**// Link List (insert, delete, sorting,searching,traversing,exit).cpp**

#include <bits/stdc++.h>

using namespace std;

int cunt = 0;

struct Node{

    int data;

    Node \*link;

};

**// Insert First.....**

struct Node \*insertFirst(Node \*head, int data){

    Node \*newNode = new Node;

    newNode->data = data;

    cunt++;

    newNode->link = head->link;

    head->link = newNode;

    return head;

}

**// Insert Last.....**

struct Node \*insertLast(Node \*head, int data){

    Node \*newNode = new Node;

    Node \*ptr = head;

    while (ptr->link != NULL){

        ptr = ptr->link;

    }

    newNode->data = data;

    cunt++;

    ptr->link = newNode;  
 newNode->link = NULL;

    return head;

}

**// Insert custom....**

struct Node \*insertCustom(Node \*head, int data, int index){

    Node \*newNode = new Node;

    Node \*ptr = head;

    for (int i = 0; i < index - 1; i++){

        ptr = ptr->link;

    }

    newNode->data = data;

    cunt++;

    newNode->link = ptr->link;

    ptr->link = newNode;

    return head;

}

**// Delete first.....**

struct Node \*deleteFirst(Node \*head){

    head->link = head->link->link;

    if (cunt <= 0)  cunt = 0;

    else  cunt--;

    return head;

}

**// Delete last.....**

struct Node \*deleteLast(Node \*head){

    Node \*ptr = head;

    while (ptr->link->link != NULL){

        ptr = ptr->link;

    }

    ptr->link = NULL;

    if (cunt <= 0)    cunt = 0;

    else  cunt--;

    return head;

}

**// Delete custom........**

struct Node \*deleteCustom(Node \*head, int index){

    Node \*ptr = head;

    for (int i = 0; i < index - 1; i++){

        ptr = ptr->link;

    }

    ptr->link = ptr->link->link;

    if (cunt <= 0)  cunt = 0;

    else  cunt--;

    return head;

}

**// Print all element……**

void display(Node \*head){

    if (cunt <= 0){

        cout << "Link List is empty!" << endl;

    }

    else{

        cout << "Link list: ";

        Node \*ptr = head->link;

        while (ptr != NULL){

            cout << ptr->data << " ";

            ptr = ptr->link;

        }

        cout << endl;

    }

}

**// Sort all element**

void SORT(Node \*head){

    Node \*ptr = head->link,\*cpt;

    int temp;

    while(ptr != NULL){

        cpt = ptr->link;

        while(cpt!= NULL){

            if(ptr->data > cpt->data){

                temp = ptr->data;

                ptr->data = cpt->data;

                cpt->data = temp;

            }

            cpt = cpt->link;

        }

        ptr = ptr->link;

    }

    display(head);

    cout << "\nThe link list is sorted!!\n";

    \_sleep(2000);

}

**// Search one element**

void search(Node \*head,int element){

    Node \*ptr = head->link;

    int p=0;

    while(ptr != NULL){

        if(ptr->data == element){

            cout << "\nThe element "<<element << " is found!!\n";

            p++;

            \_sleep(2000);

            break;

        }

        ptr = ptr->link;

    }

    if(p==0){

        cout << "\nThe element "<<element << " is NOT found!!\n";

        \_sleep(2000);

    }

}

int main(){

    Node \*head = new Node;

    head->link = NULL;

    while (1){

        system("cls");

        display(head);

        cout << "\nEnter I for insert!!\n";

        cout << "Enter D for delete!!\n";

        cout << "Enter S for sorting!!\n";

        cout << "Enter F for searching!!\n";

        cout << "Enter E for exit!!\n";

        cout << "Enter your choice: ";

        char ch; cin >> ch;

        if (ch == 'i' || ch == 'I'){

            if (cunt == 0){

                cout << "Enter element for insert: ";

                int element;  cin >> element;

                insertFirst(head, element);

                cunt++;

            }

            else{

                system("cls");

                display(head);

                cout << "\nEnter F for insert first!\n";

                cout << "Enter L for insert last!\n";

                cout << "Enter N for insert n'th position!\n";

                cout << "Enter your choice: ";

                char c;  cin >> c;

                if (c == 'f' || c == 'F'){

                    cout << "Enter element for insert first: ";

                    int element; cin >> element;

                    insertFirst(head, element);

                }

                else if (c == 'l' || c == 'L'){

                    cout << "Enter element for insert last: ";

                    int element; cin >> element;

                    insertLast(head, element);

                }

                else if (c == 'n' || c == 'N'){

                    cout << "Enter index number: ";

                    int idx; cin >> idx;

                    cout << "Enter element for insert " << idx << " position: ";

                    int element;  cin >> element;

                    insertCustom(head, element, idx);

                }

                else    cout << "Invalid input Try again!\n";

            }

        }

        else if (ch == 'd' || ch == 'D'){

            system("cls");

            display(head);

            if (cunt <= 0){

                cout << "SORRY!! there is no element for delete!\n";

                cunt = 0;

                \_sleep(1200);

                continue;

            }

            cout << "\nEnter F for delete first!!\n";

            cout << "Enter L for delete last!!\n";

            cout << "Enter N for delete n'th position!!\n";

            cout << "Enter your choice: ";

            char ch; cin >> ch;

            if (ch == 'f' || ch == 'F'){

                deleteFirst(head);

            }

            else if (ch == 'l' || ch == 'L'){

                deleteLast(head);

            }

            else if (ch == 'n' || ch == 'N'){

                cout << "Enter index for delete: ";

                int idx;

                cin >> idx;

                deleteCustom(head, idx);

            }

            else   cout << "Invalid input Try again!\n";

        }

        else if(ch == 's' || ch == 'S'){

            SORT(head);

        }

        else if(ch == 'f' || ch == 'F'){

            cout << "Which element you want to search: ";

            int element;    cin >> element;

            search(head, element);

        }

        else if(ch == 'e' || ch == 'E'){

            cout << "\nCode is exited!!\n";

            \_sleep(1000);

            break;

        }

        else{

            cout << "\nInvalid input Try again!!\n";

            \_sleep(1000);

        }

    }

}

**// Circular Link List (insert, delete, sorting, searching, traversing, exit).cpp**

#include <bits/stdc++.h>

using namespace std;

int cunt = 0;

struct Node{

    int data;

    Node \*link;

};

**// Insert First.......**

Node \*insertFirst(Node \*head, int data){

    Node \*newNode = new Node{data, head};

    newNode->data = data;

    if (head == NULL){

        head = newNode;

        newNode->link = head;

        return head;

    }

    Node \*cpt = head;

    while (cpt->link != head){

        cpt = cpt->link;

    }

    cpt->link = newNode;

    head = newNode;

    cunt++;

    return head;

}

**// Insert Last ...........**

Node \*insertLast(Node \*head, int data){

    Node \*newNode = new Node{data, head};

    newNode->data = data;

    Node \*cpt = head;

    while (cpt->link != head){

        cpt = cpt->link;

    }

    cpt->link = newNode;

    newNode = head;

    cunt++;

    return head;

}

**// Custom Insert............**

Node \*insertCustom(Node \*head, int data, int index){

    Node \*newNode = new Node{data, NULL};

    if (head == NULL){

        head = newNode;

        newNode->link = head;

        return head;

    }

    Node \*ptr = head;

    for (int i = 1; i < index - 1 && ptr->link != head; i++){

        ptr = ptr->link;

    }

    newNode->link = ptr->link;

    ptr->link = newNode;

    if (index == 1){

        head = newNode;

    }

    cunt++;

    return head;

}

**// Delete first.....**

Node \*deleteFirst(Node \*head){

    if (head->link == head){

        head = NULL;

    }

    else{

        Node \*ptr = head;

        while (ptr->link != head){

            ptr = ptr->link;

        }

        head = head->link;

        ptr->link = head;

    }

    if(cunt>0) cunt--;

    else cunt = 0;

    return head;

}

**// Delete last.....**

Node \*deleteLast(Node \*head){

    Node \*cpt = head;

    if (head->link == head){

        head = NULL;

    }

    else{

        while (cpt->link->link != head){

            cpt = cpt->link;

        }

        cpt->link = head;

    }

    if(cunt>0) cunt--;

    else cunt = 0;

    return head;

}

**// Delete custom........**

Node \*deleteCustom(Node \*head, int index){

    Node \*cpt = head;

    if (head->link == head){

        head = NULL;

    }

    else if (index == 1){

        Node \*cpt = head;

        while (cpt->link != head){

            cpt = cpt->link;

        }

        head = head->link;

        cpt->link = head;

    }

    else{

        for (int i = 1; i < index - 1 && cpt->link != head; i++){

            cpt = cpt->link;

        }

        cpt->link = cpt->link->link;

    }

    if(cunt>0) cunt--;

    else cunt = 0;

    return head;

}

**// Print all element.....**

void display(Node \*head){

    if (head == NULL){

        cout << "Circular Linked List: empty\n";

        return;

    }

    cout << "Circular Linked List: ";

    Node \*cpt = head;

    do{

        cout << cpt->data << " ";

        cpt = cpt->link;

    } while (cpt != head);

    cout << endl;

}

**// Sort all element…**

void SORT(Node \*head){

    if (head == NULL){

        return;

    }

    Node \*ptr = head;

    do{

        Node \*cpt = ptr->link;

        while (cpt != head){

            if (ptr->data > cpt->data){

                swap(ptr->data, cpt->data);

            }

            cpt = cpt->link;

        }

        ptr = ptr->link;

    } while (ptr != head);

    display(head);

    cout << "\nThe link list is sorted!!\n";

    \_sleep(2000);

}

**// Search one element**

void search(Node \*head, int element){

    if (head == NULL){

        return;

    }

    Node \*ptr = head;

    int p = 0;

    do{

        if (ptr->data == element){

            cout << "\nThe element " << element << " is found!!\n";

            p++;

            \_sleep(2000);

            break;

        }

        ptr = ptr->link;

    } while (ptr != head);

    if (p == 0){

        cout << "\nThe element " << element << " is NOT found!!\n";

        \_sleep(2000);

    }

}

int main(){

    Node \*head = NULL;

    while (1){

        system("cls");

        display(head);

        cout << "\nEnter I for insert!!\n";

        cout << "Enter D for delete!!\n";

        cout << "Enter S for sorting!!\n";

        cout << "Enter F for searching!!\n";

        cout << "Enter E for exit!!\n";

        cout << "Enter your choice: ";

        char ch;

        cin >> ch;

        if (ch == 'i' || ch == 'I'){

            if (cunt == 0){

                cout << "Enter element for insert: ";

                int element;

                cin >> element;

                head = insertFirst(head, element);

                cunt++;

            }

            else{

                system("cls");

                display(head);

                cout << "\nEnter F for insert first!\n";

                cout << "Enter L for insert last!\n";

                cout << "Enter N for insert n'th position!\n";

                cout << "Enter your choice: ";

                char c;

                cin >> c;

                if (c == 'f' || c == 'F'){

                    cout << "Enter element for insert first: ";

                    int element;

                    cin >> element;

                    head = insertFirst(head, element);

                }

                else if (c == 'l' || c == 'L'){

                    cout << "Enter element for insert last: ";

                    int element;

                    cin >> element;

                    head = insertLast(head, element);

                }

                else if (c == 'n' || c == 'N'){

                    cout << "Enter index number: ";

                    int idx;

                    cin >> idx;

                    cout << "Enter element for insert " << idx << " position: ";

                    int element;

                    cin >> element;

                    head = insertCustom(head, element, idx);

                }

                else  cout << "Invalid input Try again!\n";

            }

        }

        else if (ch == 'd' || ch == 'D'){

            system("cls");

            display(head);

            if (cunt <= 0){

                cout << "SORRY!! there is no element for delete!\n";

                cunt = 0;

                \_sleep(1200);

                continue;

            }

            cout << "\nEnter F for delete first!!\n";

            cout << "Enter L for delete last!!\n";

            cout << "Enter N for delete n'th position!!\n";

            cout << "Enter your choice: ";

            char ch;

            cin >> ch;

            if (ch == 'f' || ch == 'F'){

                head = deleteFirst(head);

            }

            else if (ch == 'l' || ch == 'L'){

                head = deleteLast(head);

            }

            else if (ch == 'n' || ch == 'N'){

                cout << "Enter index for delete: ";

                int idx; cin >> idx;

                head = deleteCustom(head, idx);

            }

            else  cout << "Invalid input Try again!\n";

        }

        else if (ch == 's' || ch == 'S'){

            SORT(head);

        }

        else if (ch == 'f' || ch == 'F'){

            cout << "Which element you want to search: ";

            int element;

            cin >> element;

            search(head, element);

        }

        else if (ch == 'e' || ch == 'E'){

            cout << "\nCode is exited!!\n";

            \_sleep(1000);

            break;

        }

        else{

            cout << "\nInvalid input Try again!!\n";

            \_sleep(1000);

        }

    }

}

**// Doubly Link List(insert, delete,sorting,searching,traversing,exit).cpp**

#include<bits/stdc++.h>

using namespace std;

int cunt = 0;

struct Node{

    int data;

    Node \*next;

    Node \*prev;

};

**// Insert First......**

Node \*insertFirst(Node \*head,int data){

    Node \*newNode = new Node;

    newNode->data = data;

    newNode->next = head;

    newNode->prev = NULL;

    if (head != NULL) {

        head->prev = newNode;

    }

    head = newNode;

    cunt++;

    return head;

}

**// Insert Last.....**

Node \*insertLast(Node \*head,int data){

    Node \*newNode = new Node;

    newNode->data = data;

    Node \*ptr = head;

    while(ptr->next != NULL) {

        ptr = ptr->next;

    }

    ptr->next = newNode;

    newNode->prev = ptr;

    newNode->next = NULL;

    cunt++;

    return head;

}

**// insert Custom......**

Node \*insertCustom(Node \*head,int data, int index){

    Node \*newNode = new Node;

    newNode->data = data;

    if(index==1){

        newNode->next = head;

        newNode->prev = NULL;

        if (head != NULL) {

            head->prev = newNode;

        }

        head = newNode;

        cunt++;

        return head;

    }

    if(head == NULL){

        newNode->next = head;

        newNode->prev = NULL;

        head = newNode;

        cunt++;

        return head;

    }

    Node \*ptr = head;

    for (int i = 1; i < index - 1 && ptr->next != NULL; i++){

        ptr = ptr->next;

    }

    newNode->next = ptr->next;

    newNode->prev = ptr;

    ptr->next = newNode;

    cunt++;

    return head;

}

**// Delete First......**

Node \*deleteFirst(Node \*head){

    if (head == NULL) {

        cout << "The list is empty." << endl;

        return NULL;

    }

    head = head->next;

    if (head != NULL) {

        head->prev = NULL;

    }

    if(cunt>0) cunt--;

    else cunt = 0;

    return head;

}

**// Delete Last.......**

Node \*deleteLast(Node \*head){

     if (head->next == NULL) {

        cout << "The list is empty." << endl;

        return NULL;

    }

    Node \*ptr = head;

    while(ptr->next->next != NULL){

        ptr=ptr->next;

    }

    ptr->next = NULL;

    if(cunt>0) cunt--;

    else cunt = 0;

    return head;

}

**// Delete custom.....**

Node \*deleteCustom(Node \*head,int index){

     if (head == NULL) {

        cout << "The list is empty." << endl;

        return NULL;

    }

    if (index == 1) {

        head = deleteFirst(head);

        return head;

    }

    Node\* ptr = head;

    for (int i = 0; i < index-1 && ptr->next != NULL; i++) {

        ptr = ptr->next;

    }

    if (ptr == NULL) {

        cout << "Index out of range." << endl;

        return head;

    }

    ptr->prev->next = ptr->next;

    if (ptr->next != NULL) {

        ptr->next->prev = ptr->prev;

    }

    if(cunt>0) cunt--;

    else cunt = 0;

    return head;

}

**// Print all element.....**

void display(Node \*head){

    Node \*ptr = head;

    if(head == NULL) {

        cout << "Doubly next list: empty\n";

    }

    else{

        cout << "Doubly next list: ";

        while (ptr != NULL) {

            cout << ptr->data << " ";

            ptr = ptr->next;

        }

        cout << endl;

    }

}

**// Sort all element.....**

void SORT(Node \*head){

    Node \*ptr = head,\*cpt;

    while(ptr != NULL){

        cpt = ptr->next;

        while(cpt!= NULL){

            if(ptr->data > cpt->data){

                swap(ptr->data, cpt->data);

            }

            cpt = cpt->next;

        }

        ptr = ptr->next;

    }

    display(head);

    cout << "\nThe link list is sorted!!\n";

    \_sleep(2000);

}

**// Search one element.....**

void search(Node \*head,int element){

    Node \*ptr = head;

    int p=0;

    while(ptr != NULL){

        if(ptr->data == element){

            cout << "\nThe element "<<element << " is found!!\n";

            p++;

            \_sleep(2000);

            break;

        }

        ptr = ptr->next;

    }

    if(p==0){

        cout << "\nThe element "<<element << " is NOT found!!\n";

        \_sleep(2000);

    }

}

int main(){

    Node \*head = NULL;

    while (1){

        system("cls");

        display(head);

        cout << "\nEnter I for insert!!\n";

        cout << "Enter D for delete!!\n";

        cout << "Enter S for sorting!!\n";

        cout << "Enter F for searching!!\n";

        cout << "Enter E for exit!!\n";

        cout << "Enter your choice: ";

        char ch; cin >> ch;

        if (ch == 'i' || ch == 'I'){

            if (cunt == 0){

                cout << "Enter element for insert: ";

                int element;  cin >> element;

                head=insertFirst(head, element);

                cunt++;

            }

            else{

                system("cls");

                display(head);

                cout << "\nEnter F for insert first!\n";

                cout << "Enter L for insert last!\n";

                cout << "Enter N for insert n'th position!\n";

                cout << "Enter your choice: ";

                char c;  cin >> c;

                if (c == 'f' || c == 'F'){

                    cout << "Enter element for insert first: ";

                    int element; cin >> element;

                    head=insertFirst(head, element);

                }

                else if (c == 'l' || c == 'L'){

                    cout << "Enter element for insert last: ";

                    int element; cin >> element;

                    head=insertLast(head, element);

                }

                else if (c == 'n' || c == 'N'){

                    cout << "Enter index number: ";

                    int idx; cin >> idx;

                    cout << "Enter element for insert " << idx << " position: ";

                    int element;  cin >> element;

                    head=insertCustom(head, element, idx);

                }

                else    cout << "Invalid input Try again!\n";

            }

        }

        else if (ch == 'd' || ch == 'D'){

            system("cls");

            display(head);

            if (cunt <= 0){

                cout << "SORRY!! there is no element for delete!\n";

                cunt = 0;

                \_sleep(1200);

                continue;

            }

            cout << "\nEnter F for delete first!!\n";

            cout << "Enter L for delete last!!\n";

            cout << "Enter N for delete n'th position!!\n";

            cout << "Enter your choice: ";

            char ch; cin >> ch;

            if (ch == 'f' || ch == 'F'){

                head=deleteFirst(head);

            }

            else if (ch == 'l' || ch == 'L'){

                head=deleteLast(head);

            }

            else if (ch == 'n' || ch == 'N'){

                cout << "Enter index for delete: ";

                int idx; cin >> idx;

                head=deleteCustom(head, idx);

            }

            else   cout << "Invalid input Try again!\n";

        }

        else if(ch == 's' || ch == 'S'){

            SORT(head);

        }

        else if(ch == 'f' || ch == 'F'){

            cout << "Which element you want to search: ";

            int element;    cin >> element;

            search(head, element);

        }

        else if(ch == 'e' || ch == 'E'){

            cout << "\nCode is exited!!\n";

            \_sleep(1000);

            break;

        }

        else{

            cout << "\nInvalid input Try again!!\n";

            \_sleep(1000);

        }

    }

}