Lab 4: Understanding the Linked List

A. Explain static and dynamic implementation of list with suitable example.

Static implementation of list: In this structure, the size of the structure is fixed. The content of the data structure can be modified but without changing the memory space allocated to it.
For example: Array

Dynamic implementation of list: In this structure, the size of the structure is not fixed and can be modified during the operations performed on it. It facilitate change of data structure in the run time.
For example: Singly linked list, Doubly linked list, vector, Stack, Queue

B. Why linked list come into existence?

A linked list is a linear data structure where each element is a separate object. It came into existence because of limitation of static list (array). Limitation of array are:
1. The size of the arrays is fixed i.e. the upper limit on the number of elements should be known in advance.
2. Inserting a new element in an array of elements is expensive because room has to be created for the new elements and to create room existing elements have to shift.

Hence, to mitigate the limitations of array, linked list came into existence. It provides dynamic size, easy way of insertion or deletion and memory utilization is efficient than array.

Implementation of operation of Singly Linked list using C++:

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

class Singlylist{
	int data;
	Singlylist *next;
public:
	void insertstart(Singlylist **Head_ref, int item){
		Singlylist *slist = new Singlylist();
		cout<<"slist address: "<<slist<<endl;
		slist->data = item;
		slist->next = *Head_ref;
		*Head_ref = slist;
		cout<<"Slist->data: "<< slist->data<<"\t";
		cout<<"Slist->next: "<< slist->next<<"\t"<<endl;
		cout<<"*Head_ref: "<< *Head_ref<<endl;
		cout<<endl;
	}
	void insertafter(Singlylist **Head_ref,int after, int item){
		Singlylist *ptr = *Head_ref;
		Singlylist *slist = new Singlylist();
		cout << "slist address: " << slist << endl;
		slist->data = item;
		while(ptr->data != after){
			ptr = ptr->next;
		}
		slist->next = ptr->next;
		ptr->next = slist;
		cout << "Slist->data: " << slist->data << "\t";
		cout << "Slist->next: " << slist->next << "\t" << endl;
		cout << "previous->next: " << ptr->next << "\t" << endl;
		cout << "*Head_ref: " << *Head_ref << endl;
		cout << endl;
	}

	void insertbefore(Singlylist **Head_ref, int before, int item){
		Singlylist *slist = new Singlylist();
		Singlylist *ptr = *Head_ref;
		Singlylist *preptr = ptr;
		cout << "Address of slist: " << slist << endl;
		slist->data = item;
		while(ptr->data != before){
			preptr = ptr;
			ptr = ptr-> next;
```

```
        }
        slist->next = ptr;
        preptr->next = slist;
        cout << "Slist->data: " << slist->data << "\t";
        cout << "Slist->next: " << slist->next << "\t" << endl;
        cout << "previous->next: " << preptr->next << "\t" << endl;
        cout << "*Head_ref: " << *Head_ref << endl;
        cout << endl;

}
void insertend(Singlylist **Head_ref, int item){
        Singlylist *slist = new Singlylist();
        Singlylist *ptr = *Head_ref;
        cout<<"slist address: "<<slist<<endl;
        slist->data = item;
        slist->next = NULL;
        if(*Head_ref == NULL) *Head_ref = slist;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = slist;
        cout << "Slist->data: " << slist->data << "\t";
        cout << "Slist->next: " << slist->next << "\t" << endl;
        cout << "previous->next: " << ptr->next << "\t" << endl;
        cout << "*Head_ref: " << *Head_ref << endl;
        cout << endl;
}

void removestart(Singlylist **Head_ref){
        Singlylist *ptr = *Head_ref;
        *Head_ref = ptr->next;
        delete ptr;
}
void removeafter(Singlylist **Head_ref, int after){
        Singlylist *ptr = *Head_ref;
        Singlylist *preptr = ptr;
        while(preptr->data != after){
                preptr = ptr;
                ptr = ptr->next;
        }
        preptr->next = ptr->next;
        delete ptr;
}
void removebefore(Singlylist **Head_ref, int before){
        Singlylist *postptr = *Head_ref;
        Singlylist *ptr = postptr;
        Singlylist *preptr = ptr;
        while(postptr->data != before){
                preptr = ptr;
                ptr = postptr;
                postptr = postptr->next;
        }
```

```
            preptr->next = postptr;
            delete ptr;
        }
        void removeend(Singlylist **Head_ref){
            Singlylist *ptr = *Head_ref;
            Singlylist *preptr = *Head_ref;
            while(ptr->next != NULL){
                preptr = ptr;
                ptr = ptr->next;
            }
            preptr->next = NULL;
            delete ptr;
        }
        void display(Singlylist *node){
            cout << "item in the list are: " << endl;
            while(node != NULL){
                cout << node->data << "\t";
                node = node->next;
            }
            cout << endl;
        }

};

int main(){
        Singlylist s1;
        Singlylist *Head = NULL;
        int val, after, before;
        char ch;
        cout << "1) INSERT START" << endl;
        cout << "2) INSERT AFTER" << endl;
        cout << "3) INSERT BEFORE" << endl;
        cout << "4) INSERTEND" << endl;
        cout << "5) REMOVE START" << endl;
        cout << "6) REMOVE AFTER" << endl;
        cout << "7) REMOVE BEFORE" << endl;
        cout << "8) REMOVE END" << endl;
        cout << "9) Display list" << endl;
        cout << "q) EXIT" << endl;
        do{
            cout << "Enter choice: ";
            cin >> ch;
            switch(ch){
            case '1':
                cout << "Enter the first value: " << endl;
                cin >> val;
                s1.insertstart(&Head, val);
                break;
            case '2':
                cout << "Enter the value after which you want to enter item: ";
```

```
            cin >> after;
            cout << "Enter the value you want to add: ";
            cin >> val;
            s1.insertafter(&Head, after, val);
            break;
        case '3':
            cout << "Enter the value before which you want to add item: ";
            cin >> before;
            cout << "Enter the value: ";
            cin >> val;
            s1.insertbefore(&Head, before, val);
            break;
        case '4':
            cout << "Enter the last value: " << endl;
            cin >> val;
            s1.insertend(&Head, val);
            break;
        case '5':
            s1.removestart(&Head);
            break;
        case '6':
            cout << "Enter the value after which you want to delete item: ";
            cin >> after;
            s1.removeafter(&Head, after);
            break;
        case '7':
            cout << "Enter the value before which you want to delete item: ";
            cin >> before;
            s1.removebefore(&Head, before);
            break;
        case '8':
            s1.removeend(&Head);
            break;
        case '9':
            s1.display(Head);
            break;
        case 'q':
            break;
        default:
            cout << "Please enter the correct choice" << endl;
            break;
        }
    }while(ch != 'q');
    return 0;
}
```

Implementation of insertion of new node at the beginning of doubly linked list using C++

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

class Doublylist{
	int data;
	Doublylist *next;
	Doublylist *prev;
public:
	void insertstart(Doublylist **Head_ref, int item){
		Doublylist *dlist = new Doublylist();
		Doublylist *ptr = *Head_ref;
		cout<<"dlist address: "<<dlist<<endl;
		dlist->data = item;
		dlist->next = *Head_ref;
		if(*Head_ref == NULL){
			ptr = dlist;
			ptr->prev = NULL;
		}
		else{
			dlist->next = *Head_ref;
			dlist->prev = NULL;
			ptr->prev = dlist;
		}

		*Head_ref = dlist;
		cout << "dlist->prev: " << dlist->prev << "\t";
		cout << "dlist->data: " << dlist->data << "\t";
		cout << "dlist->next: " << dlist->next << "\t"<<endl;
		cout << "*Head_ref: " << *Head_ref << endl;
		cout << endl;
	}
	void display(Doublylist *node){
		cout << "item in the list are: " << endl;
		while(node != NULL){
			cout << node->data << "\t";
			node = node->next;
		}
		cout << endl;
	}

};

int main(){
	Doublylist d1;
	Doublylist *Head = NULL;
```

```
        int val;
        char ch;
        cout << "1) INSERT START" << endl;
        cout << "7) Display list" << endl;
        cout << "8) EXIT" << endl;
        do{
                cout << "Enter choice: ";
                cin >> ch;
                switch(ch){
                case '1':
                        cout << "Enter the first value: " << endl;
                        cin >> val;
                        d1.insertstart(&Head, val);
                        break;
                case '7':
                        d1.display(Head);
                        break;
                case '8':
                        break;
                default:
                        cout << "Please enter the correct choice" << endl;
                        break;
                }
        }while(ch != '8');
        return 0;
}
```