**Data Structures Lab**

***Session 04***

**Course:** Data Structures (CS2001) **Semester:** Fall 2022

**Instructor:** Mafaza Mohi **T.A:** N/A

**Note:**

● Maintain discipline during the lab.

● Listen and follow the instructions as they are given.

● Just raise your hand if you have any problem.

● Completing all tasks of each lab is compulsory.

● Get your lab checked at the end of the session.

| **Doubly Link List** |
| --- |

class **Node** {

public:

int key;

int data;

Node \* next;

Node \* previous;

Node() {

key = 0;

data = 0;

next = NULL;

previous = NULL;

}

Node(int k, int d) {

key = k;

data = d;

}

};

class **DoublyLinkedList** {

public:

Node \* head;

DoublyLinkedList() {

head = NULL;

}

DoublyLinkedList(Node \* n) {

head = n;

}

appendNode();

prependNode();

insertNodeAfter();

deleteNodeByKey();

updateNodeByKey();

**};**

**Task-1:**

Create a doubly link list and perform the mentioned tasks.

i. Insert a new node at the end of the list.

ii. Insert a new node at the beginning of list.

iii. Insert a new node at given position.

iv. Delete any node.

v. Print the complete doubly link list.

**Task-2:**

Create two doubly link lists, say L and M . List L should be containing all even elements from 2 to 10 and list M should contain all odd elements from 1 to 9. Create a new list N by concatenating list L and M.

**Task-3:**

Using the above created list N, sort the contents of list N is descending order.

| **Circular Link List** |
| --- |

class **Node** {

public:

int key;

int data;

Node \* next;

Node() {

key = 0;

data = 0;

next = NULL;

}

Node(int k, int d) {

key = k;

data = d;

}

**};**

class **CircularLinkedList** {

public:

Node \* head;

CircularLinkedList() {

head = NULL;

}

appendNode();

prependNode();

insertNodeAfter();

deleteNodeByKey();

updateNodeByKey();

print();

};

**Task-4:**

Create a circular link list and perform the mentioned tasks.

i. Insert a new node at the end of the list.

ii. Insert a new node at the beginning of list.

iii. Insert a new node at given position.

iv. Delete any node.

v. Print the complete circular link list.

**Task-5: Break the above-created circular linked list into two halves.**

| **Circular Double Link List** |
| --- |

class **node**

{

public:

int info;

node \*next;

node \*prev;

};

class **double\_clist**

{

public:

node \*create\_node(int);

insert\_begin();

insert\_last();

insert\_pos();

delete\_pos();

update();

display();

node \*start, \*last;

double\_clist()

{

start = NULL;

last = NULL;

}

};

**Task-6:**

Create a circular Double link list and perform the mentioned tasks.

i. Insert two new nodes at the end of the list with the same data.

ii. Insert two new nodes at the beginning of the list with the same data..

iii. Insert a new node at a given position.

iv. Delete any node.

v. Print the complete circular double link list.

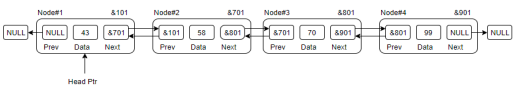
**Task-7:**

Remove duplicates from the above created Doubly Circular Linked list.

**Reference:**

**1. Doubly Link List**

Doubly Linked List is a variation of the Linked list in which navigation is possible in both ways, either forward or backward easily as compared to Single Linked List.

● **Link** − each link of a linked list can store data called an element.

● **Next** − each link of a linked list contains a link to the next link called Next.

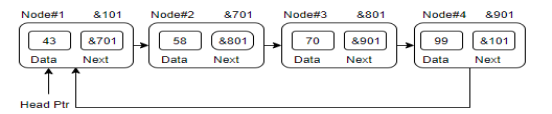
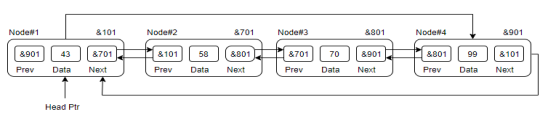
● **Prev** − each link of a linked list contains a link to the previous link called Prev.

2. **Circular Link List**

In a doubly linked list, the next pointer of the last node points to the first node, and the previous pointer of the first node points to the last node making the circular in both directions.

● The last link's next points to the first link of the list in both cases of singly as well as a doubly-linked list.

● The first link's previous points to the last of the list in case of a doubly linked list.



**Basic Operations**

● Traverse ● Append ● Prepend

● Insert ● Delete ● Count ● Display

| **Lab4: Doubly Link List & Circular Link List** | | |
| --- | --- | --- |
| **Std Name: Std\_ID:** | | |
|  | | |
| **Lab1-Tasks** | **Completed** | **Checked** |
| Task #1 |  |  |
| Task #2 |  |  |
| Task #3 |  |  |
| Task# 4 |  |  |
| Task# 5 |  |  |
| Task# 6 |  |  |
| Task# 7 |  |  |