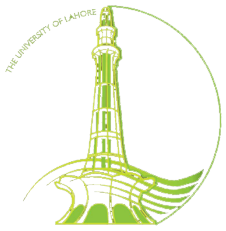

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Data Structure and Algorithms

Lab Report

Name:	Waqar Nawaz Khan
Registration #:	CSU-F16-115
Lab Report #:	06
Dated:	21-04-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 06
Introduction to Doubly Linked list

Objective
The objectives of this lab session are to understand the basics of doubly linked list.

Software Tool
1. Dev C++

1 Theory

A Doubly Linked List (DLL) contains an extra pointer, typically called previous pointer, together with next pointer and data which are there in singly linked list..
Advantages over singly linked list:
1) A DLL can be traversed in both forward and backward direction.
2) The delete operation in DLL is more efficient if pointer to the node to be deleted is given.
In singly linked list, to delete a node, pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using previous pointer.

Disadvantages over singly linked list:
1) Every node of DLL Require extra space for an previous pointer. It is possible to implement DLL with single pointer though (See this and this).
2) All operations require an extra pointer previous to be maintained. For example, in insertion, we need to modify previous pointers together with next pointers. For example in following functions for insertions at different positions, we need 1 or 2 extra steps to set previous pointer.

2 Program

```
#include <stdio.h>
```

```
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
    struct Node *prev;
};
void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data  = new_data;
    new_node->next = (*head_ref);
    new_node->prev = NULL;
     if((*head_ref) !=  NULL)
      (*head_ref)->prev = new_node ;
    (*head_ref)    = new_node;
}
void insertAfter(struct Node* prev_node, int new_data)
{
    if (prev_node == NULL)
    {
        printf("the given previous node cannot be NULL");
        return;
    }
    struct Node* new_node =(struct Node*) malloc(sizeof(struct Node));
    new_node->data  = new_data;
    new_node->next = prev_node->next;
    prev_node->next = new_node;
    new_node->prev = prev_node;
    if (new_node->next != NULL)
      new_node->next->prev = new_node;
}
void append(struct Node** head_ref, int new_data)
{
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    struct Node *last = *head_ref;
    new_node->data  = new_data;
    new_node->next = NULL;
    if (*head_ref == NULL)
    {
```

```
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }
    while (last->next != NULL)
        last = last->next;
    last->next = new_node;
    new_node->prev = last;

    return;
}

void printList(struct Node *node)
{
    struct Node *last;
    printf("\nTraversal in forward direction \n");
    while (node != NULL)
    {
        printf(" %d ", node->data);
        last = node;
        node = node->next;
    }

    printf("\nTraversal in reverse direction \n");
    while (last != NULL)
    {
        printf(" %d ", last->data);
        last = last->prev;
    }
}

int main()
{

    struct Node* head = NULL;
    append(&head, 6);
    push(&head, 7);
    push(&head, 1);
    append(&head, 4);
    insertAfter(head->next, 8);
```

```
    printf("Created DLL is: ");
    printList(head);
    getchar();
    return 0;
}
```