**A4-16010122083**

**Copy kar rahe ho toh todhe change karo proper**

**Single Linked List**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct node

{

    int data;

    struct node \*next;

};

void push(struct node \*\*head, int data, int pos)

{

    struct node \*newNode = malloc(sizeof(struct node));

    newNode->data = data;

    if (\*head == NULL || pos == 1)

    {

        newNode->next = \*head;

        \*head = newNode;

    }

    else

    {

        struct node \*current = \*head;

        int i = 1;

        while (i < pos - 1 && current->next != NULL)

        {

            current = current->next;

            i++;

        }

        newNode->next = current->next;

        current->next = newNode;

    }

}

int pop(int pos, struct node \*\*head)

{

    int val;

    if (pos == 1)

    {

        val=(\*head)->data;

        struct node \*temp = \*head;

        \*head = (\*head)->next;

        free(temp);

    }

    else

    {

        struct node \*current = \*head;

        int i = 1;

        while (i < pos - 1 && current->next != NULL)

        {

            current = current->next;

            i++;

        }

        if (current->next != NULL)

        {

            struct node \*temp = current->next;

            current->next = temp->next;

            val=(temp)->data;

            free(temp);

        }

    }

    return val;

}

void search(struct node \*\*head, int num)

{

    struct node \*ptr=\*head;

    int pos;

    bool found=false;

    while(ptr!=NULL)

    {

        pos++;

        if(ptr->data==num)

        {

            found=true;

            break;

        }

        ptr=ptr->next;

    }

    if(found==true)

    {

        printf("The number is found at positon %d \n",pos);

    }

    else

    {

        printf("Not found\n");

    }

}

void display(struct node \*head)

{

    if (head == NULL)

    {

        printf("The List is Empty\n");

    }

    else

    {

        struct node \*ptr = head;

        printf("The Linked List\n");

        while (ptr != NULL)

        {

            printf("%d ->", ptr->data);

            ptr = ptr->next;

        }

        printf("NULL\n");

    }

}

int main()

{

    int choice;

    struct node \*head = NULL;

    int num = 0;

    int pos = 0;

    while (choice != 5)

    {

        printf("\nSingle Linked List Module\n");

        printf("1. Insert \n");

        printf("2. Delete\n");

        printf("3. Print All\n");

        printf("4. Searching\n");

        printf("5. Exit\n");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

        {

            int val;

            printf("Enter the number to be inserted\n");

            scanf("%d", &val);

            if (num != 0)

            {

                printf("Enter the position of the Number in range (1-%d)\n", num + 1);

                scanf("%d", &pos);

                if (pos < 1 || pos > num + 1)

                {

                    printf("Invalid position\n");

                }

                else

                {

                    push(&head, val, pos);

                    num++;

                }

            }

            else

            {

                push(&head, val, 1);

                num++;

            }

            break;

        }

        case 2:

        {

            int posi;

            printf("Enter the position of the node to be deleted\n");

            scanf("%d",&posi);

            if(posi<=num)

            {

                int popped=pop(posi,&head);

                printf("The popped number is %d\n",popped);

                num--;

            }

            else

            {

                printf("Enter the position from the range\n");

            }

            break;

        }

        case 3:

        {

            display(head);

            break;

        }

        case 4:

        {

            int find;

            printf("Enter the number to be found");

            scanf("%d",&find);

            search(&head,find);

            break;

        }

        case 5:

        {

            printf("Thank you for using Single Linked List Modulation");

            break;

        }

        default:

        {

            printf("Invalid Input\n");

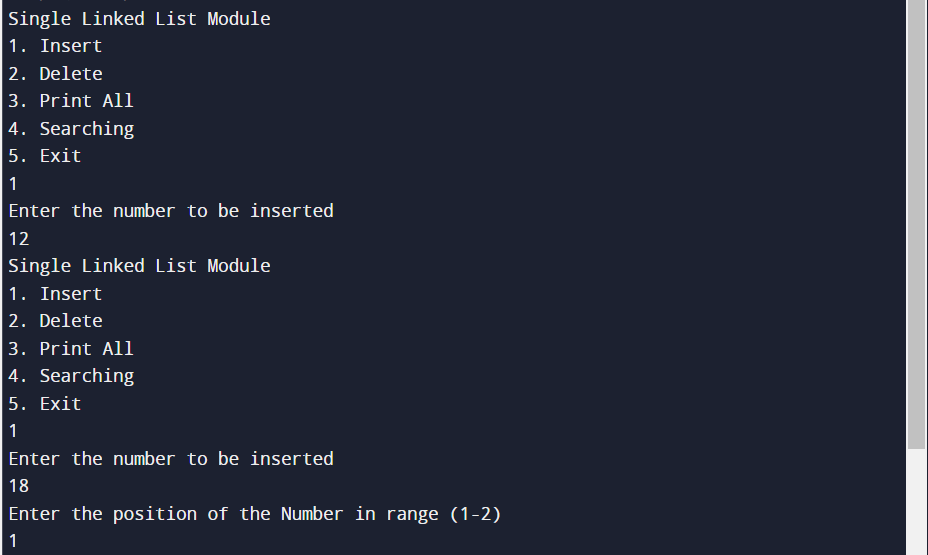
        }

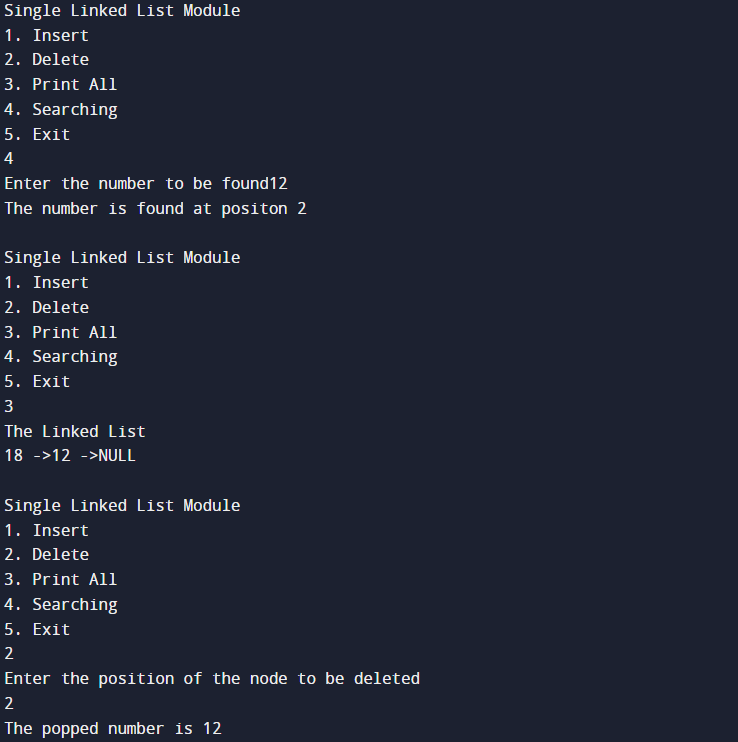
        }

    }

    return 0;

}





**Circular Linked List**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct node {

    int data;

    struct node \*next;

};

void insert(struct node \*\*head, int data) {

    struct node \*newNode = malloc(sizeof(struct node));

    newNode->data = data;

    if (\*head == NULL) {

        newNode->next = newNode;

        \*head = newNode;

    } else {

        struct node \*tail = \*head;

        while (tail->next != \*head) {

            tail = tail->next;

        }

        newNode->next = \*head;

        tail->next = newNode;

    }

}

void delete(struct node \*\*head, int data) {

    if (\*head == NULL) {

        printf("The list is empty\n");

        return;

    }

    struct node \*current = \*head, \*prev = NULL;

    do {

        if (current->data == data)

            break;

        prev = current;

        current = current->next;

    } while (current != \*head);

    if (current == \*head && prev != NULL) {

        printf("Element not found in the list\n");

        return;

    }

    if (current == \*head) {

        \*head = current->next;

    }

    if (prev != NULL) {

        prev->next = current->next;

    }

    free(current);

}

void display(struct node \*head) {

    if (head == NULL) {

        printf("The List is Empty\n");

    } else {

        struct node \*current = head;

        printf("The Circular Linked List\n");

        do {

            printf("%d -> ", current->data);

            current = current->next;

        } while (current != head);

        printf("...\n");

    }

}

int main() {

    int choice;

    struct node \*head = NULL;

    int data;

    while (1)

    {

        printf("\nCircular Linked List Module\n");

        printf("1. Insert\n");

        printf("2. Delete\n");

        printf("3. Print All\n");

        printf("4. Exit\n");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1:

                printf("Enter the number to be inserted: ");

                scanf("%d", &data);

                insert(&head, data);

                break;

            case 2:

                printf("Enter the number to be deleted: ");

                scanf("%d", &data);

                delete(&head, data);

                break;

            case 3:

                display(head);

                break;

            case 4:

                printf("Thank you for using Circular Linked List Module\n");

                exit(0);

            default:

                printf("Invalid Input\n");

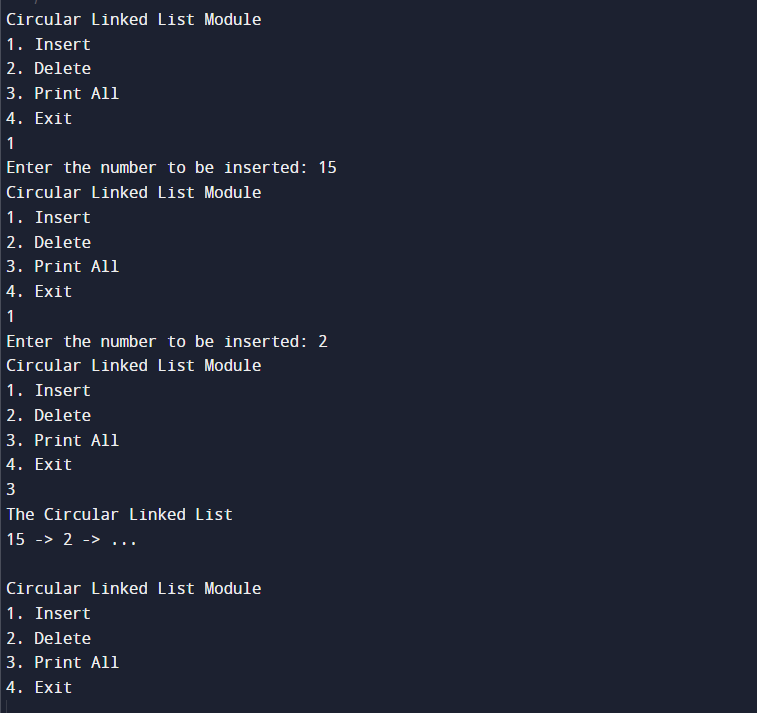
        }

    }

    return 0;

}

Output:



**Doubly Linked List**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct node {

    int data;

    struct node\* next;

    struct node\* prev;

};

void insert(struct node\*\* head, int data)

{

    struct node\* newNode = (struct node\*)malloc(sizeof(struct node));

    newNode->data = data;

    newNode->next = NULL;

    newNode->prev = NULL;

    if (\*head == NULL)

    {

        \*head = newNode;

    }

    else

    {

        struct node\* current = \*head;

        while (current->next != NULL)

        {

            current = current->next;

        }

        current->next = newNode;

        newNode->prev = current;

    }

}

void delete(struct node\*\* head, int data)

{

    struct node\* current = \*head;

    while (current != NULL)

    {

        if (current->data == data)

        {

            if (current->prev != NULL)

            {

                current->prev->next = current->next;

            }

            else

            {

                \*head = current->next;

            }

            if (current->next != NULL)

            {

                current->next->prev = current->prev;

            }

            free(current);

            return;

        }

        current = current->next;

    }

    printf("Element not found in the list\n");

}

void display(struct node\* head)

{

    if (head == NULL)

    {

        printf("The List is Empty\n");

    }

    else

    {

        struct node\* current = head;

        printf("The Doubly Linked List\n");

        while (current != NULL) {

            printf("%d <-> ", current->data);

            current = current->next;

        }

        printf("NULL\n");

    }

}

int main()

{

    int choice;

    struct node\* head = NULL;

    int data;

    while (1)

    {

        printf("\nDoubly Linked List Module\n");

        printf("1. Insert\n");

        printf("2. Delete\n");

        printf("3. Print All\n");

        printf("4. Exit\n");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1:

                printf("Enter the number to be inserted: ");

                scanf("%d", &data);

                insert(&head, data);

                break;

            case 2:

                printf("Enter the number to be deleted: ");

                scanf("%d", &data);

                delete(&head, data);

                break;

            case 3:

                display(head);

                break;

            case 4:

                printf("Thank you for using Doubly Linked List Module\n");

                exit(0);

            default:

                printf("Invalid Input\n");

        }

    }

    return 0;

}

