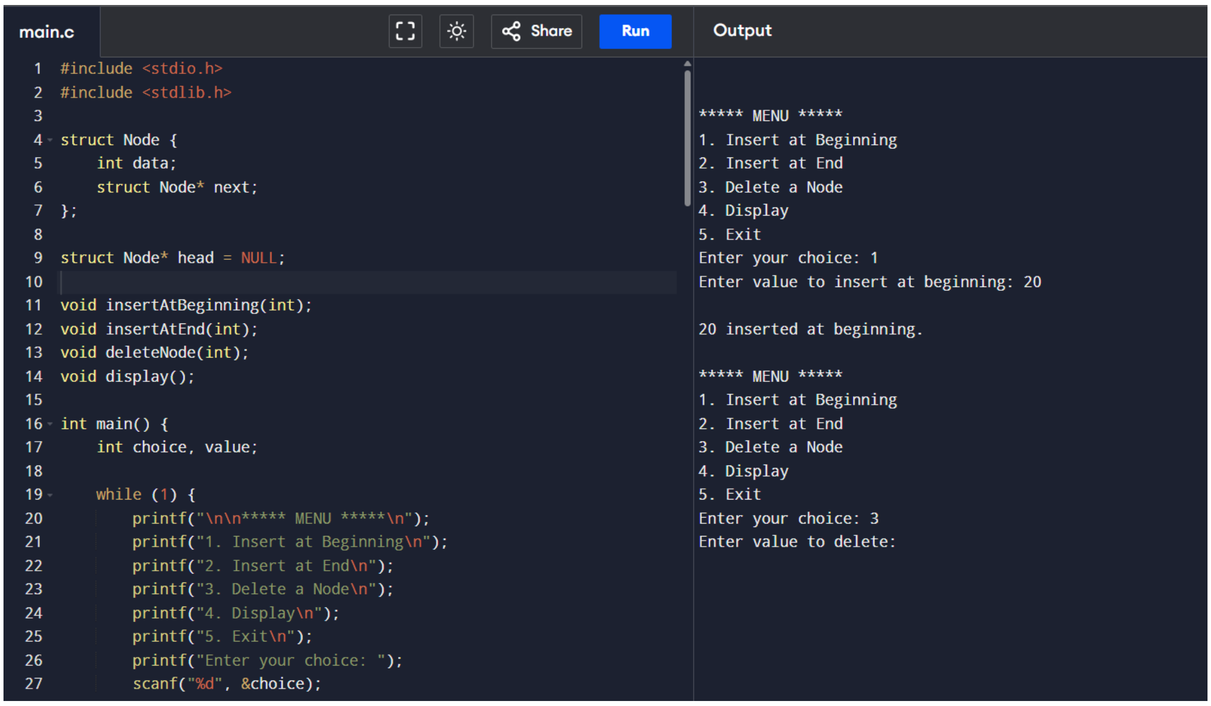
10.Write a C program to implement Linked list operations

```c
#include <stdio.h>

#include <stdlib.h>

struct Node {

   int data;

   struct Node* next;

};

struct Node* head = NULL;

void insertAtBeginning(int);

void insertAtEnd(int);

void deleteNode(int);

void display();

int main() {

   int choice, value;

   while (1) {

      printf("\n\n***** MENU *****\n");

      printf("1. Insert at Beginning\n");

      printf("2. Insert at End\n");

      printf("3. Delete a Node\n");

      printf("4. Display\n");

      printf("5. Exit\n");

      printf("Enter your choice: ");

      scanf("%d", &choice);

      switch (choice) {

         case 1:

            printf("Enter value to insert at beginning: ");

            scanf("%d", &value);

            insertAtBeginning(value);

            break;

         case 2:

            printf("Enter value to insert at end: ");
```

```c
                scanf("%d", &value);
                insertAtEnd(value);
                break;
            case 3:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                deleteNode(value);
                break;
            case 4:
                display();
                break;
            case 5:
                exit(0);
            default:
                printf("\nInvalid choice! Try again.");
        }
    }
    return 0;
}
void insertAtBeginning(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
    printf("\n%d inserted at beginning.", value);
}
void insertAtEnd(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    struct Node* temp = head;
    newNode->data = value;
    newNode->next = NULL;
    if (head == NULL) {
        head = newNode;
    } else {
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    printf("\n%d inserted at end.", value);
}
void deleteNode(int value) {
    struct Node *temp = head, *prev = NULL;
    if (temp == NULL) {
        printf("\nList is empty. Deletion not possible.");
        return;
    }
    if (temp != NULL && temp->data == value) {
        head = temp->next;
        free(temp);
        printf("\n%d deleted from list.", value);
        return;
    }
    while (temp != NULL && temp->data != value) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("\n%d not found in the list.", value);
        return;
    }
    prev->next = temp->next;
    free(temp);
    printf("\n%d deleted from list.", value);
}
void display() {
    struct Node* temp = head;
    if (temp == NULL) {
        printf("\nList is empty.");
        return;
    }
    printf("\nLinked List elements: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL");
}
```



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

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

struct Node* head = NULL;

void insertAtBeginning(int);
void insertAtEnd(int);
void deleteNode(int);
void display();

int main() {
    int choice, value;

    while (1) {
        printf("\n\n***** MENU *****\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at End\n");
        printf("3. Delete a Node\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
***** MENU *****
1. Insert at Beginning
2. Insert at End
3. Delete a Node
4. Display
5. Exit
Enter your choice: 1
Enter value to insert at beginning: 20

20 inserted at beginning.

***** MENU *****
1. Insert at Beginning
2. Insert at End
3. Delete a Node
4. Display
5. Exit
Enter your choice: 3
Enter value to delete:
```