UNIVERSITY OF GUJRAT

Assignment No. 2

Submitted to:

Mr. Azib Mahood

Submitted by:

Abdul Rehman

Enrollment:

22011556-065

Section:

IT-22-B

Course code:

IT-209

Course name:

Data Structure and Algorithm

PROGRAM OF LINKED LIST

```
#include <iostream>
using namespace std;
class Node {
public:
   int data;
   Node* next;

   Node(int val) {
      data = val;
      next = NULL;
   }
};

class LinkedList {
private:
   Node* head;

public:
   LinkedList() {
      head = NULL;
   }

   void insertAtBeginning(int val) {
      Node* newNode = new Node(val);
      newNode->next = head;
      head = newNode;
      cout<<"Successfully insert"<<endl;
      cout<<endl;

   }

   void insertAtPosition(int val, int position) {
      Node* newNode = new Node(val);
      if (position == 1) {
         newNode->next = head;
         head = newNode;
         return;
      }
```

```
    Node* temp = head;
    for (int i = 1; i < position - 1 && temp; i++) {
        temp = temp->next;
    }

    if (temp) {
        newNode->next = temp->next;
        temp->next = newNode;
    } else {
        std::cout << "Invalid position." << std::endl;
    }
}
void deleteFromBeginning() {
    if (head) {
        Node* temp = head;
        head = head->next;
        delete temp;
        cout<<"Successfully Delete"<<endl;
    } else {
        std::cout << "List is empty." << std::endl;
    }
}


void deleteFromEnd() {
    if (head) {
        if (head->next == NULL) {
            delete head;
            head = NULL;
            return;
        }

        Node* temp = head;
        while (temp->next->next) {
            temp = temp->next;
        }

        delete temp->next;
        temp->next = NULL;
    } else {
        cout << "List is empty." <<endl;
    }
}
```

```
void deleteFromPosition(int position) {
   if (head) {
      if (position == 1) {
         Node* temp = head;
         head = head->next;
         delete temp;
         return;
      }

      Node* temp = head;
      for (int i = 1; i < position - 1 && temp; i++) {
         temp = temp->next;
      }

      if (temp && temp->next) {
         Node* toDelete = temp->next;
         temp->next = temp->next->next;
         delete toDelete;
      } else {
         cout<< "Invalid position." <<endl;
      }
   } else {
      cout<< "List is empty." <<endl;
   }
}
void searchAndUpdate(int oldValue, int newValue) {
   Node* temp = head;
   while (temp) {
      if (temp->data == oldValue) {
         temp->data = newValue;
         return;
      }
      temp = temp->next;
   }

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


void display() {
   Node* temp = head;
   while (temp) {
```

```
            std::cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    LinkedList myList;
    int choice;
    cout<<"Warnig! Value Store in the Start of Link or Program is NULL"<<endl;

    do {
        cout << "\nLinked List Operations:\n";
        cout << "1. Insert at the beginning\n";
        cout << "2. Insert at any position\n";
        cout << "3. Delete from the beginning\n";
        cout << "4. Delete from the end\n";
        cout << "5. Delete from any position\n";
        cout << "6. Search and update at any point\n";
        cout << "7. Display the linked list\n";
        cout << "8. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        system("cls");
        switch (choice) {
            case 1: {
                int value;
                cout << "Enter the value to insert: ";
                cin >> value;
                myList.insertAtBeginning(value);
                break;
            }
            case 2: {
                int value, position;
                cout << "Enter the value to insert: ";
                cin >> value;
                cout << "Enter the position to insert: ";
                cin >> position;
                myList.insertAtPosition(value, position);
                break;
            }
            case 3:
                myList.deleteFromBeginning();
```

```
            break;
        case 4:
            myList.deleteFromEnd();
            break;
        case 5: {
            int position;
            cout << "Enter the position to delete: ";
            cin >> position;
            myList.deleteFromPosition(position);
            break;
        }
        case 6: {
            int oldValue, newValue;
            cout << "Enter the value to search: ";
            cin >> oldValue;
            cout << "Enter the new value: ";
            cin >> newValue;
            myList.searchAndUpdate(oldValue, newValue);
            break;
        }
        case 7:
            myList.display();
            break;
        case 8:
            cout << "Exiting the program.\n";
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
      }
   } while (choice != 8);


   return 0;
}
```