**(ASSIGNMENT # 02…… SEMESTER FALL-2023)**

**BY**

**RIZWAN AKRAM**

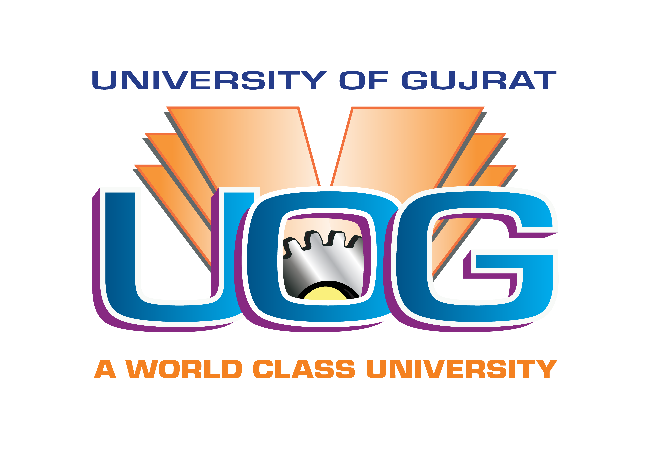
**DATA STRUCTURE AND ALGORITHMS**

**ROLL # 22011556-041**

**BS IT (Section A)**

**Submitted to Sir Azib Mahmood**

**Department of IT**

****

**UNIVERSITY OF GUJRAT**

**LINKLIST OPERATIONS**

**CODE**

**#include <iostream>**

**using namespace std;**

**class Node{**

**public:**

**int data;**

**Node\* next;**

**Node(int val) {**

**data = val;**

**next = NULL;**

**}**

**void insertHead(Node\* &head,int data) {**

**Node\* new\_node = new Node(data);**

**new\_node->next = head;**

**head = new\_node;**

**}**

**void deleteHead(Node\* &head) {**

**if (head == NULL) {**

**cout << "List is empty. Cannot delete from an empty list." << endl;**

**return;**

**}**

**Node\* ptr = head;**

**head = head->next;**

**delete (ptr);**

**}**

**void insertEnd(Node\* &head,int data) {**

**Node\* new\_node = new Node(data);**

**if (head == NULL) {**

**head = new\_node;**

**return;**

**}**

**Node\* current = head;**

**while (current->next != NULL) {**

**current = current->next;**

**}**

**current->next = new\_node;**

**}**

**void deleteEnd(Node\* &head) {**

**if (head == NULL) {**

**cout << "List is empty. Cannot delete from an empty list." << endl;**

**return;**

**}**

**Node\* ptr = head;**

**Node\* ptr2 = head->next;**

**while (ptr2->next != NULL) {**

**ptr = ptr->next;**

**ptr2 = ptr2->next;**

**}**

**ptr->next = NULL;**

**delete (ptr2);**

**}**

**void insertPosition(Node\* &head,int data, int pos) {**

**if (pos == 0) {**

**insertHead(head,data);**

**return;**

**}**

**Node\* new\_node = new Node(data);**

**Node\* temp = head;**

**int current\_pos = 0;**

**while (current\_pos != pos - 1) {**

**temp = temp->next;**

**current\_pos++;**

**}**

**new\_node->next = temp->next;**

**temp->next = new\_node;**

**}**

**void deletePosition(Node\* &head,int pos) {**

**if (head == NULL) {**

**cout << "List is empty. Cannot delete from an empty list." << endl;**

**return;**

**}**

**Node\* ptr = head;**

**Node\* ptr2 = head->next;**

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

**ptr = ptr->next;**

**ptr2 = ptr2->next;**

**}**

**ptr->next = ptr2->next;**

**delete (ptr2);**

**}**

**void search(Node\* &head, int value) {**

**Node\* temp = head;**

**int position = 0;**

**while (temp != NULL) {**

**if (temp->data == value) {**

**cout << "Value " << value << " found at position " << position << "." << endl;**

**return;**

**}**

**temp = temp->next;**

**position++;**

**}**

**cout << "Value " << value << " not found in the list." << endl;**

**}**

**void update(Node\* &head, int oldValue, int newValue) {**

**Node\* temp = head;**

**while (temp != NULL) {**

**if (temp->data == oldValue) {**

**temp->data = newValue;**

**cout << "Value updated successfully." << endl;**

**return;**

**}**

**temp = temp->next;**

**}**

**cout << "Value " << oldValue << " not found in the list. Update failed." << endl;**

**}**

**void display(Node\* &head) {**

**Node\* temp = head;**

**while (temp != NULL) {**

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

**temp = temp->next;**

**}**

**cout << endl;**

**}**

**};**

**int main() {**

**Node\* n;**

**Node\* head;**

**int option, data, position;**

**do {**

**cout << "What operation do you want to perform? Select Option number. Enter 0 to exit." << endl;**

**cout << "1. Insert at the beginning" << endl;**

**cout << "2. Delete from the beginning" << endl;**

**cout << "3. Insert at the end" << endl;**

**cout << "4. Delete from the end" << endl;**

**cout << "3. Insert at a specific position" << endl;**

**cout << "6. Delete from a specific position" << endl;**

**cout << "7. Search at a specific position" << endl;**

**cout << "8. Update at a specific position" << endl;**

**cout << "9. Display the list" << endl;**

**cout << "0. Exit" << endl;**

**cin >> option;**

**switch (option) {**

**case 0:**

**cout << "Exiting program." << endl;**

**break;**

**case 1:**

**cout << "Enter a value to insert at the beginning: ";**

**cin >> data;**

**n->insertHead(head,data);**

**break;**

**case 2:**

**n->deleteHead(head);**

**cout << "Deleted from the beginning." << endl;**

**break;**

**case 3:**

**cout << "Enter a value to insert at the end: ";**

**cin >> data;**

**n->insertEnd(head,data);**

**break;**

**case 4:**

**n->deleteEnd(head);**

**cout << "Deleted from the end." << endl;**

**break;**

**case 5:**

**cout << "Enter a value to insert: ";**

**cin >> data;**

**cout << "Enter the position to insert at: ";**

**cin >> position;**

**n->insertPosition(head,data, position);**

**break;**

**case 6:**

**cout << "Enter the position to delete: ";**

**cin >> position;**

**n->deletePosition(head,position);**

**cout << "Deleted from position " << position << "." << endl;**

**break;**

**case 7:**

**cout << "Enter a value to search: ";**

**cin >> data;**

**n->search(head, data);**

**break;**

**case 8:**

**int oldValue,newValue;**

**cout << "Enter the value to update: ";**

**cin >> oldValue;**

**cout << "Enter the new value: ";**

**cin >> newValue;**

**n->update(head, oldValue, newValue);**

**break;**

**case 9:**

**cout << "Displaying the list: ";**

**n->display(head);**

**break;**

**default:**

**cout << "Invalid option. Please enter a valid option." << endl;**

**}**

**} while (option != 0);**

**delete n;**

**return 0;**

**}**

**OUTPUT**

