LAB MANUAL

DATA STRUCTURE

Submitted by: Tahira Inam
Roll No: 23010101-105
Instructor: Azka Mir
3rd Semester (Blue)

Department of Software Engineering
University of Sialkot

Table of Contents

Introduction

This lab manual covers essential operations on data structures, specifically focusing on singly linked lists. It demonstrates insertion, deletion, searching, updating, and splitting operations, reinforcing understanding of linked list data structures.

Objectives

- Implement basic operations on a singly linked list.
- Develop problem-solving skills by manipulating linked list structures.
- Understand how to dynamically manage memory using pointers.

Exercises

Exercise 1.1: Singly Linked List Operations

Problem Statement:
Create a singly linked list to manage employee records for a company, including fields for employee number, name, salary, and department number. Implement the following functions:

1. Insert: Add a new employee record.
2. Delete: Remove an employee record.
3. Search: Find an employee by their number or department.
4. Modify: Update employee information.
5. Display: Show all employee records and the total count.

Exercise 1.2: Splitting a Linked List

Problem Statement:
Develop a function to split the singly linked list created in Exercise 1.1 into two separate lists based on the midpoint. Display both lists after splitting.

Source Code and Outputs

Exercise 1.1 Code

```
#include <iostream>

#include <string>

using namespace std;


struct Employee {

        //employee attributes

        int emp_num, dep_num;

        string name;
```

```
        float salary;
        Employee *next;
};
// class for handling employee functionality
class Emp_linked_list {
        private:
          Employee *head;
          Employee *tail;
          int count;
   public:
        //constructor
        Emp_linked_list() {
                head = NULL;
                tail = NULL;
                count = 0;
                }
        // Function for insertion
        void insert() {
                cout<<"\n\t----------INSERT----------\n";
                Employee * temp = new Employee;
                cout<<"\n\tEnter Employee Number: ";
                cin>>temp->emp_num;
                cout<<"\tEnter Employee Name: ";
                cin.ignore();
           getline(cin, temp->name);
           cout<<"\tEnter Salary: ";
```

```
    cin>>temp->salary;
    cout<<"\tEnter Demaprtment Number: ";
    cin>>temp->dep_num;

    temp->next = NULL;

    if (head == NULL){
            head = temp;
            tail = temp;
            }
            else {
                    tail->next = temp;
                    tail = temp;
            }
            count++;
}
// Function for display
void display() {
            cout<<"\n\t-----------DISPLAY-----------\n";
            Employee *temp = head;
            while (temp != NULL) {
                    cout<<"\n\n\tEmployee Number: "<<temp->emp_num;
               cout<<"\n\tEmployee Name: "<<temp->name;
      cout<<"\n\tSalary: "<<temp->salary;
      cout<<"\n\tDemaprtment Number: "<<temp->dep_num;
                    temp = temp->next;
```

```
        }
        cout<<"\n\n\tTotal Employees: "<<count<<"\n";
}
// Function for updation
void update() {
        cout<<"\n\t-----------UPDATE-----------\n";
        int f=0, num;
        Employee *temp = head;
        cout<<"\n\tEnter Employee/Department Number to Update: ";
        cin>>num;
        while (temp != NULL){
                if (temp->dep_num == num || temp->emp_num == num){
                        f = 1;
                        break;
                }
                temp = temp->next;
        }
        if (f==1) {
                int choice;
                do {
                        cout<<"\n\t----UPDATE----";
                        cout<<"\n\t0. Exit";
                        cout<<"\n\t1. Update Employee Number";
                        cout<<"\n\t2. Update Employee Name";
                        cout<<"\n\t3. Update Salary";
                        cout<<"\n\t4. Update Department";
```

```
cout<<"\n\t----------\n";
cout<<"\n\tEnter Choice: ";
cin>>choice;
switch (choice){
        case 0:
                cout<<"\n\tExiting...\n";
                break;
        case 1:
                cout<<"\n\tUpdate Employee Number: ";
cin>>temp->emp_num;
cout<<"\n\tEmployee Number Updated successfuly\n";
break;
        case 2:
                cout<<"\n\tUpdate Employee Name: ";
cin.ignore();
getline(cin, temp->name);
cout<<"\n\tName Updated successfuly\n";
break;
        case 3:
                cout<<"\n\tUpdate Salary: ";
cin>>temp->salary;
cout<<"\n\tSalary Updated successfuly\n";
break;
        case 4:
                cout<<"\n\tUpdate Department Number: ";
cin>>temp->dep_num;
```

```
                        cout<<"\n\tDepartment Number Updated successfuly\n";
                        break;
                    default:
                        cout<<"\n\tInvalid Choice. Try Again";
                        }
                } while (choice != 0);
        }
        else {
                cout<<"\n\tRECORD NOT FOUND\n";
        }
}

// Function for searching
void search() {
        cout<<"\n\t----------SEARCH----------\n";
        int f=0, num;
        Employee *temp = head;
        cout<<"\n\tEnter Employee/Department Number to search: ";
        cin>>num;
        while (temp != NULL){
                if (temp->dep_num == num || temp->emp_num == num){
                        f = 1;
                        break;
                }
                temp = temp->next;
        }
```

```
                if (f == 1) {
                        cout<<"\n\tRECORD FOUND\n";
                        cout<<"\n\tEmployee Number: "<<temp->emp_num;
                   cout<<"\n\tEmployee Name: "<<temp->name;
            cout<<"\n\tSalary: "<<temp->salary;
            cout<<"\n\tDemaprtment Number: "<<temp->dep_num;
                }
                else {
                        cout<<"\n\tRECORD NOT FOUND\n";
                }
        }
        // Function for deletion
        void Delete() {
                cout << "\n\t-----------DELETE-----------\n";
int num;
cout << "\n\tEnter Employee/Department Number to Delete: ";
cin >> num;

Employee *temp = head;
Employee *prev = NULL;

while (temp != NULL) {
   if (temp->emp_num == num || temp->dep_num == num) {
      if (prev == NULL) {
         // Deleting the head
         head = temp->next;
```

```
        } else {
            // Deleting a non-head node
            prev->next = temp->next;
        }

        //deleting the tail
        if (temp == tail) {
            tail = prev;
        }

        delete temp;
        count--;
        cout << "\n\tRecord Deleted Successfully\n";
        return;
      }

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

   cout << "\n\tRECORD NOT FOUND\n";
}
};

int main()
{
```

```
Emp_linked_list ell;


while (true)
{
	cout<<"\n\n\t----MENU----";
	cout<<"\n\t1. Insert";
	cout<<"\n\t2. Display";
	cout<<"\n\t3. Update";
	cout<<"\n\t4. Search";
	cout<<"\n\t5. Delete";
	cout<<"\n\t6. Exit";
	cout<<"\n\t-------------\n";
	int choice;
	cout<<"\n\tEnter Your Choice: ";
	cin>>choice;
	switch (choice)
	{
		case 1:
			ell.insert();
			break;
		case 2:
			ell.display();
			break;
		case 3:
			ell.update();
			break;
```

```
                case 4:
                        ell.search();
                        break;
                case 5:
                        ell.Delete();
                        break;
                case 6:
                        return 0;
                        break;
                default:
                        cout<<"\n\tInvalid Choice. Please try again\n";
            }
        }

        return 0;
}
```

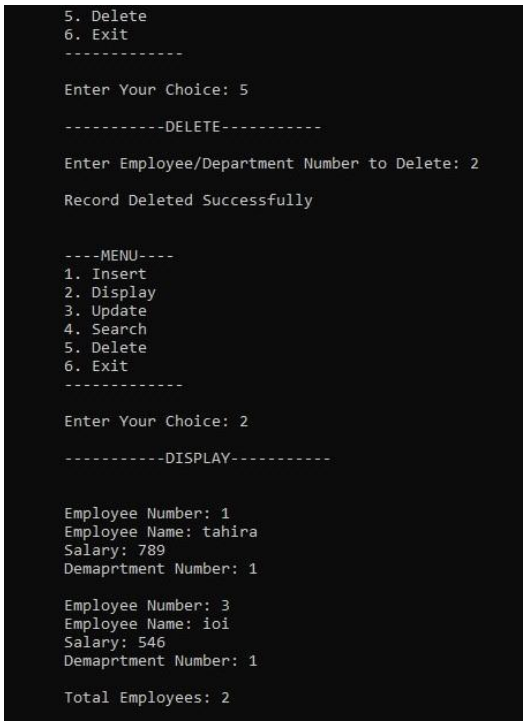
Output:

```
----MENU----
1. Insert
2. Display
3. Update
4. Search
5. Delete
6. Exit
-------------

Enter Your Choice: 1

-----------INSERT-----------

Enter Employee Number: 1
Enter Employee Name: tahira
Enter Salary: 789
Enter Demaprtment Number: 1
```

```
----MENU----
1. Insert
2. Display
3. Update
4. Search
5. Delete
6. Exit
-------------

Enter Your Choice: 2

-----------DISPLAY-----------


Employee Number: 1
Employee Name: tahira
Salary: 789
Demaprtment Number: 1

Employee Number: 2
Employee Name: fgf
Salary: 7896
Demaprtment Number: 1

Employee Number: 3
Employee Name: ioi
Salary: 546
Demaprtment Number: 1

Total Employees: 3
```

```
----MENU----
1. Insert
2. Display
3. Update
4. Search
5. Delete
6. Exit
-------------

Enter Your Choice: 4

-----------SEARCH-----------

Enter Employee/Department Number to search: 3

RECORD FOUND

Employee Number: 3
Employee Name: ioi
Salary: 546
Demaprtment Number: 1
```

```
----MENU----
1. Insert
2. Display
3. Update
4. Search
5. Delete
6. Exit
-------------

Enter Your Choice: 3

-----------UPDATE-----------

Enter Employee/Department Number to Update: 2

----UPDATE----
0. Exit
1. Update Employee Number
2. Update Employee Name
3. Update Salary
4. Update Department
-------------

Enter Choice: 2

Update Employee Name: aqsa inam

Name Updated successfuly

----UPDATE----
0. Exit
1. Update Employee Number
2. Update Employee Name
3. Update Salary
4. Update Department
-------------
```



Exercise 1.2 Code

```
#include <iostream>

#include <string>

using namespace std;


struct Employee {

   int emp_num, dep_num;

   string name;

   float salary;

   Employee *next;

};


class Emp_linked_list {

private:

   Employee *head;
```

```
    Employee *tail;
    int count;

public:
    Emp_linked_list() {
        head = NULL;
        tail = NULL;
        count = 0;
    }

    // Function for insertion
    void insert() {
        cout << "\n\t-----------INSERT-----------\n";
        Employee *temp = new Employee;
        cout << "\n\tEnter Employee Number: ";
        cin >> temp->emp_num;
        cout << "\tEnter Employee Name: ";
        cin.ignore();
        getline(cin, temp->name);
        cout << "\tEnter Salary: ";
        cin >> temp->salary;
        cout << "\tEnter Department Number: ";
        cin >> temp->dep_num;

        temp->next = NULL;
```

```
    if (head == NULL) {
        head = temp;
        tail = temp;
    } else {
        tail->next = temp;
        tail = temp;
    }
    count++;
}

// Function for display
void display() {
    cout << "\n\t-----------DISPLAY-----------\n";
    Employee *temp = head;
    while (temp != NULL) {
        cout << "\n\n\tEmployee Number: " << temp->emp_num;
        cout << "\n\tEmployee Name: " << temp->name;
        cout << "\n\tSalary: " << temp->salary;
        cout << "\n\tDepartment Number: " << temp->dep_num;
        temp = temp->next;
    }
    cout << "\n\n\tTotal Employees: " << count << "\n";
}

// Function to split the linked list into two new lists based on midpoint
void split() {
```

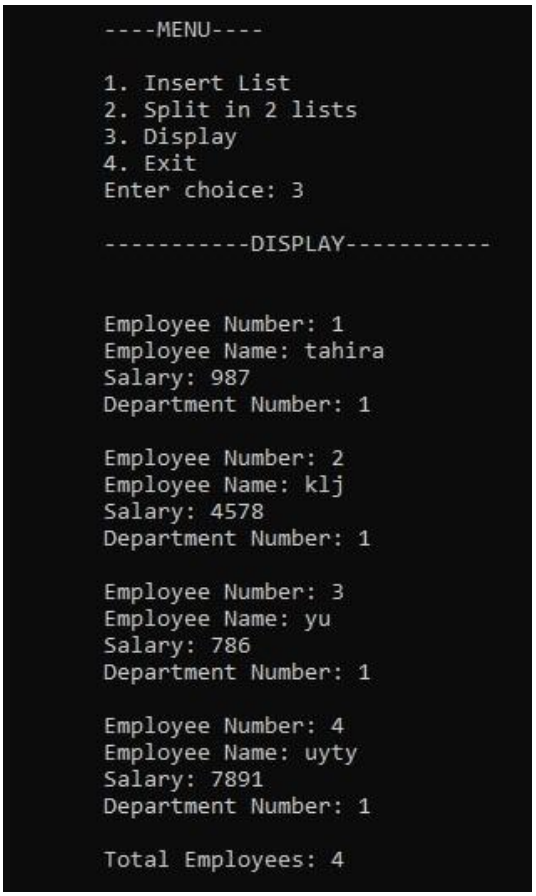
```
if (head == NULL) {
   cout << "\n\tList is empty, cannot split.\n";
   return;
}
int mid = count / 2;
Employee *temp = head;
Emp_linked_list list1;
Emp_linked_list list2;

// Fill list1 up to the midpoint
for (int i = 0; i < mid; i++) {
   Employee *newNode = new Employee;
   newNode->emp_num = temp->emp_num;
   newNode->name = temp->name;
   newNode->salary = temp->salary;
   newNode->dep_num = temp->dep_num;
   newNode->next = NULL;

   if (list1.head == NULL) {
      list1.head = newNode;
      list1.tail = newNode;
   } else {
      list1.tail->next = newNode;
      list1.tail = newNode;
   }
   list1.count++;
```

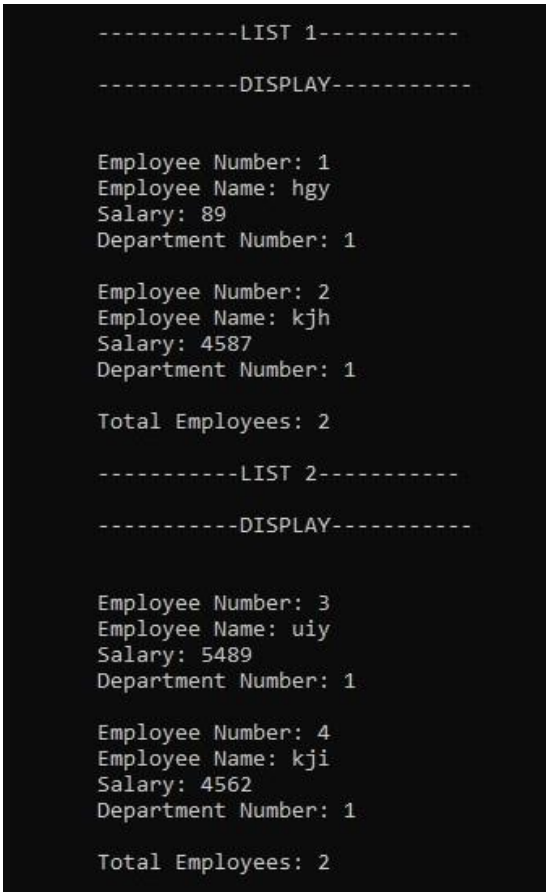
```
    temp = temp->next;

}

// Fill list2 with the remaining employees
while (temp != NULL) {
    Employee *newNode = new Employee;
    newNode->emp_num = temp->emp_num;
    newNode->name = temp->name;
    newNode->salary = temp->salary;
    newNode->dep_num = temp->dep_num;
    newNode->next = NULL;

    if (list2.head == NULL) {
        list2.head = newNode;
        list2.tail = newNode;
    } else {
        list2.tail->next = newNode;
        list2.tail = newNode;
    }
    list2.count++;
    temp = temp->next;

}
cout << "\n\t----------LIST 1----------\n";
list1.display();
```

```
        cout << "\n\t-----------LIST 2-----------\n";
        list2.display();
    }
};

int main() {
    Emp_linked_list ell;

    while(true) {
                    cout<<"\n\t----MENU----\n";
            cout<<"\n\t1. Insert List";
            cout<<"\n\t2. Split in 2 lists";
            cout<<"\n\t3. Display";
            cout<<"\n\t4. Exit";
            int choice;
            cout<<"\n\tEnter choice: ";
            cin>>choice;
            switch (choice)
            {
                    case 1:
                            ell.insert();
                            break;
                    case 2:
                            ell.split();
                            break;
                    case 3:
```

```
                    ell.display();
                    break;
                case 4:
                    return 0;
                    break;
                default:
                    cout<<"\n\tInvalid Input\n";
                }
        }

    return 0;
}
```

Output:





Conclusion

This lab manual provided hands-on experience with singly linked lists. By implementing various operations, we reinforced our understanding of linked list manipulation, memory management, and the application of pointers in dynamic data structures.