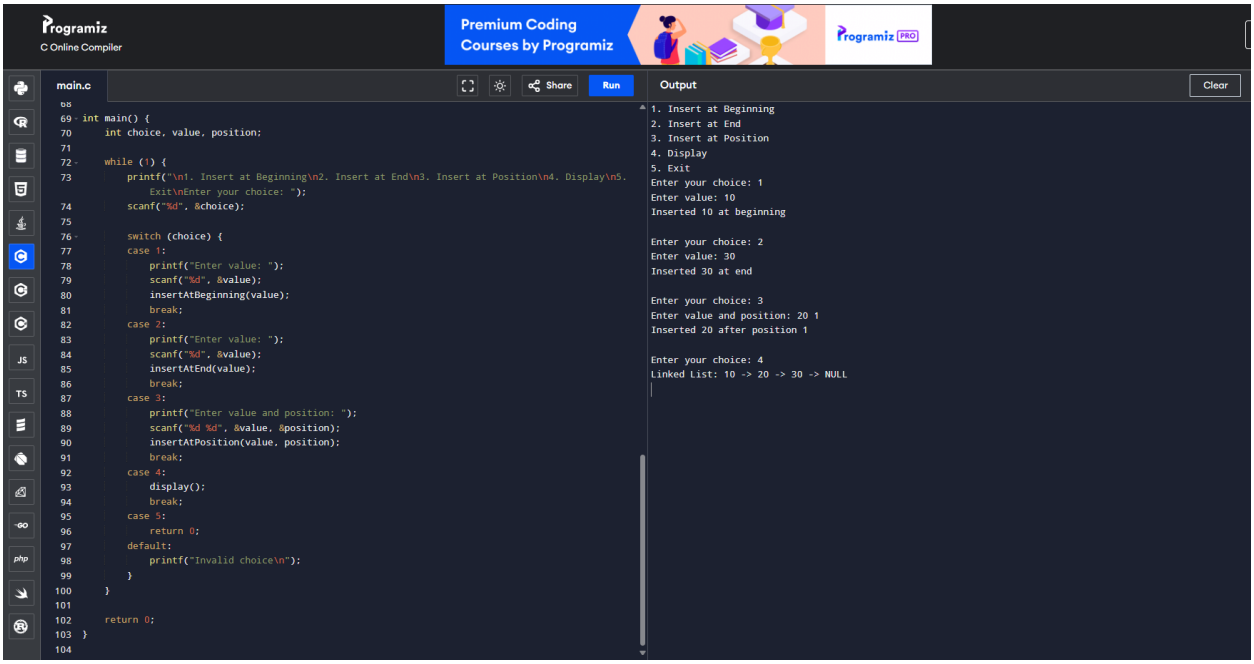
main.c

Share Run

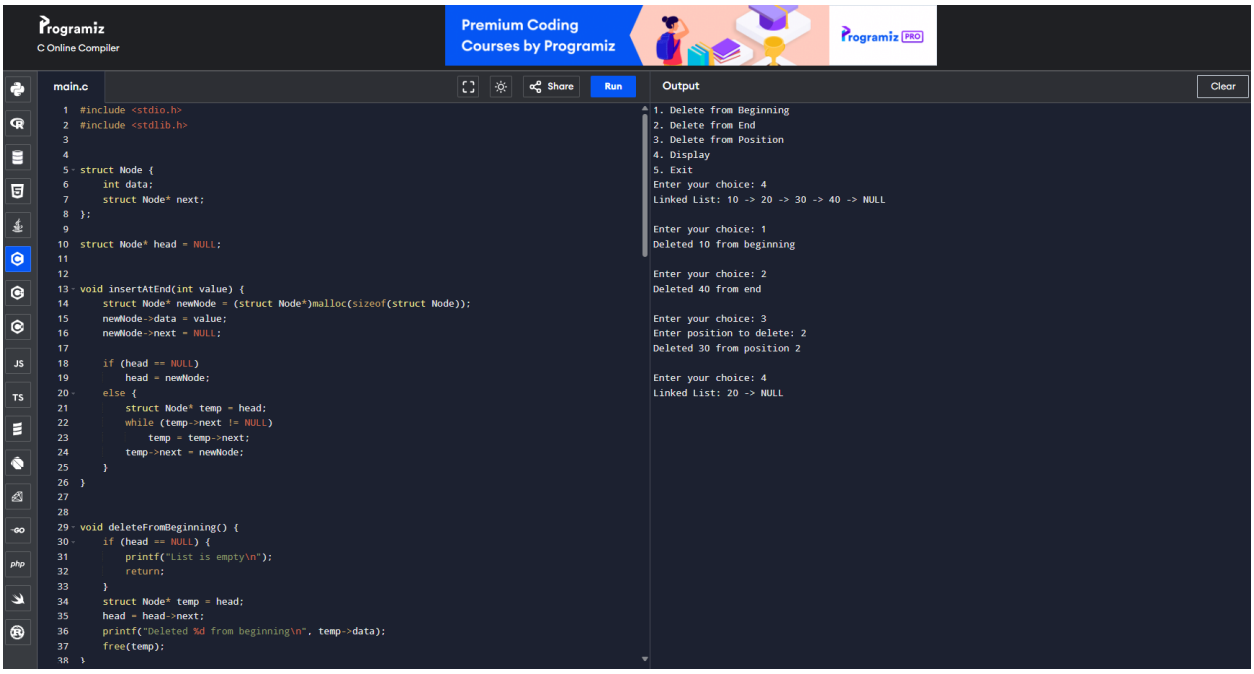
```c
void insertAtPosition(int value, int position) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;

    struct Node* temp = head;
    for (int i = 1; i < position && temp != NULL; i++) {
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Position not found.\n");
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
    printf("Inserted %d after position %d\n", value, position);
}


void display() {
    struct Node* temp = head;
    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice, value, position;

    while (1) {
        printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5.
            Exit\nEnter your choice: ");
        scanf("%d", &choice);
```

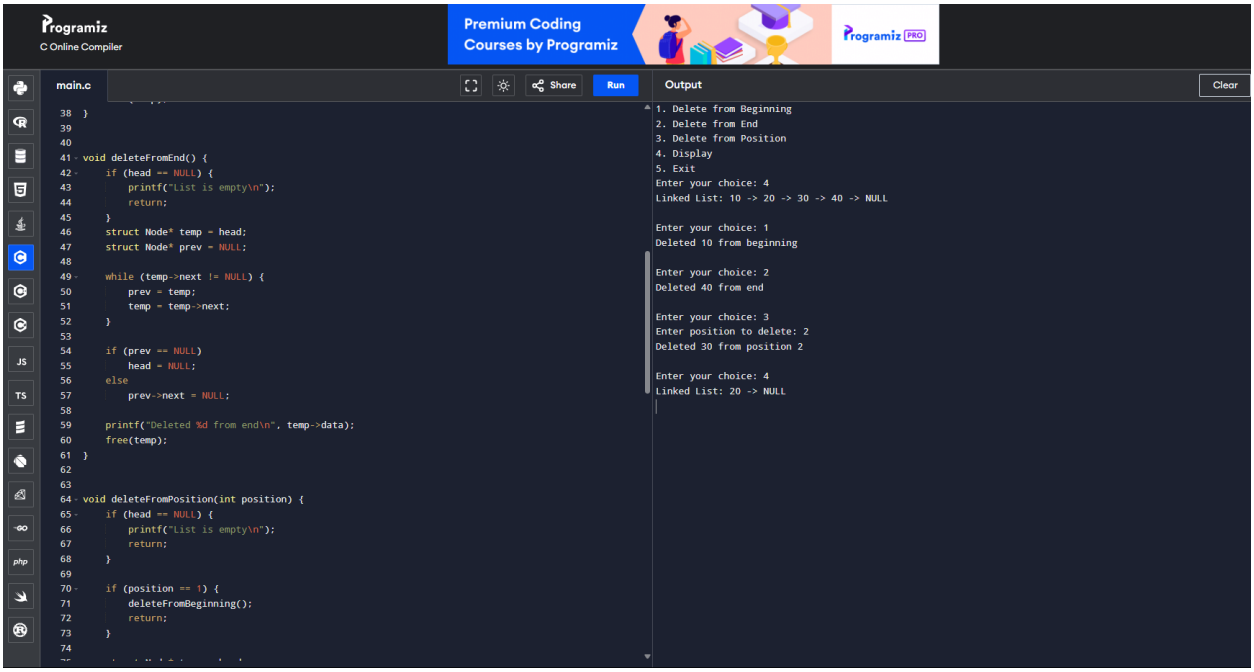
Output Clear

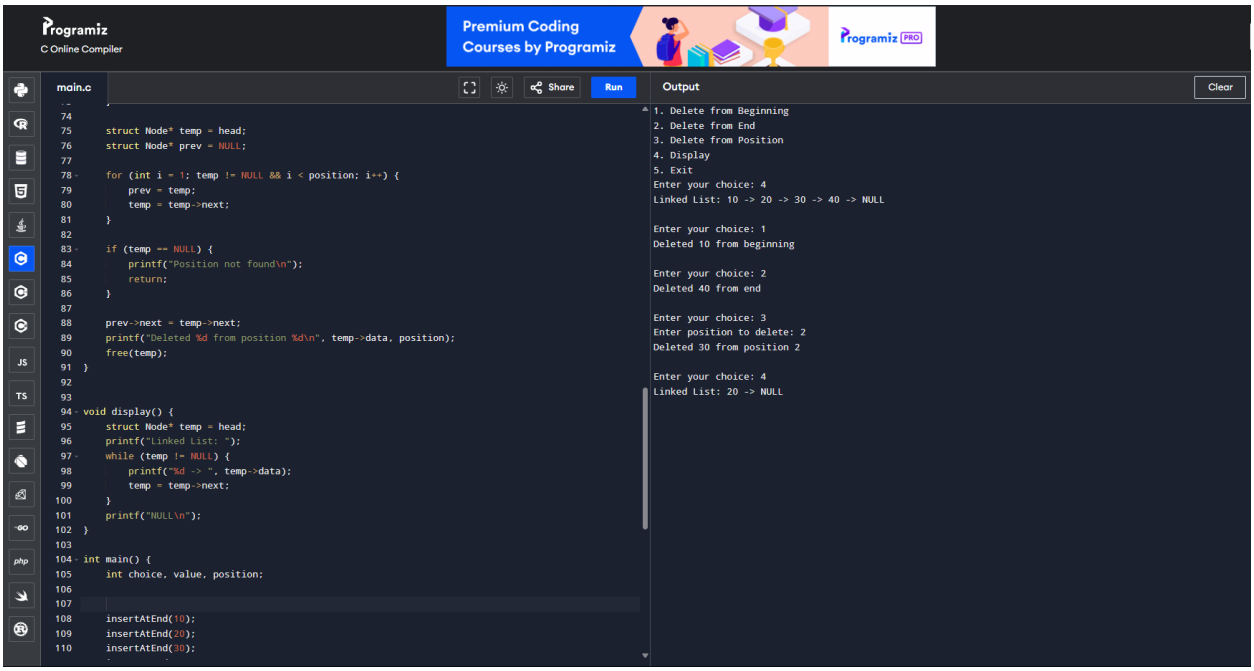
```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> NULL
```

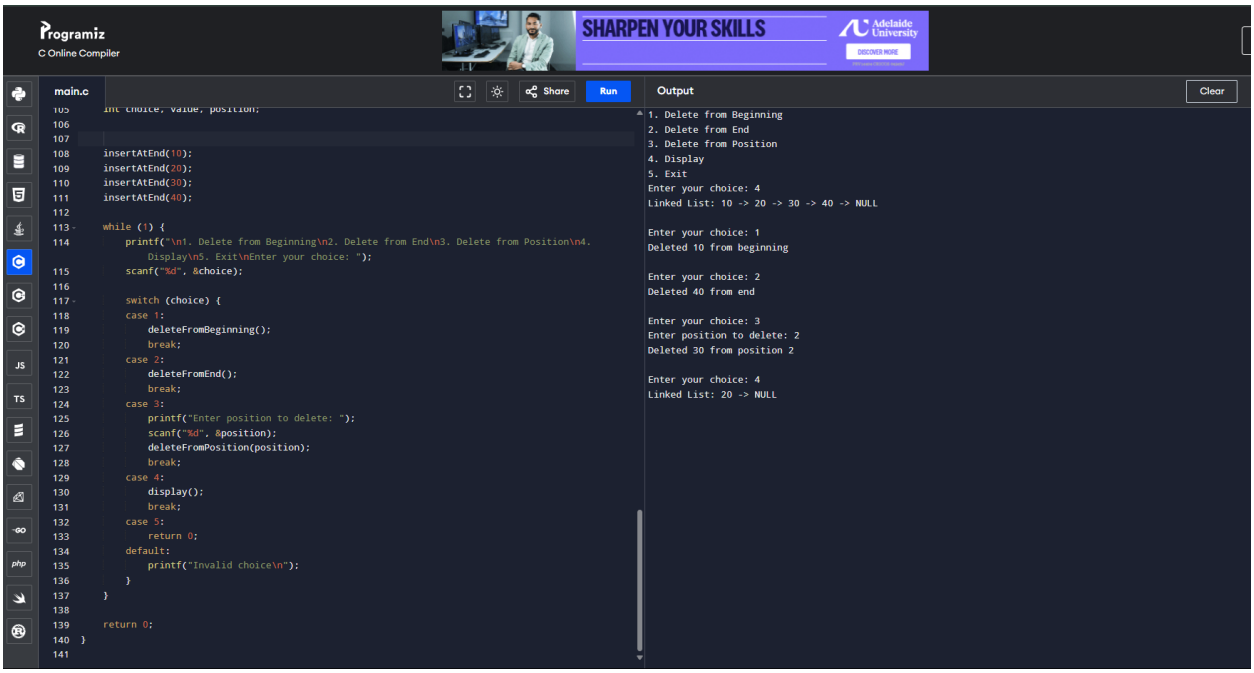

Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
int main() {
int choice, value, position;
while (1) {
printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5. Exit\nEnter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter value: ");
scanf("%d", &value);
insertAtBeginning(value);
break;
case 2:
printf("Enter value: ");
scanf("%d", &value);
insertAtEnd(value);
break;
case 3:
printf("Enter value and position: ");
scanf("%d %d", &value, &position);
insertAtPosition(value, position);
break;
case 4:
display();
break;
case 5:
return 0;
default:
printf("Invalid choice\n");
}
}
return 0;
}
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning
Enter your choice: 2
Enter value: 30
Inserted 30 at end
Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1
Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> NULL

14.Write a program in C to implement deletion from a linked list(beg; mid; end).

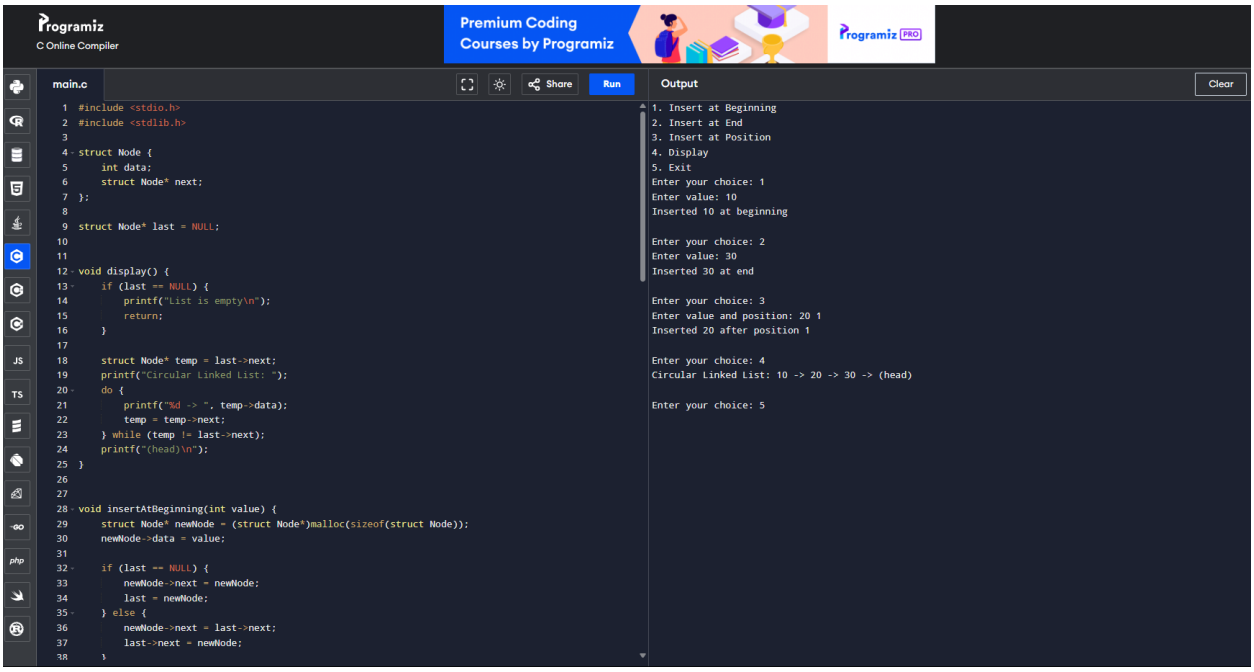


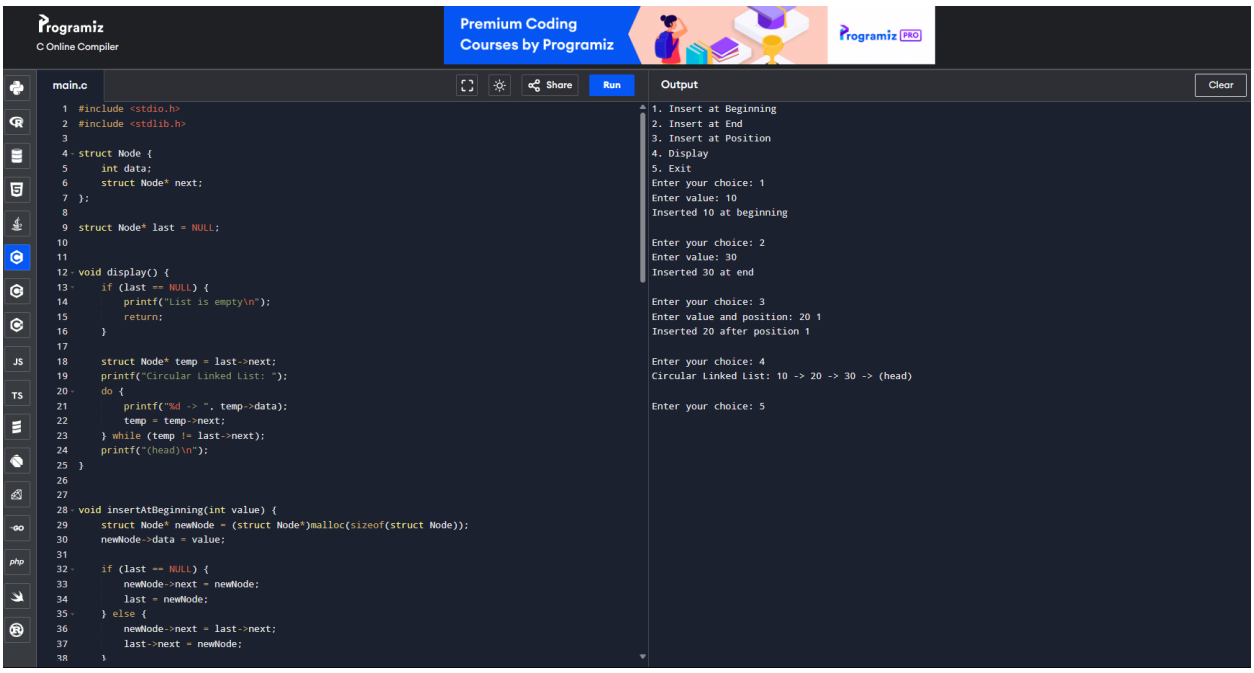


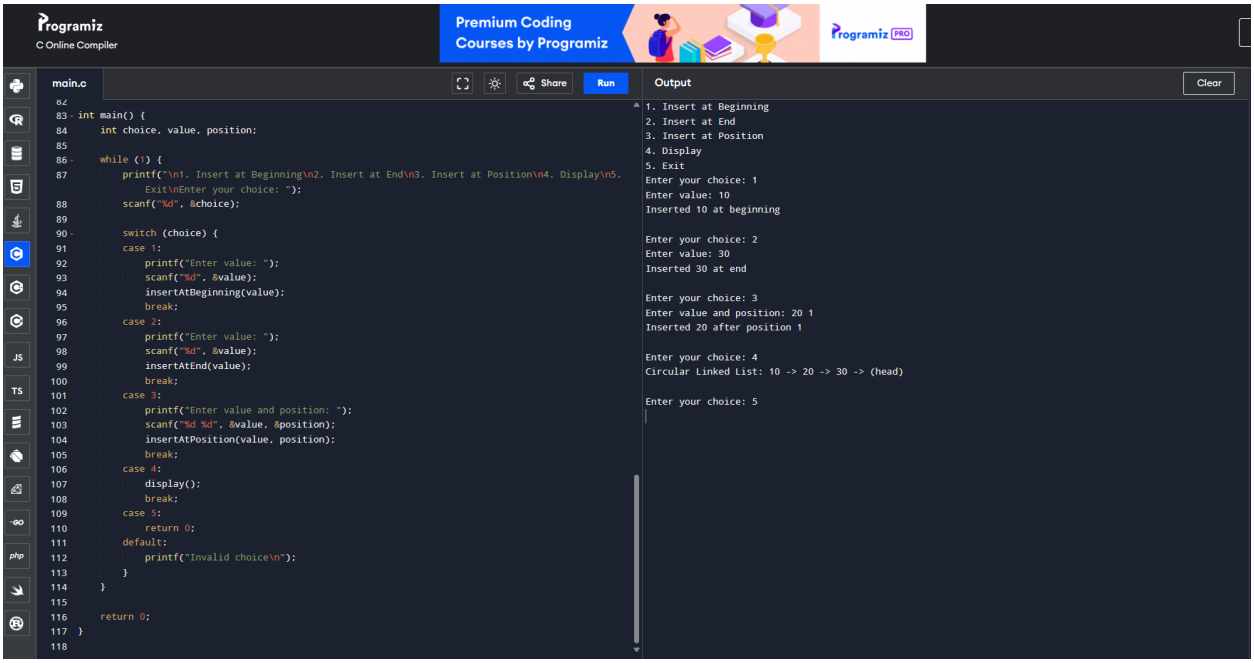

Programiz
C Online Compiler
Premium Coding Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
struct Node* temp = head;
struct Node* prev = NULL;
for (int i = 1; temp != NULL && i < position; i++) {
prev = temp;
temp = temp->next;
}
if (temp == NULL) {
printf("Position not found\n");
return;
}
prev->next = temp->next;
printf("Deleted %d from position %d\n", temp->data, position);
free(temp);
}
void display() {
struct Node* temp = head;
printf("Linked List: ");
while (temp != NULL) {
printf("%d -> ", temp->data);
temp = temp->next;
}
printf("NULL\n");
}
int main() {
int choice, value, position;
insertAtEnd(10);
insertAtEnd(20);
insertAtEnd(30);
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> 40 -> NULL
Enter your choice: 1
Deleted 10 from beginning
Enter your choice: 2
Deleted 40 from end
Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2
Enter your choice: 4
Linked List: 20 -> NULL


Programiz
C Online Compiler
SHARPEN YOUR SKILLS
Adelaide University
DISCOVER MORE
main.c
Share
Run
Output
Clear
insertAtEnd(10);
insertAtEnd(20);
insertAtEnd(30);
insertAtEnd(40);
while (1) {
printf("\n1. Delete from Beginning\n2. Delete from End\n3. Delete from Position\n4.
Display\n5. Exit\nEnter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
deleteFromBeginning();
break;
case 2:
deleteFromEnd();
break;
case 3:
printf("Enter position to delete: ");
scanf("%d", &position);
deleteFromPosition(position);
break;
case 4:
display();
break;
case 5:
return 0;
default:
printf("Invalid choice\n");
}
}
return 0;
}
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> 40 -> NULL
Enter your choice: 1
Deleted 10 from beginning
Enter your choice: 2
Deleted 40 from end
Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2
Enter your choice: 4
Linked List: 20 -> NULL

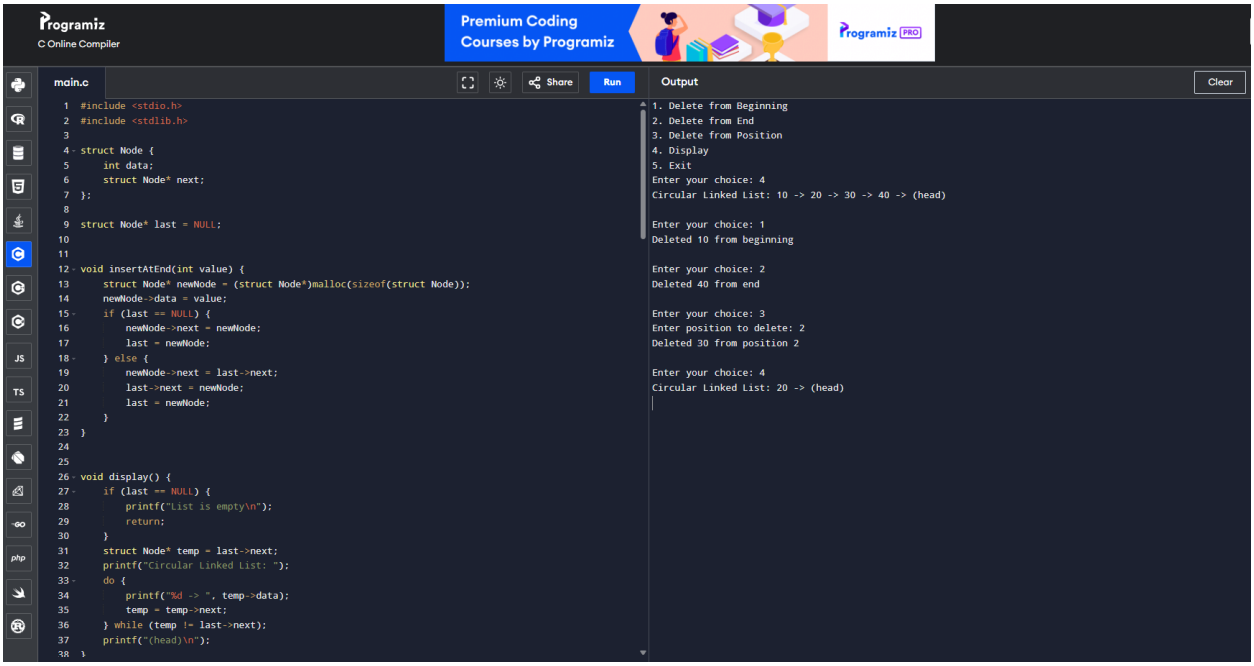
15.Write a program in C to implement insertion in a circular linked list(beg; mid; end)

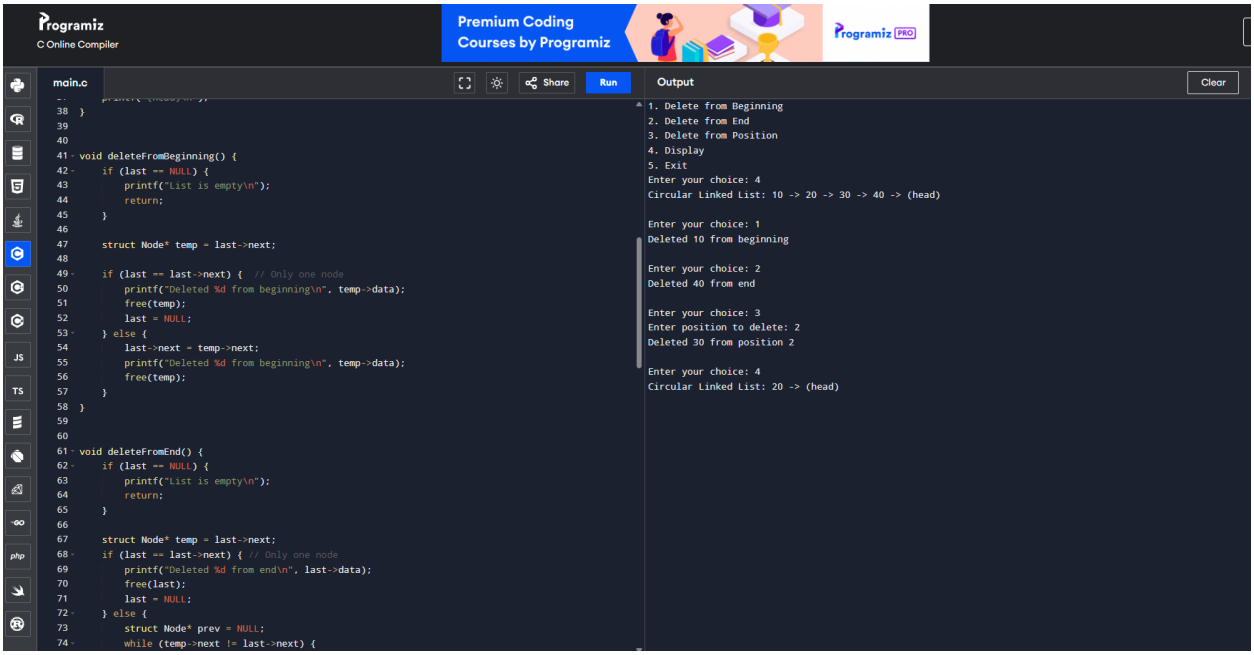


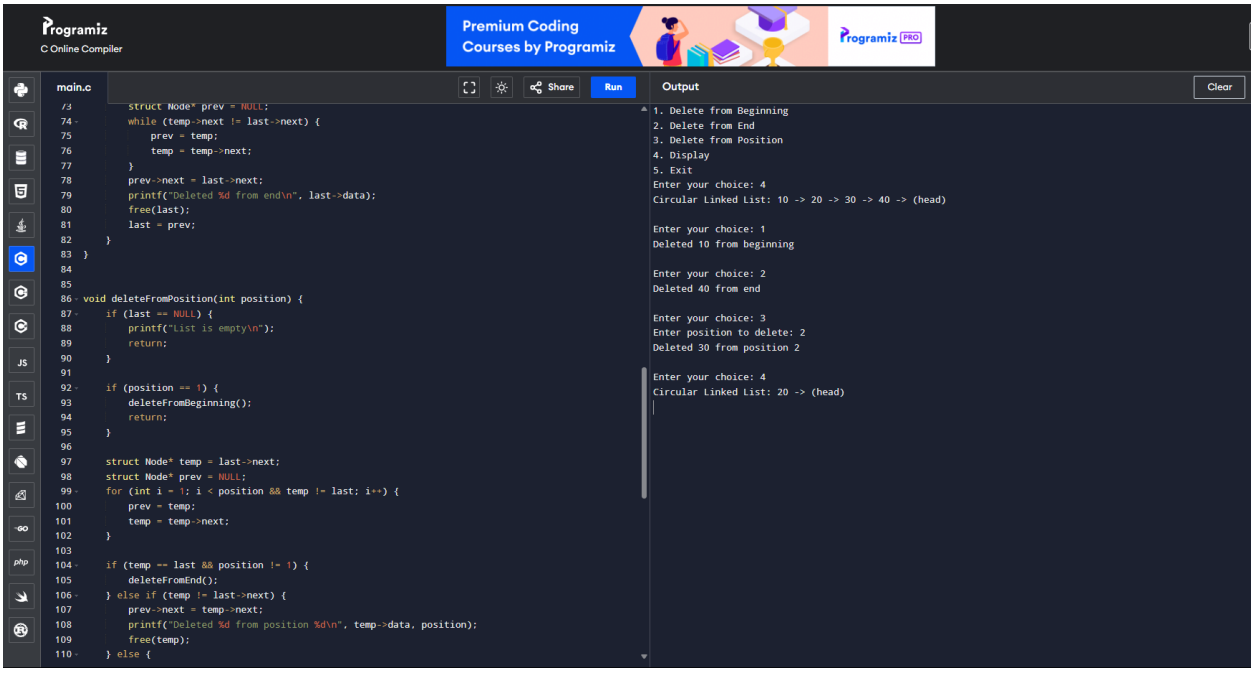


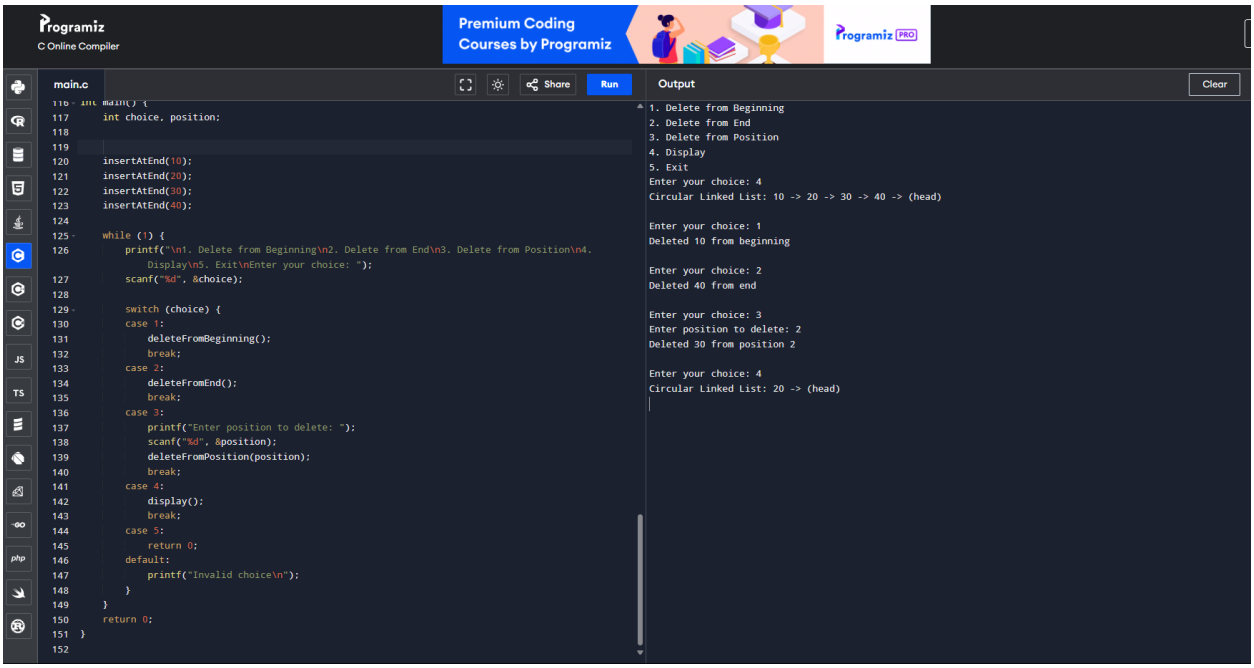

Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
int main() {
int choice, value, position;
while (1) {
printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5. Exit\nEnter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter value: ");
scanf("%d", &value);
insertAtBeginning(value);
break;
case 2:
printf("Enter value: ");
scanf("%d", &value);
insertAtEnd(value);
break;
case 3:
printf("Enter value and position: ");
scanf("%d %d", &value, &position);
insertAtPosition(value, position);
break;
case 4:
display();
break;
case 5:
return 0;
default:
printf("Invalid choice\n");
}
}
return 0;
}
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning
Enter your choice: 2
Enter value: 30
Inserted 30 at end
Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1
Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> (head)
Enter your choice: 5

16.Write a program in C to implement deletion from a circular linked list(beg; mid; end) queue.

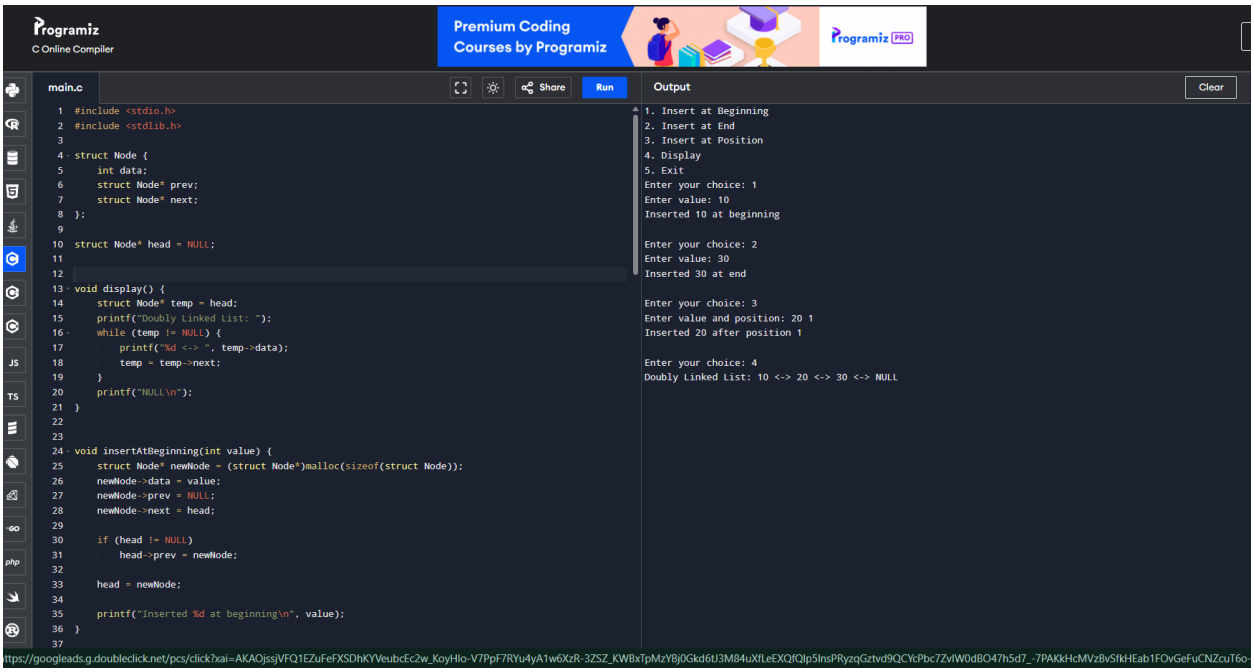


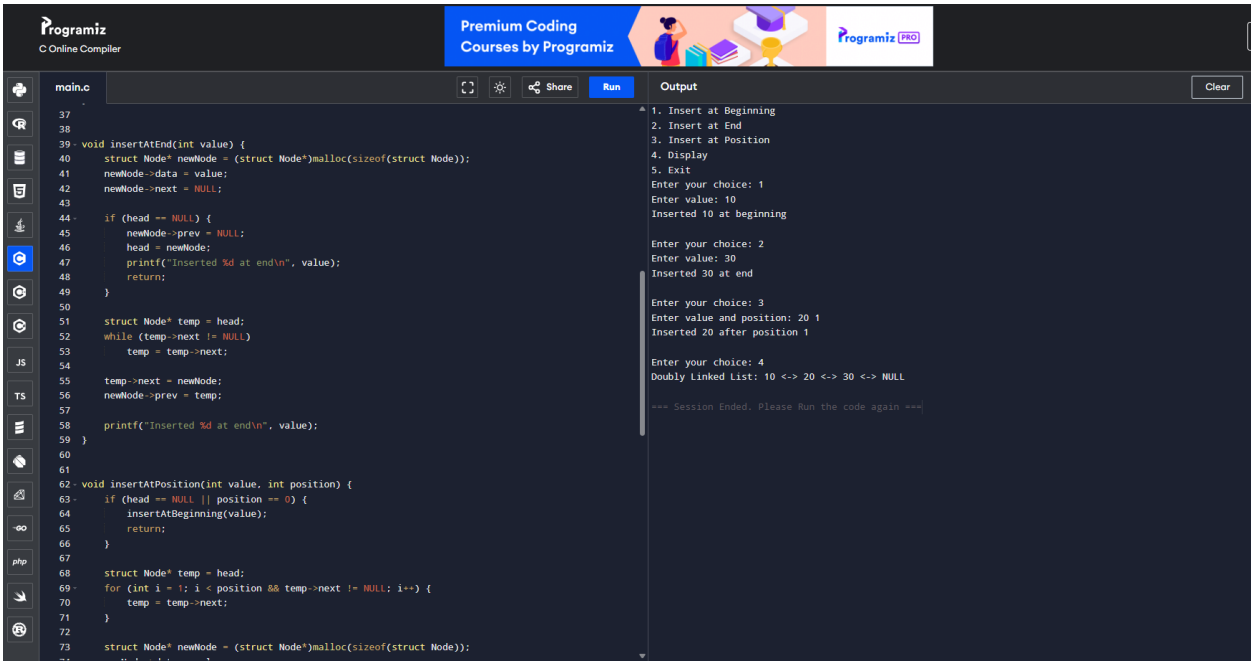


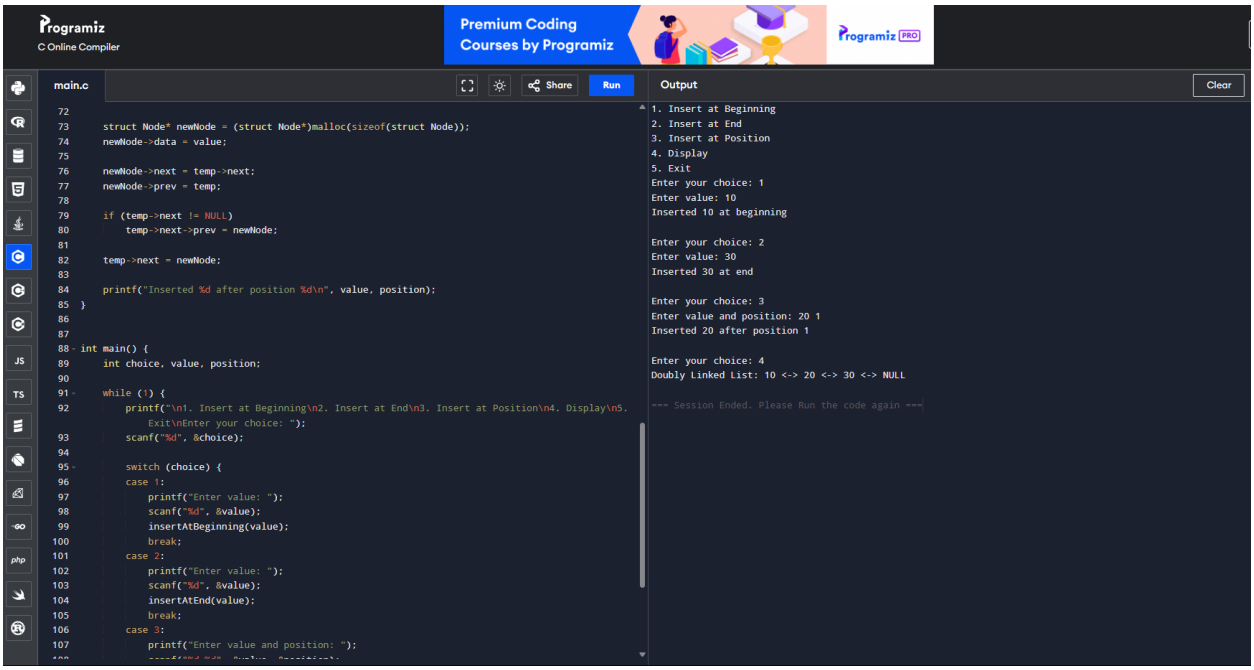

Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
struct Node* prev = NULL;
while (temp->next != last->next) {
prev = temp;
temp = temp->next;
}
prev->next = last->next;
printf("Deleted %d from end\n", last->data);
free(last);
last = prev;
}
}
void deleteFromPosition(int position) {
if (last == NULL) {
printf("List is empty\n");
return;
}
if (position == 1) {
deleteFromBeginning();
return;
}
struct Node* temp = last->next;
struct Node* prev = NULL;
for (int i = 1; i < position && temp != last; i++) {
prev = temp;
temp = temp->next;
}
if (temp == last && position != 1) {
deleteFromEnd();
} else if (temp != last->next) {
prev->next = temp->next;
printf("Deleted %d from position %d\n", temp->data, position);
free(temp);
} else {
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> 40 -> (head)
Enter your choice: 1
Deleted 10 from beginning
Enter your choice: 2
Deleted 40 from end
Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2
Enter your choice: 4
Circular Linked List: 20 -> (head)


Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
int main() {
int choice, position;
insertAtEnd(10);
insertAtEnd(20);
insertAtEnd(30);
insertAtEnd(40);
while (1) {
printf("\n1. Delete from Beginning\n2. Delete from End\n3. Delete from Position\n4.
Display\n5. Exit\nEnter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
deleteFromBeginning();
break;
case 2:
deleteFromEnd();
break;
case 3:
printf("Enter position to delete: ");
scanf("%d", &position);
deleteFromPosition(position);
break;
case 4:
display();
break;
case 5:
return 0;
default:
printf("Invalid choice\n");
}
}
return 0;
}
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> 40 -> (head)
Enter your choice: 1
Deleted 10 from beginning
Enter your choice: 2
Deleted 40 from end
Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2
Enter your choice: 4
Circular Linked List: 20 -> (head)

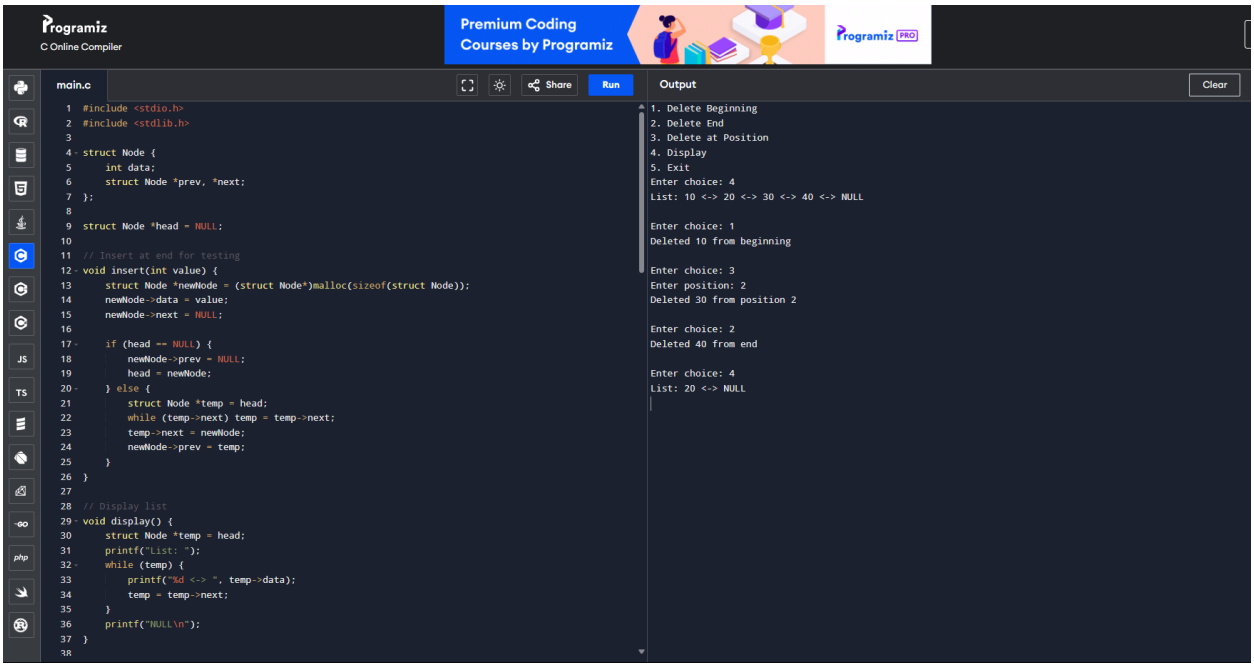
17.Write a program in C to implement insertion in a doubly linked list(beg; mid; end)





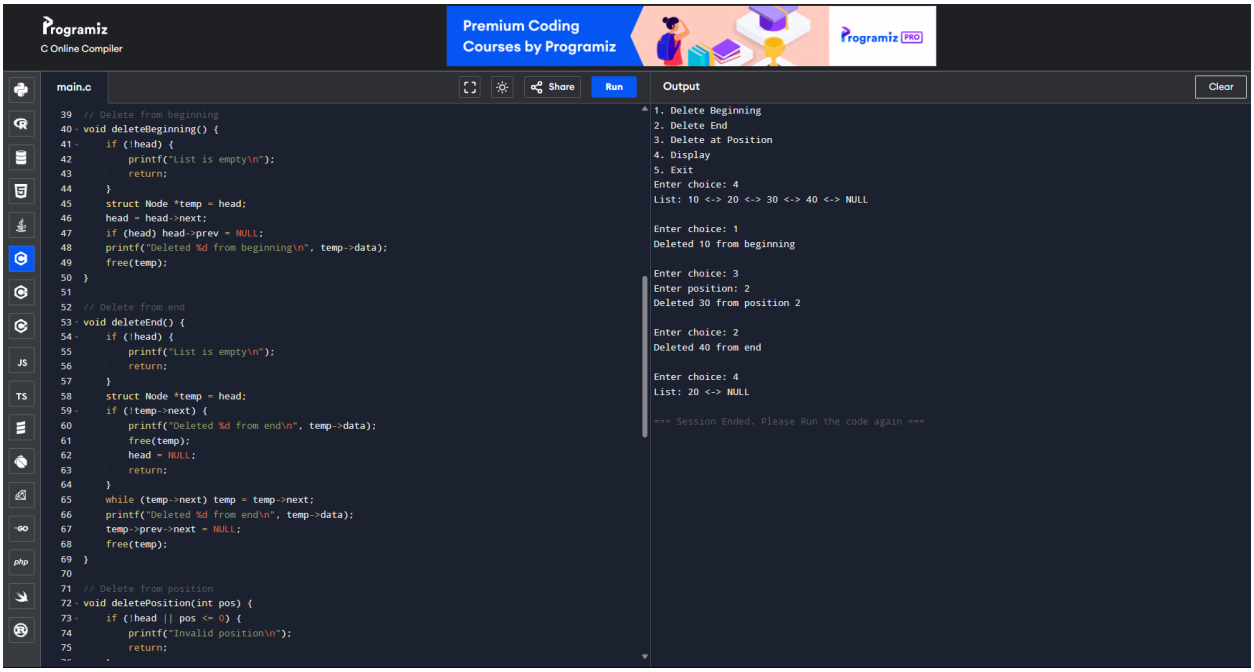

Programiz
C Online Compiler
Premium Coding Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
}
int main() {
int choice, value, position;
while (1) {
printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5. Exit\nEnter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter value: ");
scanf("%d", &value);
insertAtBeginning(value);
break;
case 2:
printf("Enter value: ");
scanf("%d", &value);
insertAtEnd(value);
break;
case 3:
printf("Enter value and position: ");
scanf("%d %d", &value, &position);
insertAtPosition(value, position);
break;
case 4:
display();
break;
case 5:
return 0;
default:
printf("Invalid choice\n");
}
}
}
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning
Enter your choice: 2
Enter value: 30
Inserted 30 at end
Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1
Enter your choice: 4
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL
=== Session Ended. Please Run the code again ===

18.Write a program in C to implement deletion from a doubly linked list(beg; mid; end)



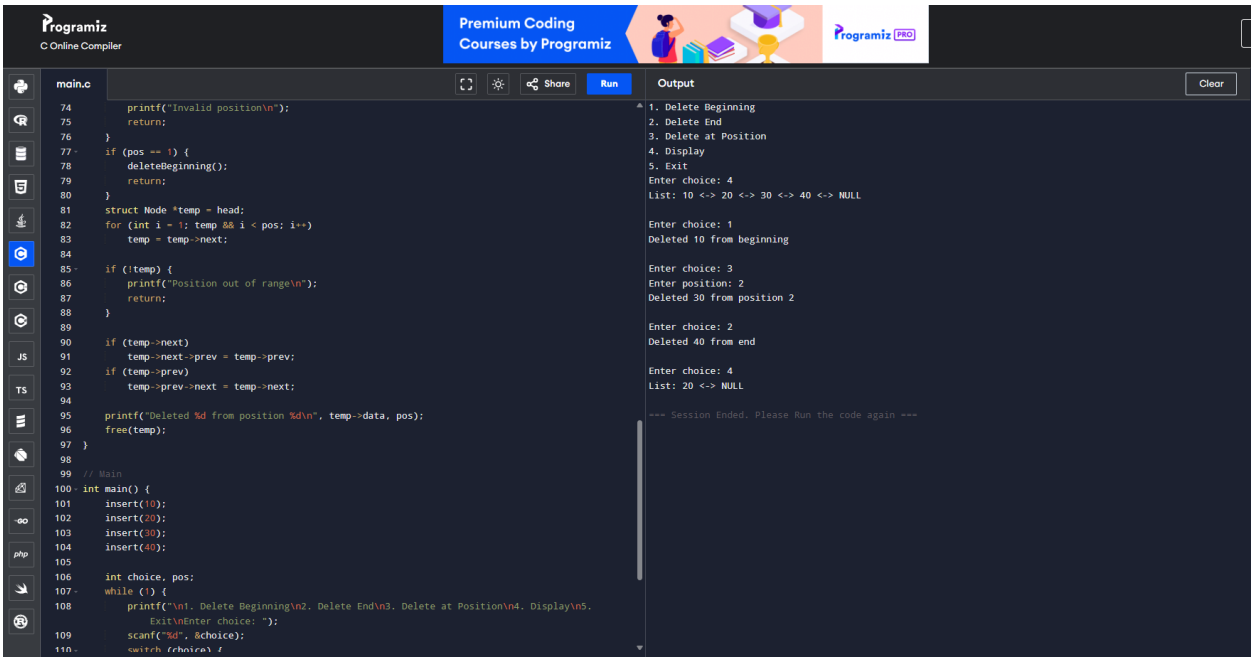
```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node *prev, *next;
};

struct Node *head = NULL;

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

    if (head == NULL) {
        newNode->prev = NULL;
        head = newNode;
    } else {
        struct Node *temp = head;
        while (temp->next) temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
    }
}

// Display list
void display() {
    struct Node *temp = head;
    printf("List: ");
    while (temp) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```

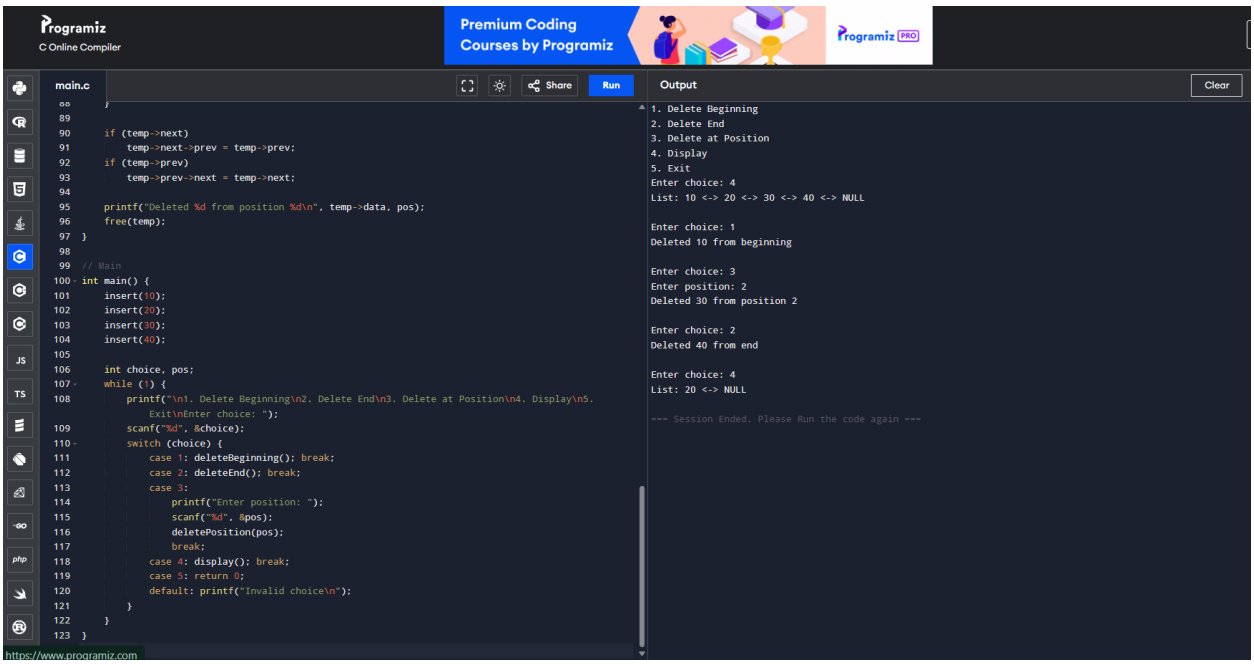
```
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL

Enter choice: 1
Deleted 10 from beginning

Enter choice: 3
Enter position: 2
Deleted 30 from position 2

Enter choice: 2
Deleted 40 from end

Enter choice: 4
List: 20 <-> NULL
```



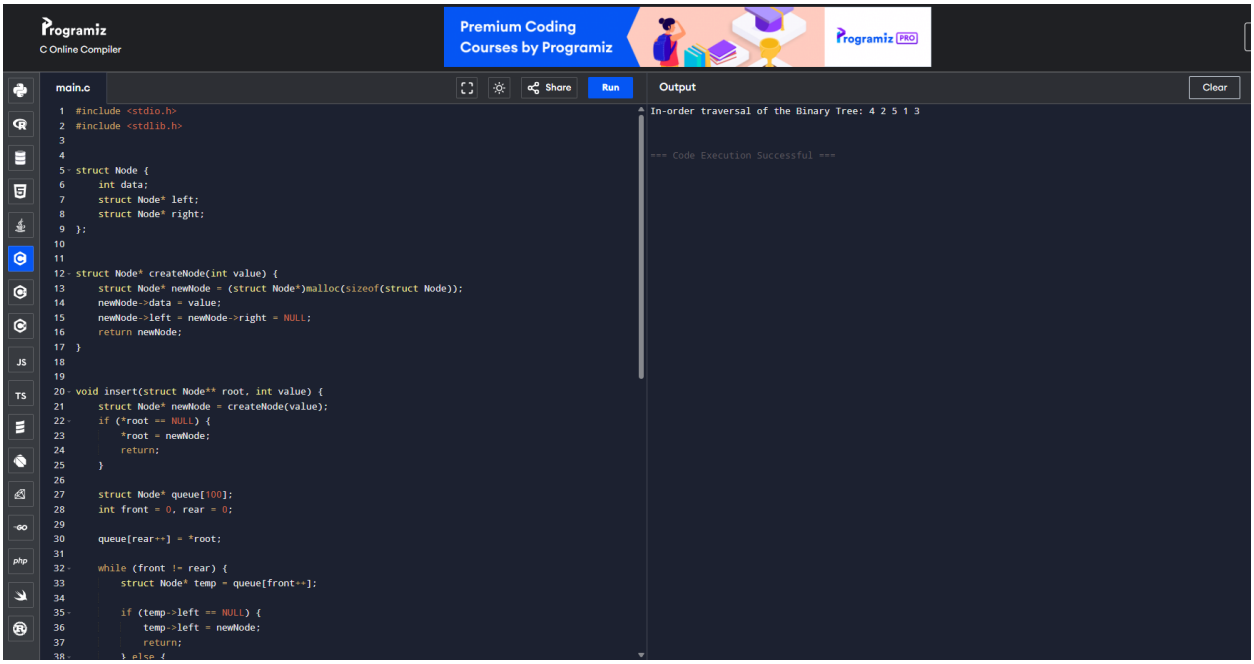
```c
// Delete from beginning
void deleteBeginning() {
    if (!head) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = head;
    head = head->next;
    if (head) head->prev = NULL;
    printf("Deleted %d from beginning\n", temp->data);
    free(temp);
}

// Delete from end
void deleteEnd() {
    if (!head) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = head;
    if (!temp->next) {
        printf("Deleted %d from end\n", temp->data);
        free(temp);
        head = NULL;
        return;
    }
    while (temp->next) temp = temp->next;
    printf("Deleted %d from end\n", temp->data);
    temp->prev->next = NULL;
    free(temp);
}

// Delete from position
void deletePosition(int pos) {
    if (!head || pos <= 0) {
        printf("Invalid position\n");
        return;
```

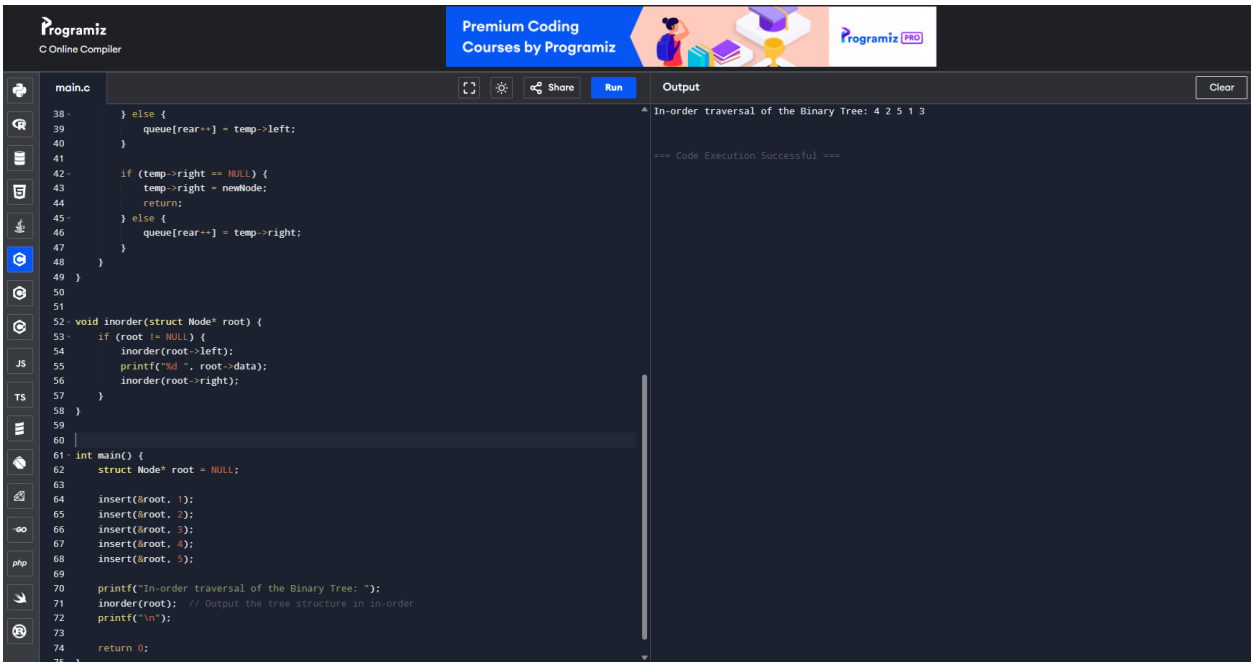
```
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL

Enter choice: 1
Deleted 10 from beginning

Enter choice: 3
Enter position: 2
Deleted 30 from position 2

Enter choice: 2
Deleted 40 from end

Enter choice: 4
List: 20 <-> NULL

=== Session Ended. Please Run the code again ===
```


Programiz
C Online Compiler
Premium Coding Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
printf("Invalid position\n");
return;
}
if (pos == 1) {
deleteBeginning();
return;
}
struct Node *temp = head;
for (int i = 1; temp && i < pos; i++)
temp = temp->next;
if (!temp) {
printf("Position out of range\n");
return;
}
if (temp->next)
temp->next->prev = temp->prev;
if (temp->prev)
temp->prev->next = temp->next;
printf("Deleted %d from position %d\n", temp->data, pos);
free(temp);
}
// Main
int main() {
insert(10);
insert(20);
insert(30);
insert(40);
int choice, pos;
while (1) {
printf("\n1. Delete Beginning\n2. Delete End\n3. Delete at Position\n4. Display\n5.
Exit\nEnter choice: ");
scanf("%d", &choice);
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL
Enter choice: 1
Deleted 10 from beginning
Enter choice: 3
Enter position: 2
Deleted 30 from position 2
Enter choice: 2
Deleted 40 from end
Enter choice: 4
List: 20 <-> NULL
=== Session Ended. Please Run the code again ===

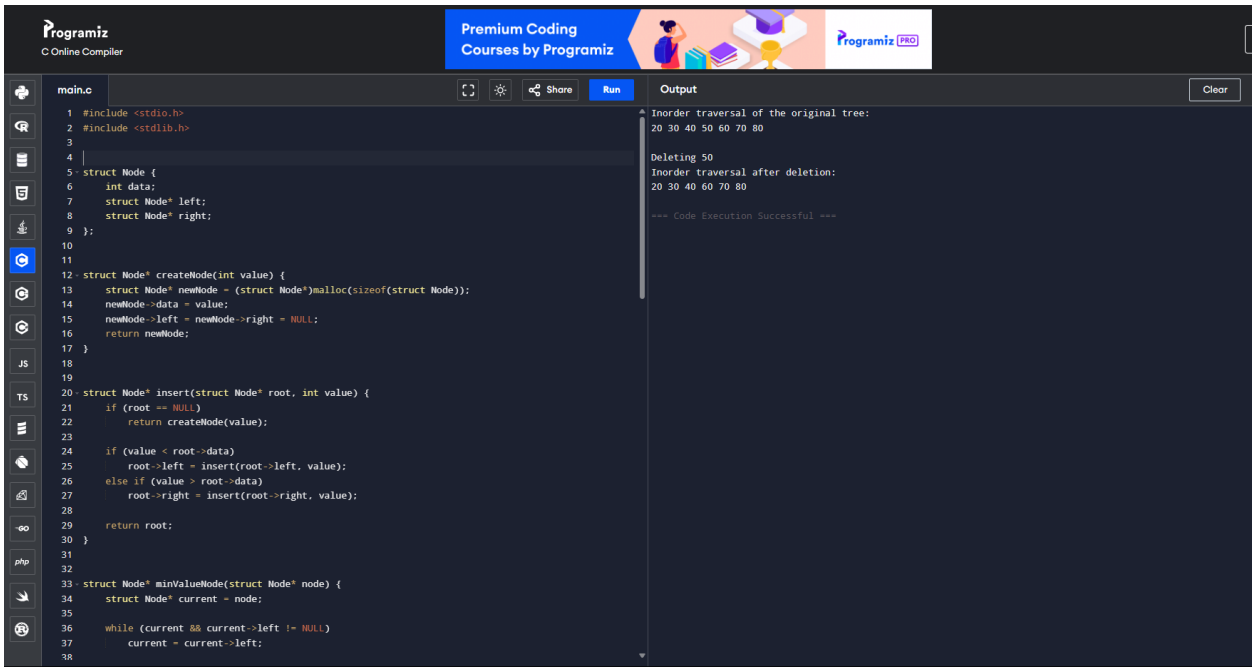

Programiz
C Online Compiler
Premium Coding Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
}
if (temp->next)
temp->next->prev = temp->prev;
if (temp->prev)
temp->prev->next = temp->next;
printf("Deleted %d from position %d\n", temp->data, pos);
free(temp);
}
// Main
int main() {
insert(10);
insert(20);
insert(30);
insert(40);
int choice, pos;
while (1) {
printf("\n1. Delete Beginning\n2. Delete End\n3. Delete at Position\n4. Display\n5.
Exit\nEnter choice: ");
scanf("%d", &choice);
switch (choice) {
case 1: deleteBeginning(); break;
case 2: deleteEnd(); break;
case 3:
printf("Enter position: ");
scanf("%d", &pos);
deletePosition(pos);
break;
case 4: display(); break;
case 5: return 0;
default: printf("Invalid choice\n");
}
}
}
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL
Enter choice: 1
Deleted 10 from beginning
Enter choice: 3
Enter position: 2
Deleted 30 from position 2
Enter choice: 2
Deleted 40 from end
Enter choice: 4
List: 20 <-> NULL
=== Session Ended. Please Run the code again ===
https://www.programiz.com

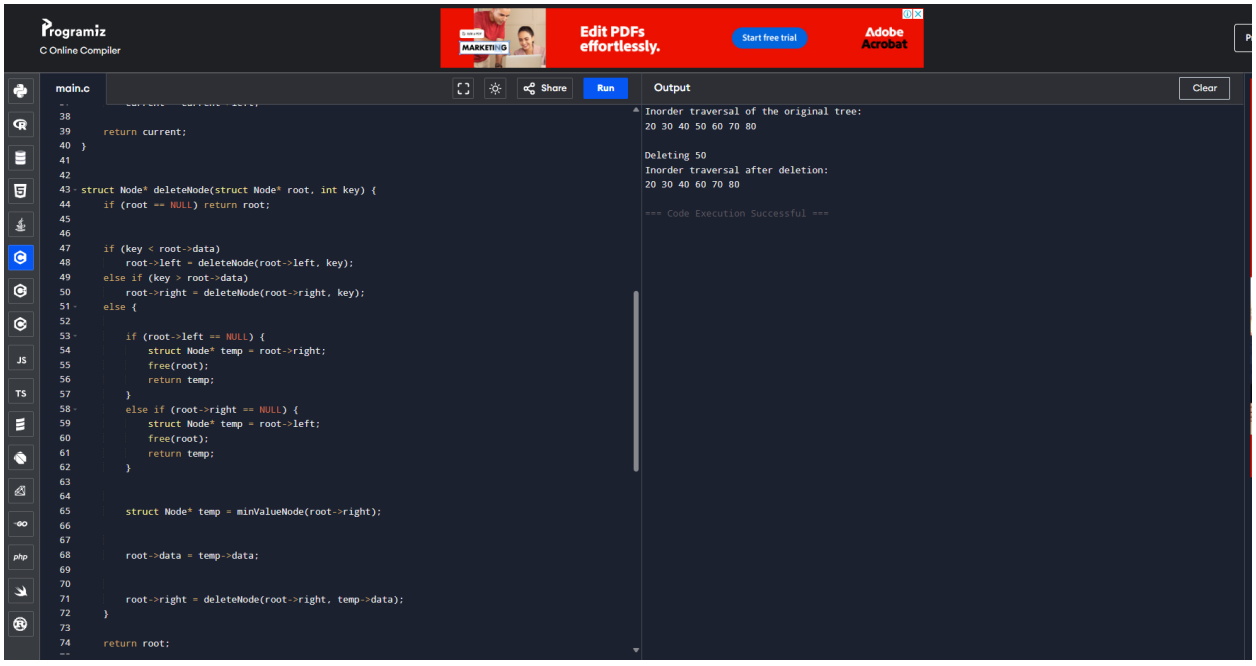
19. Write a program in C to implement insertion in Binary tree 20. Write a program in C to implement deletion from Binary tree simple tree

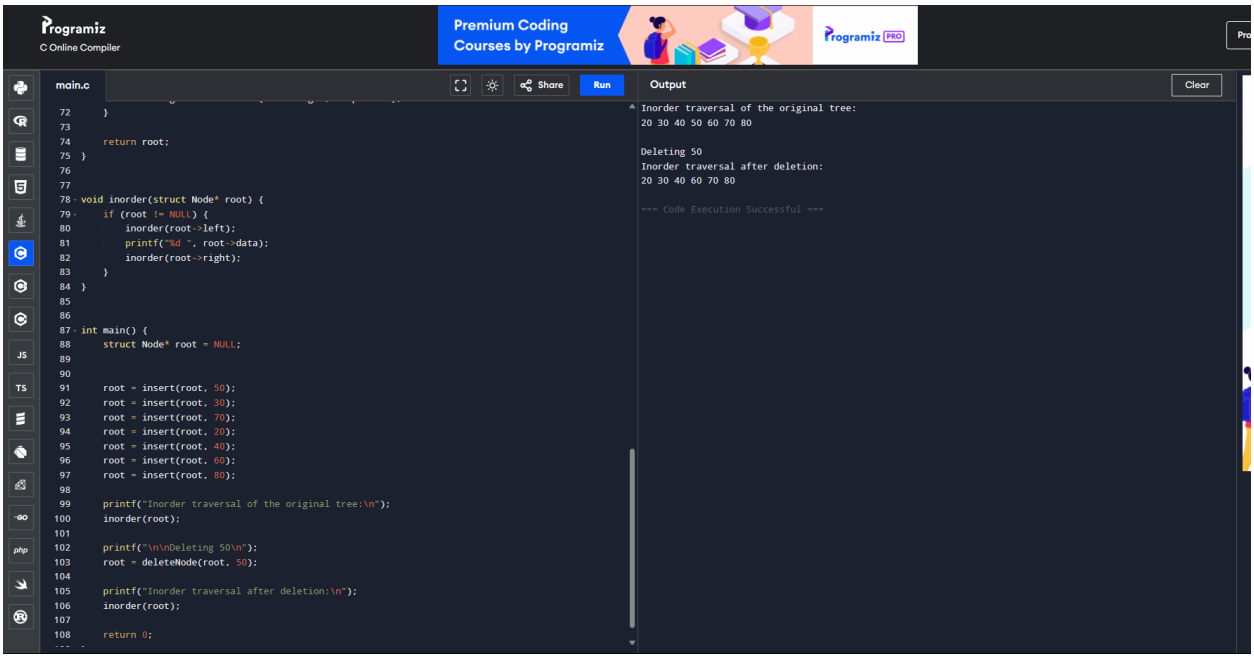


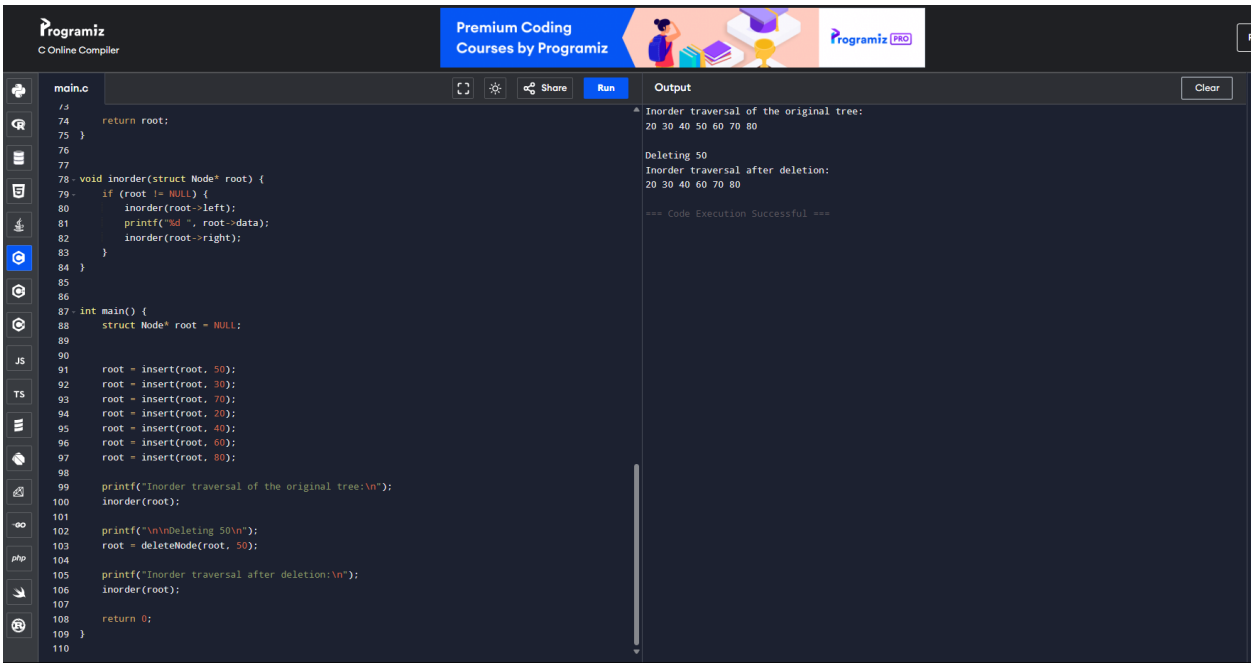


20. Write a program in C to implement deletion from Binary tree

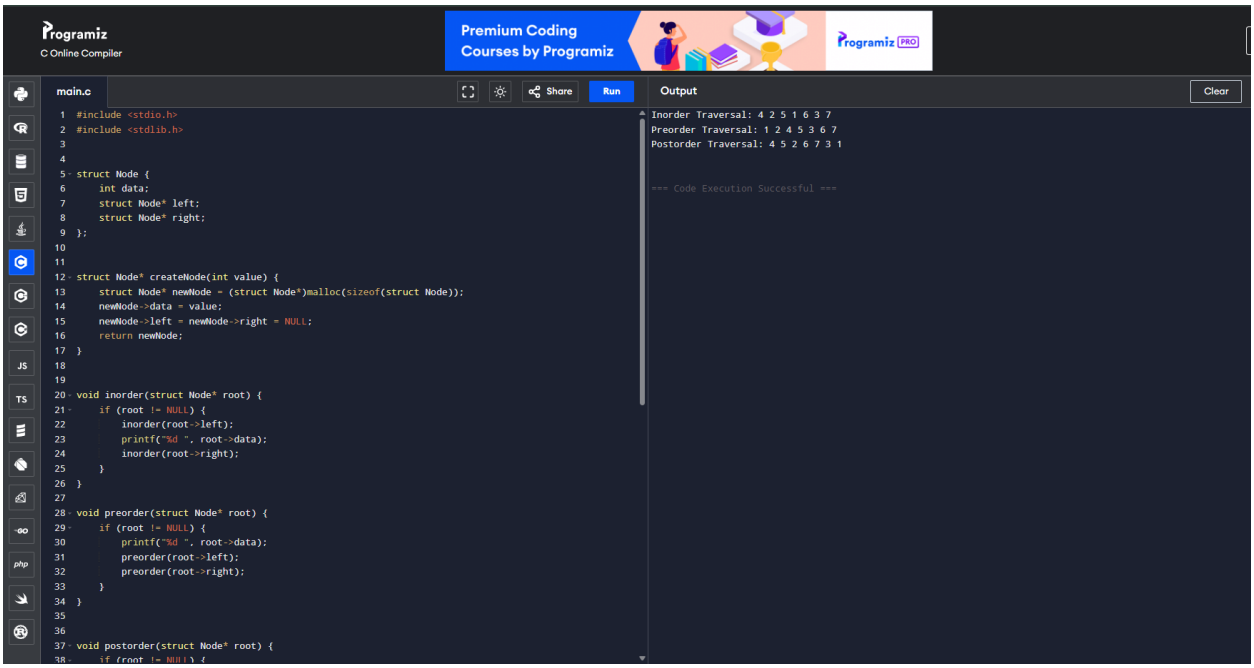


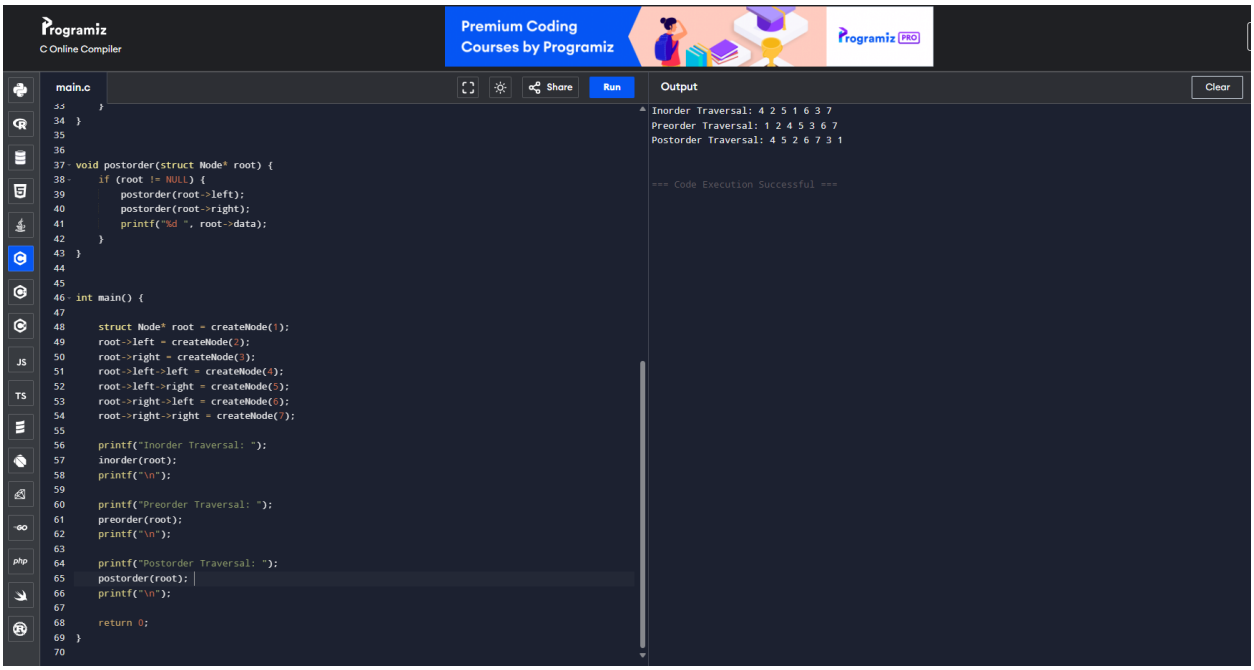



Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
}
return root;
}
void inorder(struct Node* root) {
if (root != NULL) {
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
}
}
int main() {
struct Node* root = NULL;
root = insert(root, 50);
root = insert(root, 30);
root = insert(root, 70);
root = insert(root, 20);
root = insert(root, 40);
root = insert(root, 60);
root = insert(root, 80);
printf("Inorder traversal of the original tree:\n");
inorder(root);
printf("\n\nDeleting 50\n");
root = deleteNode(root, 50);
printf("Inorder traversal after deletion:\n");
inorder(root);
return 0;
Inorder traversal of the original tree:
20 30 40 50 60 70 80
Deleting 50
Inorder traversal after deletion:
20 30 40 60 70 80
=== Code Execution Successful ===

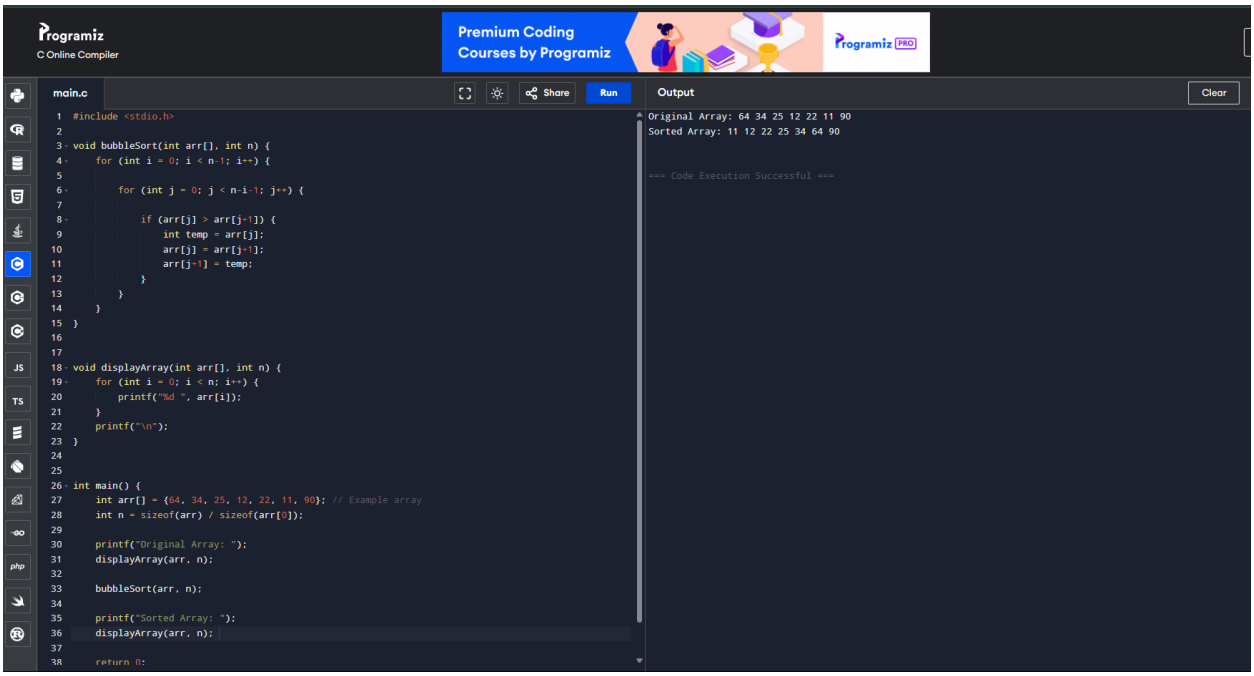

Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Share
Run
Output
Clear
return root;
}
void inorder(struct Node* root) {
if (root != NULL) {
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
}
}
int main() {
struct Node* root = NULL;
root = insert(root, 50);
root = insert(root, 30);
root = insert(root, 70);
root = insert(root, 20);
root = insert(root, 40);
root = insert(root, 60);
root = insert(root, 80);
printf("Inorder traversal of the original tree:\n");
inorder(root);
printf("\n\nDeleting 50\n");
root = deleteNode(root, 50);
printf("Inorder traversal after deletion:\n");
inorder(root);
return 0;
}
Inorder traversal of the original tree:
20 30 40 50 60 70 80
Deleting 50
Inorder traversal after deletion:
20 30 40 60 70 80
=== Code Execution Successful ===

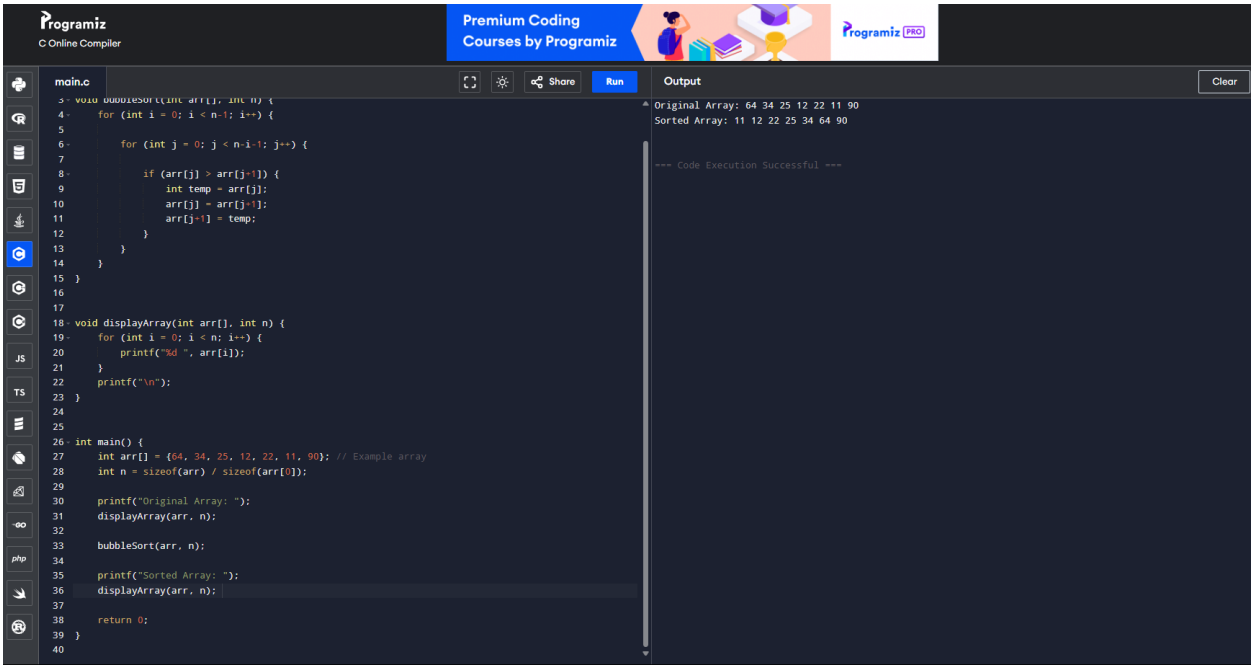
21. Write a program in C to implement recursive tree traversals (Inorder; Preorder; Postor)



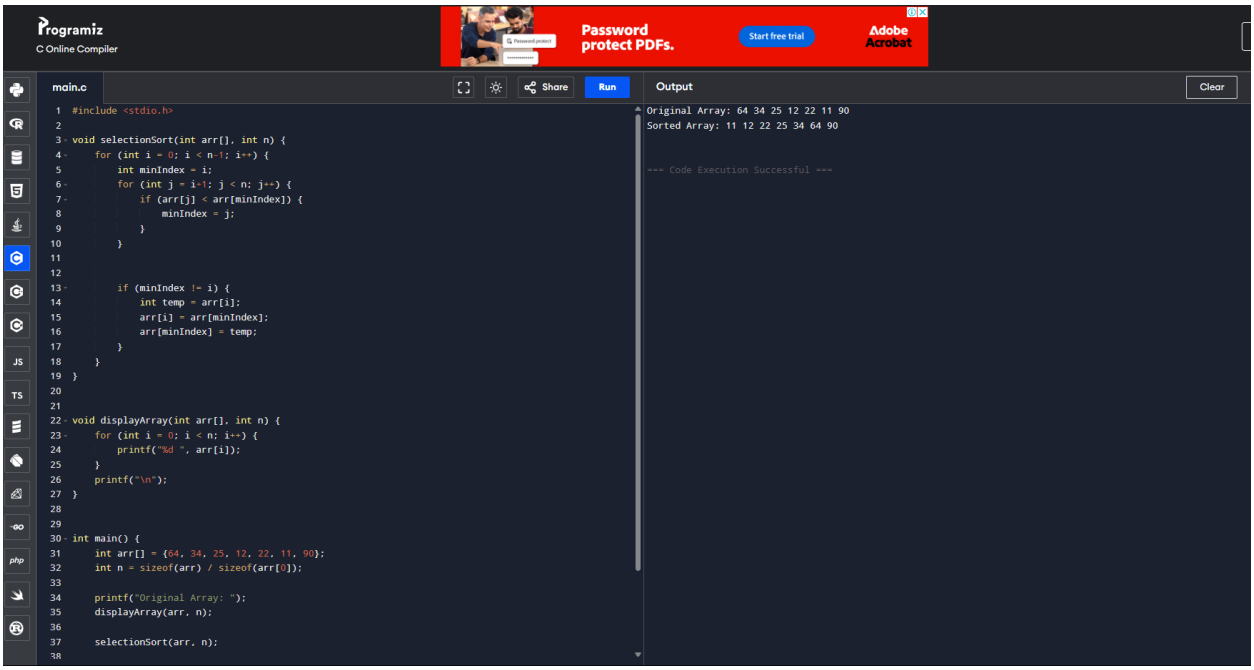


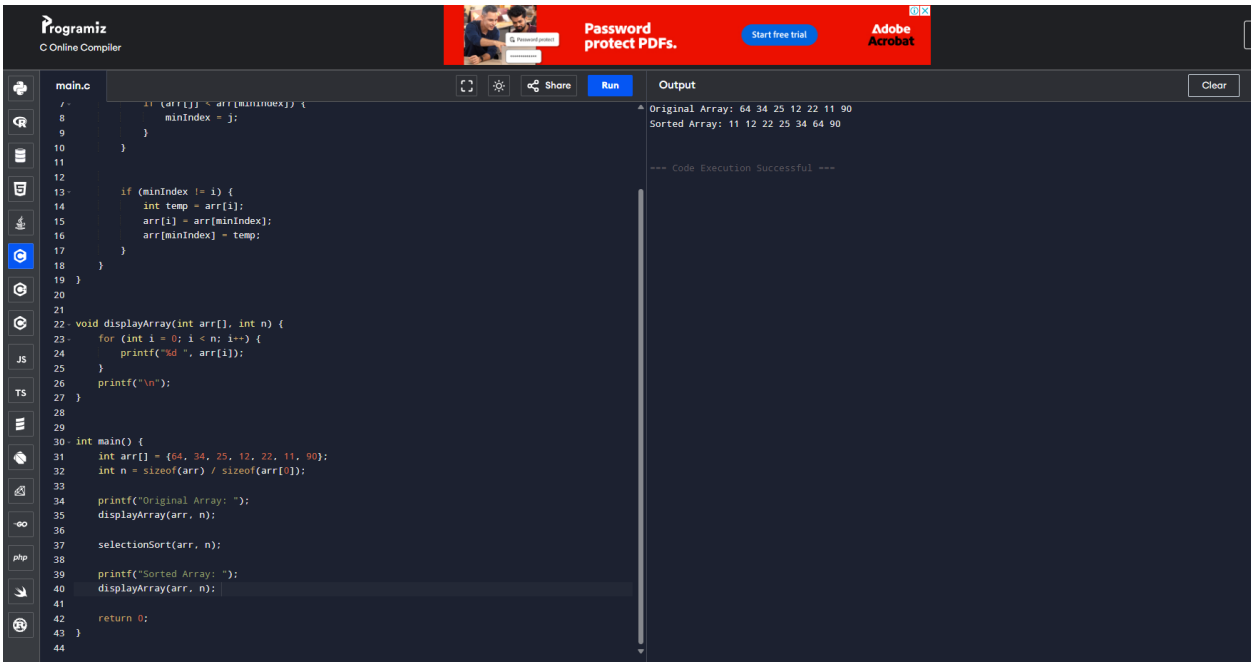
22.Write a program in C to Sort a list using Bubble Sort.



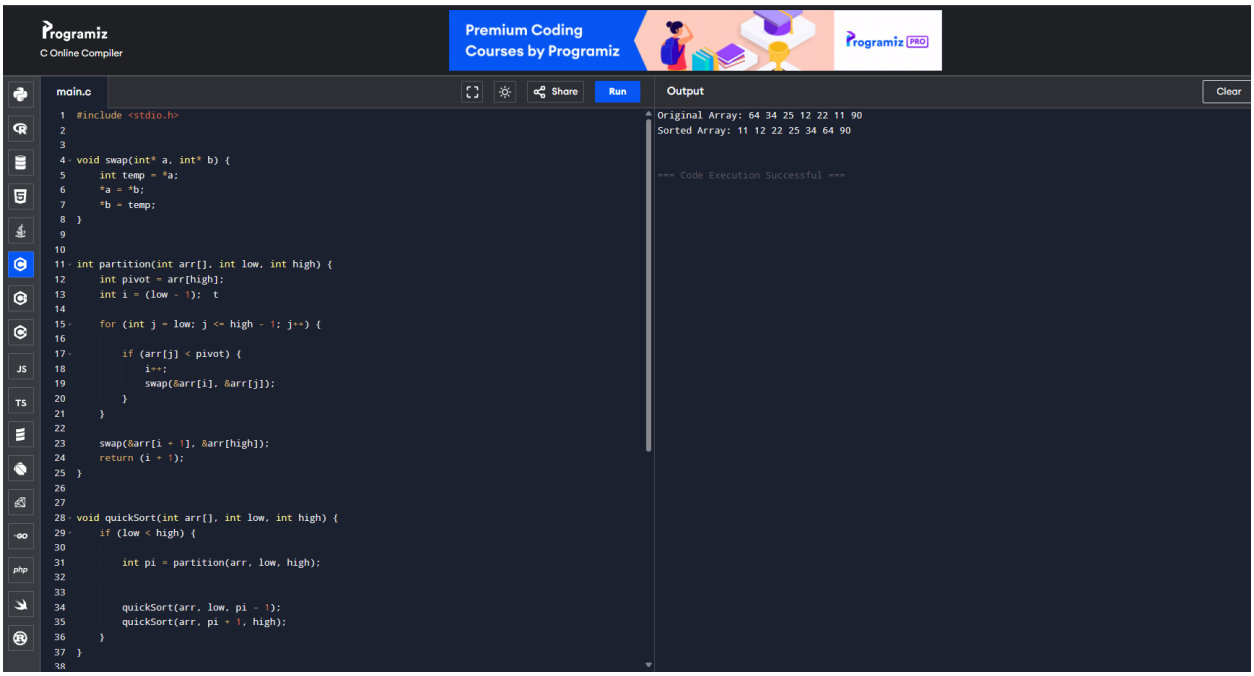


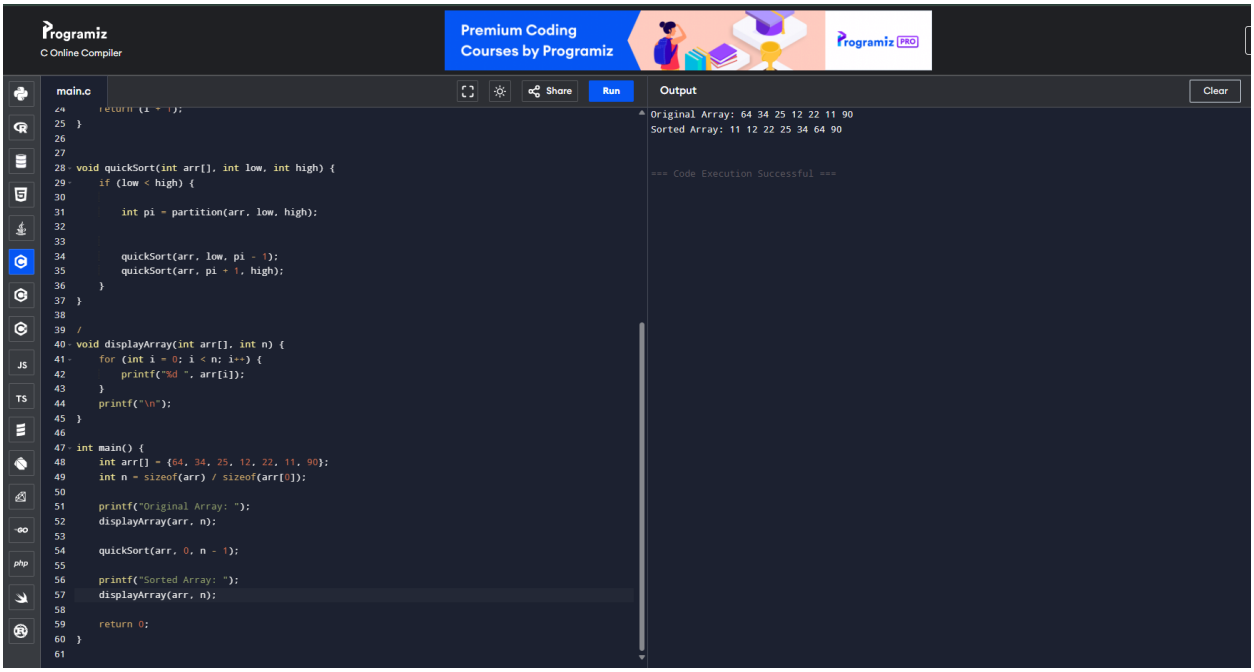
23. Write a program in C to Sort a list using Selection Sort.



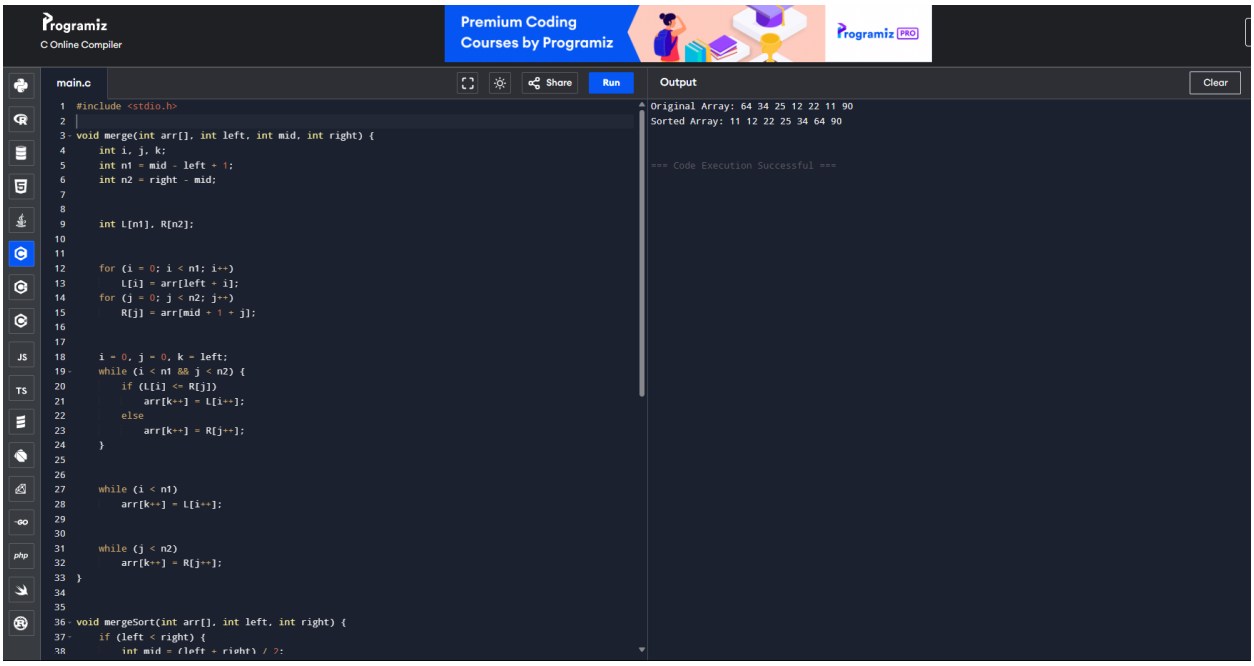


24.Write a program in C to sort a list using Quick Sort

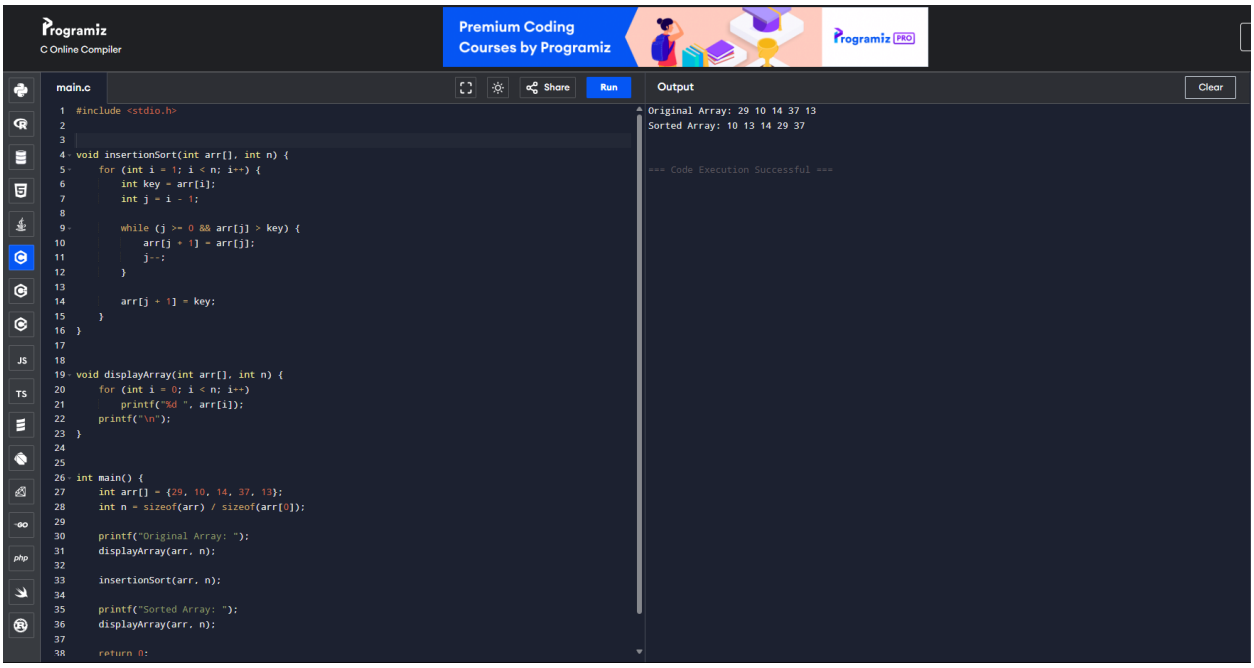


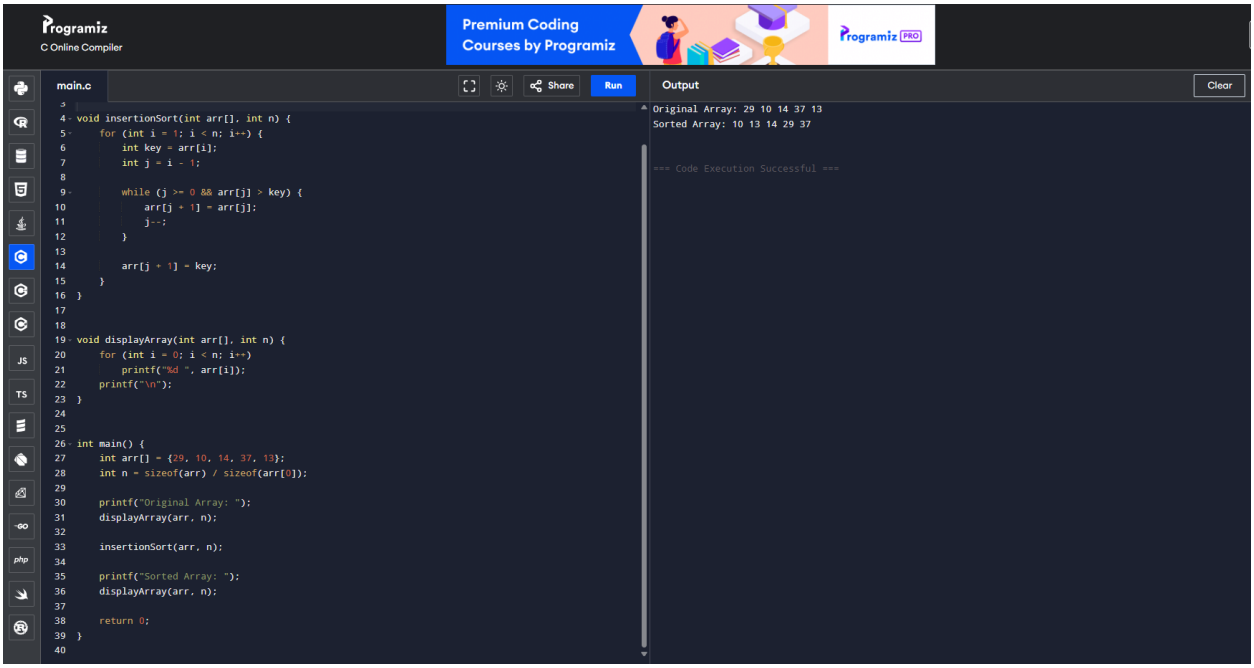


25.Write a program in C to sort a list using Merge Sort



26. Write a program in C to sort a list using Insertion Sort





27. Write a program in C to sort a list using Heap Sort