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

#define MAX 100

int stack[MAX];
int top = -1;


void push() {
    int value;
    if (top == MAX - 1) {
        printf("Stack Overflow! Cannot push element.\n");
    } else {
        printf("Enter the value to be pushed: ");
        scanf("%d", &value);
        top++;
        stack[top] = value;
        printf("Element %d pushed onto the stack.\n", value);
    }
}

void pop() {
    if (top == -1) {
        printf("Stack Underflow! No element to pop.\n");
    } else {
        printf("Element %d popped from the stack.\n", stack[top]);
        top--;
    }
}

void display() {
    if (top == -1) {
        printf("Stack is empty.\n");
    } else {
        printf("Stack elements are:\n");
        for (int i = top; i >= 0; i--) {
            printf("%d\n", stack[i]);
        }
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nStack Operations Menu:\n");
        printf("1. PUSH\n");
        printf("2. POP\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}

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

#define MAX 100

int queue[MAX];
int front = -1, rear = -1;


void enqueue() {
    int value;
    if (rear == MAX - 1) {
        printf("Queue Overflow! Cannot enqueue element.\n");
    } else {
        printf("Enter the value to be enqueued: ");
        scanf("%d", &value);
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("Element %d enqueued into the queue.\n", value);
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow! No element to dequeue.\n");
    } else {
        printf("Element %d dequeued from the queue.\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1;  // Reset the queue if empty
        }
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue elements are:\n");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
2.#include <stdio.h>
#include <stdlib.h>

#define MAX 100

int queue[MAX];
int front = -1, rear = -1;


void enqueue() {
    int value;
    if (rear == MAX - 1) {
        printf("Queue Overflow! Cannot enqueue element.\n");
    } else {
        printf("Enter the value to be enqueued: ");
        scanf("%d", &value);
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("Element %d enqueued into the queue.\n", value);
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow! No element to dequeue.\n");
    } else {
        printf("Element %d dequeued from the queue.\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1;  // Reset the queue if empty
        }
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue elements are:\n");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

int queue[MAX];
int front = -1, rear = -1;


void enqueue() {
    int value;
    if (rear == MAX - 1) {
        printf("Queue Overflow! Cannot enqueue element.\n");
    } else {
        printf("Enter the value to be enqueued: ");
        scanf("%d", &value);
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("Element %d enqueued into the queue.\n", value);
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow! No element to dequeue.\n");
    } else {
        printf("Element %d dequeued from the queue.\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1;  // Reset the queue if empty
        }
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue elements are:\n");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}

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

#define MAX 100

int queue[MAX];
int front = -1, rear = -1;


void enqueue() {
    int value;
    if (rear == MAX - 1) {
        printf("Queue Overflow! Cannot enqueue element.\n");
    } else {
        printf("Enter the value to be enqueued: ");
        scanf("%d", &value);
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("Element %d enqueued into the queue.\n", value);
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow! No element to dequeue.\n");
    } else {
        printf("Element %d dequeued from the queue.\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1;  // Reset the queue if empty
        }
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue elements are:\n");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}



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

#define MAX 100

int queue[MAX];
int front = -1, rear = -1;


void enqueue() {
    int value;
    if (rear == MAX - 1) {
        printf("Queue Overflow! Cannot enqueue element.\n");
    } else {
        printf("Enter the value to be enqueued: ");
        scanf("%d", &value);
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("Element %d enqueued into the queue.\n", value);
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow! No element to dequeue.\n");
    } else {
        printf("Element %d dequeued from the queue.\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1;  // Reset the queue if empty
        }
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue elements are:\n");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}


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

#define MAX 100

int queue[MAX];
int front = -1, rear = -1;


void enqueue() {
    int value;
    if (rear == MAX - 1) {
        printf("Queue Overflow! Cannot enqueue element.\n");
    } else {
        printf("Enter the value to be enqueued: ");
        scanf("%d", &value);
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("Element %d enqueued into the queue.\n", value);
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow! No element to dequeue.\n");
    } else {
        printf("Element %d dequeued from the queue.\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1;  // Reset the queue if empty
        }
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue elements are:\n");
```

```c
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nQueue Operations Menu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}

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

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

struct Node* head = NULL;

struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}


void insertAtBeginning() {
    int value;
    printf("Enter the value to insert at the beginning: ");
    scanf("%d", &value);
    struct Node* newNode = createNode(value);
    newNode->next = head;
    head = newNode;
    printf("Node inserted at the beginning.\n");
```

```c
}

void insertAtEnd() {
    int value;
    printf("Enter the value to insert at the end: ");
    scanf("%d", &value);
    struct Node* newNode = createNode(value);
    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    printf("Node inserted at the end.\n");
}

void insertAtPosition() {
    int value, pos;
    printf("Enter the value to insert: ");
    scanf("%d", &value);
    printf("Enter the position to insert: ");
    scanf("%d", &pos);

    struct Node* newNode = createNode(value);
    if (pos == 1) {
        newNode->next = head;
        head = newNode;
    } else {
        struct Node* temp = head;
        for (int i = 1; i < pos - 1 && temp != NULL; i++) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Invalid position!\n");
            free(newNode);
            return;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
    printf("Node inserted at position %d.\n", pos);
}


void deleteFromBeginning() {
    if (head == NULL) {
        printf("List is empty. Nothing to delete.\n");
    } else {
        struct Node* temp = head;
        head = head->next;
        printf("Node with value %d deleted from the beginning.\n", temp->data);
        free(temp);
    }
}

void deleteFromEnd() {
    if (head == NULL) {
        printf("List is empty. Nothing to delete.\n");
    } else if (head->next == NULL) {
        printf("Node with value %d deleted from the end.\n", head->data);
        free(head);
        head = NULL;
    } else {
        struct Node* temp = head;
        while (temp->next->next != NULL) {
            temp = temp->next;
        }
        printf("Node with value %d deleted from the end.\n", temp->next->data);
        free(temp->next);
        temp->next = NULL;
    }
}

void deleteFromPosition() {
    int pos;
    printf("Enter the position to delete: ");
    scanf("%d", &pos);

    if (head == NULL) {
        printf("List is empty. Nothing to delete.\n");
    } else if (pos == 1) {
        struct Node* temp = head;
        head = head->next;
        printf("Node with value %d deleted from position 1.\n", temp->data);
        free(temp);
    } else {
        struct Node* temp = head;
        for (int i = 1; i < pos - 1 && temp != NULL; i++) {
            temp = temp->next;
        }
        if (temp == NULL || temp->next == NULL) {
            printf("Invalid position!\n");
            return;
        }
        struct Node* toDelete = temp->next;
        temp->next = toDelete->next;
        printf("Node with value %d deleted from position %d.\n", toDelete->data, pos);
        free(toDelete);
    }
}

void search() {
    int value, pos = 1;
    printf("Enter the value to search: ");
    scanf("%d", &value);

    struct Node* temp = head;
    while (temp != NULL) {
        if (temp->data == value) {
            printf("Element %d found at position %d.\n", value, pos);
            return;
        }
        temp = temp->next;
        pos++;
    }
    printf("Element %d not found in the list.\n", value);
}

void display() {
    if (head == NULL) {
        printf("List is empty.\n");
    } else {
        printf("List elements are: ");
        struct Node* temp = head;
        while (temp != NULL) {
            printf("%d -> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}

int main() {
    int choice;
    while (1) {
        printf("\nSingly Linked List Operations Menu:\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Position\n");
        printf("4. Delete from Beginning\n");
        printf("5. Delete from End\n");
        printf("6. Delete from Position\n");
        printf("7. Search\n");
        printf("8. Display\n");
        printf("9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertAtBeginning();
                break;
            case 2:
                insertAtEnd();
                break;
            case 3:
                insertAtPosition();
                break;
            case 4:
                deleteFromBeginning();
                break;
            case 5:
                deleteFromEnd();
                break;
            case 6:
                deleteFromPosition();
                break;
            case 7:
                search();
                break;
            case 8:
                display();
                break;
            case 9:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}

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

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

struct Node* head = NULL;


struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertAtBeginning() {
    int value;
    printf("Enter the value to insert at the beginning: ");
    scanf("%d", &value);
    struct Node* newNode = createNode(value);

    if (head == NULL) {
        head = newNode;
    } else {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    }
    printf("Node inserted at the beginning.\n");
}

void insertAtEnd() {
    int value;
    printf("Enter the value to insert at the end: ");
    scanf("%d", &value);
    struct Node* newNode = createNode(value);

    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
    printf("Node inserted at the end.\n");
}

void insertAtPosition() {
    int value, pos;
    printf("Enter the value to insert: ");
    scanf("%d", &value);
    printf("Enter the position to insert: ");
    scanf("%d", &pos);

    struct Node* newNode = createNode(value);

    if (pos == 1) {
        newNode->next = head;
        if (head != NULL) {
            head->prev = newNode;
        }
        head = newNode;
    } else {
        struct Node* temp = head;
        for (int i = 1; i < pos - 1 && temp != NULL; i++) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Invalid position!\n");
            free(newNode);
            return;
        }
        newNode->next = temp->next;
        newNode->prev = temp;
        if (temp->next != NULL) {
            temp->next->prev = newNode;
        }
        temp->next = newNode;
    }
    printf("Node inserted at position %d.\n", pos);
}

void deleteFromBeginning() {
    if (head == NULL) {
        printf("List is empty. Nothing to delete.\n");
    } else {
        struct Node* temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        printf("Node with value %d deleted from the beginning.\n", temp->data);
        free(temp);
    }
}

void deleteFromEnd() {
    if (head == NULL) {
        printf("List is empty. Nothing to delete.\n");
    } else if (head->next == NULL) {
        printf("Node with value %d deleted from the end.\n", head->data);
        free(head);
        head = NULL;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        printf("Node with value %d deleted from the end.\n", temp->data);
        temp->prev->next = NULL;
        free(temp);
    }
}

void deleteFromPosition() {
    int pos;
    printf("Enter the position to delete: ");
    scanf("%d", &pos);

    if (head == NULL) {
        printf("List is empty. Nothing to delete.\n");
    } else if (pos == 1) {
        struct Node* temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        printf("Node with value %d deleted from position 1.\n", temp->data);
        free(temp);
    } else {
        struct Node* temp = head;
        for (int i = 1; i < pos && temp != NULL; i++) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Invalid position!\n");
            return;
        }
        printf("Node with value %d deleted from position %d.\n", temp->data, pos);
        if (temp->prev != NULL) {
            temp->prev->next = temp->next;
        }
        if (temp->next != NULL) {
            temp->next->prev = temp->prev;
        }
        free(temp);
    }
}

void search() {
    int value, pos = 1;
    printf("Enter the value to search: ");
    scanf("%d", &value);

    struct Node* temp = head;
    while (temp != NULL) {
        if (temp->data == value) {
            printf("Element %d found at position %d.\n", value, pos);
            return;
        }
        temp = temp->next;
        pos++;
    }
    printf("Element %d not found in the list.\n", value);
}


void display() {
    if (head == NULL) {
        printf("List is empty.\n");
    } else {
        printf("List elements are: ");
        struct Node* temp = head;
        while (temp != NULL) {
            printf("%d <-> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nDoubly Linked List Operations Menu:\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Position\n");
        printf("4. Delete from Beginning\n");
        printf("5. Delete from End\n");
        printf("6. Delete from Position\n");
        printf("7. Search\n");
        printf("8. Display\n");
        printf("9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertAtBeginning();
                break;
            case 2:
                insertAtEnd();
                break;
            case 3:
                insertAtPosition();
                break;
            case 4:
                deleteFromBeginning();
                break;
            case 5:
                deleteFromEnd();
                break;
            case 6:
                deleteFromPosition();
                break;
            case 7:
                search();
                break;
            case 8:
                display();
                break;
            case 9:
                printf("Exiting program. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```