Name :M Ijaz

Submitted to

Sir Rasikh

Roll no . 073

Subject . Lab Data structure

Lab . 4

Topic. Singly link list

Questions no 1. Write a function to insert a node at a specific position in a singly linked list, ensuring valid position handling.

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;
    Node(int value) {
        data = value;
        next = nullptr;
    }
};

class SinglyLinkedList {
public:
```

```
Node* head;
SinglyLinkedList() {
    head = nullptr;
}

void insertAtStart(int data) {
    Node* newNode = new Node(data);
    newNode->next = head;
    head = newNode;
}

void insertAtEnd(int data) {
    Node* newNode = new Node(data);
    if (head == nullptr) {
        head = newNode;
        return;
    }
    Node* last = head;
    while (last->next != nullptr) {
        last = last->next;
    }
    last->next = newNode;
}

void insertAtPosition(int data, int position) {
    if (position < 1) {
        cout << "Invalid position!" << endl;
```

```
        return;
    }
    if (position == 1) {
        insertAtStart(data);
        return;
    }
    Node* newNode = new Node(data);
    Node* temp = head;
    for (int i = 1; i < position - 1; i++) {
        if (temp == nullptr) {
            cout << "Position exceeds list length!" << endl;
            return;
        }
        temp = temp->next;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

void display() {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
```

```
            temp = temp->next;
        }
        cout << "None" << endl;
    }
};

int main() {
    SinglyLinkedList list;
    list.insertAtStart(10);
    list.display();
    list.insertAtEnd(20);
    list.display();
    list.insertAtPosition(15, 2);
    list.display();
    list.insertAtPosition(25, 4);
    list.display();
    list.insertAtPosition(30, 10);
    list.display();
    return 0;
```


```
10 -> None
10 -> 20 -> None
10 -> 15 -> 20 -> None
10 -> 15 -> 20 -> 25 -> None
Position exceeds list length!
10 -> 15 -> 20 -> 25 -> None


=== Code Execution Successful ===
```

}