**Submitted by : Roshni raza**

**Submitted to : Mam Rehana**

**Sap id :**

**Task 1:**

#include <iostream>

using namespace std;

struct Node {

int data;

Node \*next;

Node \*prev;

};

Node \*head = NULL;

void insert(int n) {

Node \*newNode = new Node;

newNode->data = n;

if (head == NULL) {

newNode->next = newNode;

newNode->prev = newNode;

head = newNode;

} else {

newNode->next = head;

newNode->prev = head->prev;

head->prev->next = newNode;

head->prev = newNode;

head = newNode;

}

}

void print() {

if (head == NULL) {

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

return;

}

cout << "Data elements in the list are: ";

Node \*temp = head;

do {

cout << temp->data << " ";

temp = temp->next;

} while (temp != head);

cout << endl;

}

void DeleteAtBeginning() {

if (head == NULL) {

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

return;

}

if (head->next == head) {

delete head;

head = NULL;

} else {

Node \*temp = head;

head = head->next;

head->prev = temp->prev;

temp->prev->next = head;

delete temp;

}

}

int main() {

insert(5);

insert(4);

insert(3);

insert(2);

insert(1);

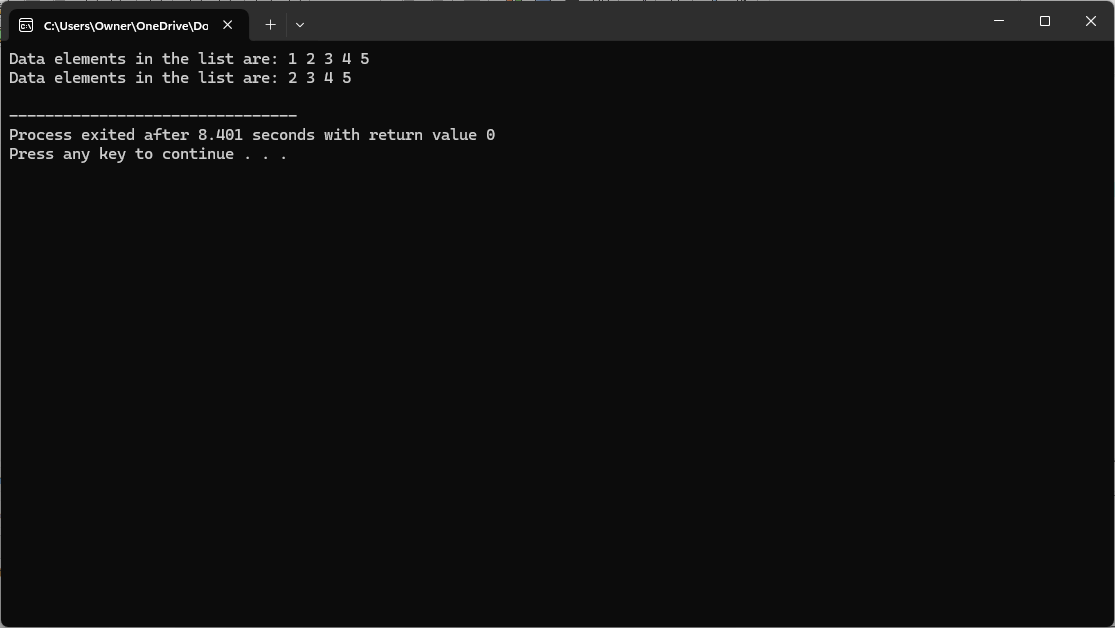
print();

DeleteAtBeginning();

print();

return 0;

}

****

**Task 2 :**

#include <iostream>

using namespace std;

struct Node {

int data;

Node \*next;

Node \*prev;

};

Node \*head = NULL;

void insert(int n) {

Node \*newNode = new Node;

newNode->data = n;

if (head == NULL) {

newNode->next = newNode;

newNode->prev = newNode;

head = newNode;

} else {

newNode->next = head;

newNode->prev = head->prev;

head->prev->next = newNode;

head->prev = newNode;

head = newNode;

}

}

void print() {

if (head == NULL) {

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

return;

}

cout << "Data elements in the list are: ";

Node \*temp = head;

do {

cout << temp->data << " ";

temp = temp->next;

} while (temp != head);

cout << endl;

}

void DeleteAtSpecificPosition(int position) {

if (head == NULL) {

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

return;

}

Node \*temp = head;

int count = 1;

if (position == 1) {

if (head->next == head) {

delete head;

head = NULL;

return;

} else {

Node \*temp = head;

head = head->next;

head->prev = temp->prev;

temp->prev->next = head;

delete temp;

return;

}

}

while (temp->next != head && count < position) {

temp = temp->next;

count++;

}

if (count != position) {

cout << "Position out of range!" << endl;

return;

}

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

delete temp;

}

int main() {

insert(5);

insert(4);

insert(3);

insert(2);

insert(1);

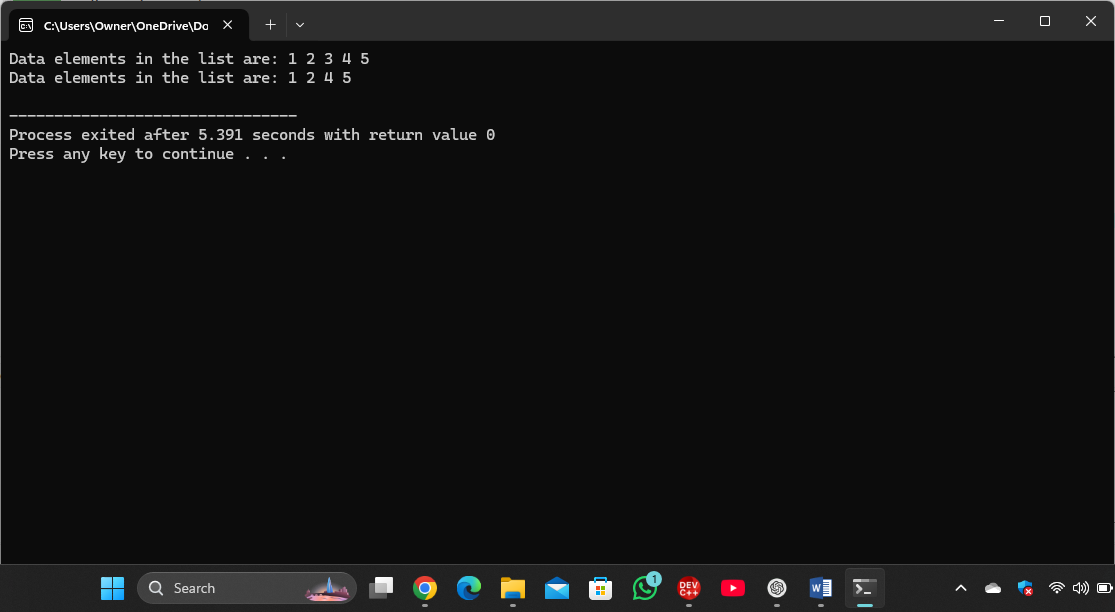
print();

DeleteAtSpecificPosition(3);

print();

return 0;

}

****

**Task 3:**

#include <iostream>

using namespace std;

struct Node {

int data;

Node \*next;

Node \*prev;

};

Node \*head = NULL;

void insert(int n) {

Node \*newNode = new Node;

newNode->data = n;

if (head == NULL) {

// If the list is empty, point next and prev to the newNode itself.

newNode->next = newNode;

newNode->prev = newNode;

head = newNode;

} else {

// Inserting at the beginning of the circular doubly linked list.

newNode->next = head;

newNode->prev = head->prev;

head->prev->next = newNode;

head->prev = newNode;

head = newNode;

}

}

void print() {

if (head == NULL) {

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

return;

}

cout << "Data elements in the list are: ";

Node \*temp = head;

do {

cout << temp->data << " ";

temp = temp->next;

} while (temp != head);

cout << endl;

}

void DeleteAtEnd() {

if (head == NULL) {

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

return;

}

// Check if there's only one node in the list

if (head->next == head) {

delete head;

head = NULL;

return;

}

// Traverse to the last node

Node \*last = head->prev;

// Make the second to last node the new last node

last->prev->next = head;

head->prev = last->prev;

// Delete the last node

delete last;

}

int main() {

insert(5);

insert(4);

insert(3);

insert(2);

insert(1);

print();

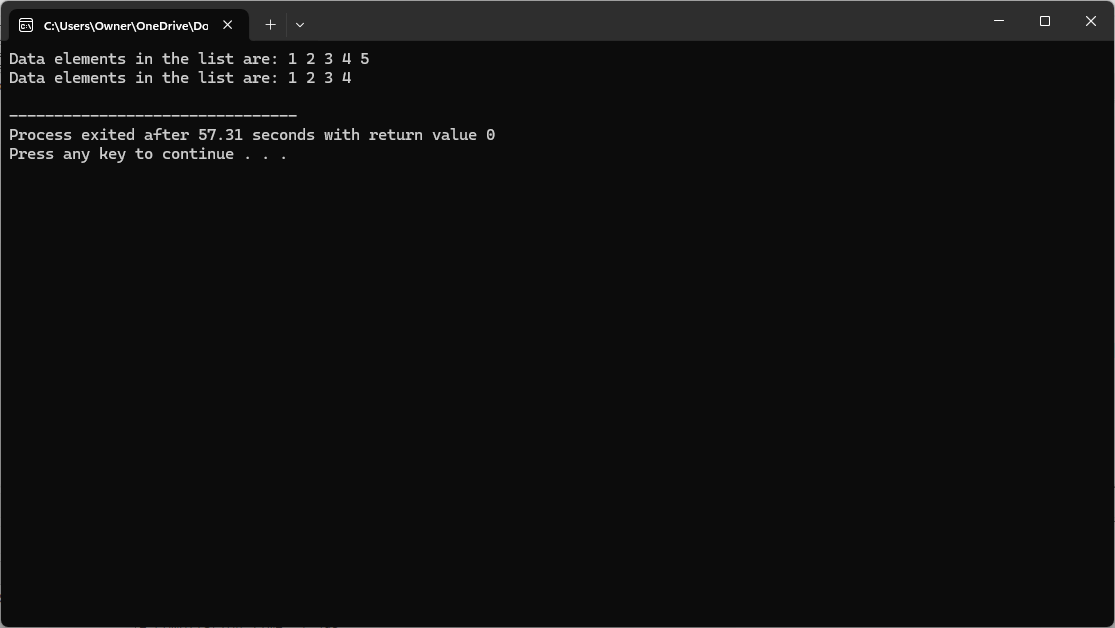
// Delete the last node

DeleteAtEnd();

print();

return 0;

}

****

**Task 4:**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* createNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertAtStart(Node\*\* head, int data) {

Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

newNode->next = \*head;

} else {

Node\* lastNode = \*head;

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

lastNode = lastNode->next;

}

lastNode->next = newNode;

newNode->next = \*head;

\*head = newNode;

}

}

void printList(Node\* head) {

if (head == NULL) {

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

return;

}

Node\* temp = head;

do {

cout << temp->data << " ";

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = NULL;

insertAtStart(&head, 10);

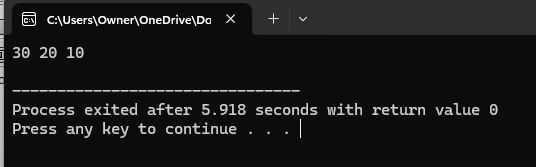
insertAtStart(&head, 20);

insertAtStart(&head, 30);

printList(head);

return 0;

}

****

**Task 5 :**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* createNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertAtSpecificPoint(Node\*\* head, int data, int position) {

Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

newNode->next = \*head;

} else if (position == 0) {

Node\* lastNode = \*head;

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

lastNode = lastNode->next;

}

lastNode->next = newNode;

newNode->next = \*head;

\*head = newNode;

} else {

Node\* temp = \*head;

int count = 0;

while (count < position - 1 && temp->next != \*head) {

temp = temp->next;

count++;

}

if (temp->next == \*head) {

temp->next = newNode;

newNode->next = \*head;

} else {

Node\* nextNode = temp->next;

temp->next = newNode;

newNode->next = nextNode;

}

}

}

void printList(Node\* head) {

if (head == NULL) {

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

return;

}

Node\* temp = head;

do {

cout << temp->data << " ";

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = NULL;

insertAtSpecificPoint(&head, 10, 0);

insertAtSpecificPoint(&head, 20, 1);

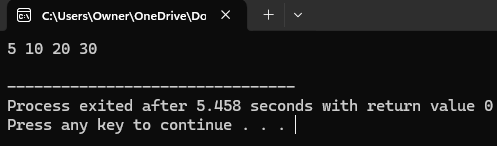
insertAtSpecificPoint(&head, 30, 2);

insertAtSpecificPoint(&head, 5, 0);

printList(head);

return 0;

}

****

**Task 6 :**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* createNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

newNode->next = NULL;

return newNode;a

}

void insertAtEnd(Node\*\* head, int data) {

Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

newNode->next = \*head;

} else {

Node\* lastNode = \*head;

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

lastNode = lastNode->next;

}

lastNode->next = newNode;

newNode->next = \*head;

}

}

void printList(Node\* head) {

if (head == NULL) {

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

return;

}

Node\* temp = head;

do {

cout << temp->data << " ";

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = NULL;

insertAtEnd(&head, 10);

insertAtEnd(&head, 20);

insertAtEnd(&head, 30);

printList(head);

return 0;

}

