**LAB REPORT ON**

**Data Structures**



**Lab No. 4**

**Date 2020/10/02**

**Topic: Singly Linked List**

**Submitted By**  **Submitted To**

Tapendra Pandey

BCT-D Department of Electronics

075BCT093 and Computer Engineering

**INTRODUCTION**

**Singly Linked List**

Singly linked lists contain nodes, which have a data part as well as an address part i.e. next, which points to the next node in the sequence of nodes. The operations we can perform on singly linked lists are insertion, deletion and traversal.

Linked lists has flexible size i.e. they can grow or shrink as needed to store data during program runtime. Memory can be optimized by doing this and operations can be faster.

**Dynamic implementation of Stack**

Considering array implementation of stack, it works only for fixed number of data. Here the size of data must be specified at the beginning of the implementation itself. It becomes inefficient, when we don't know the size of data, which we are going to use. So a stack data structure can be implemented by using linked list data structure. Using linked lists works for variable size of data.

**Dynamic implementation of Queue**

Considering array implementation of queue, it works only for fixed number of data. Here the size of data must be specified at the beginning of the implementation itself. It becomes inefficient, when we don’t know the size of data, which we are going to use. The queue, which is implemented using linked list, can work for unlimited number of values. The Queue implemented using linked list can organize as many data values as we want. In linked list implementation of a queue, the last inserted node is always pointed by 'rear' and the first node is always pointed by 'front'.

**Application of Singly Linked List**

Different operations, such as addition, subtraction, division and multiplication of polynomials can be performed using linked list.

In the polynomial linked list, the coefficients and exponents of the polynomial are defined as the data node of the list.

For addition operation, two polynomials that are stored as a linked list. We need to add the coefficients of variables with the same power. In a linked list node contains three members, coefficient, power of coefficient and a link to the next node.

**ALGORITHMS**

**1. Stack Using Linked List**

**Push Operation:**

1. Input the DATA to be pushed

2. Create a New Node

3. NewNode → DATA = DATA

4. NewNode → Next = TOP

5. TOP = NewNode

6. Exit

**Pop Operation:**

1. if (TOP =NULL)

Print “The stack is empty”

End if

2. Else

TEMP = TOP

Display “The popped element TOP → DATA”

TOP = TOP → Next

TEMP → Next = NULL

Free the TEMP node

3. Exit

**2. Queue Using Linked List**

**Push Operation:**

1. Input the DATA element to be pushed

2. Create a New Node

3. NewNode → DATA = DATA

4. NewNode → Next = NULL

5. If (REAR not equal to NULL)

REAR → next = NewNode;

End if

6. REAR =NewNode;

7. Exit

**Pop Operation:**

1. If (FRONT is equal to NULL)

Print “Empty Queue”

2. Else

Print “The popped element is FRONT → DATA”

If(FRONT is not equal to REAR)

FRONT = FRONT → Next

End if

Else

FRONT = NULL;

3. Exit

**3. Addition of two polynomials**

1. Start

2. Read the number of terms in the first polynomial, n

3. Read the coefficient and exponents of the first polynomial, coeff and expo.

4. Read the number of terms in the second polynomial, n.

5. Read the coefficient and exponents of the second polynomial, coeff and expo.

6. Set the temporary pointers poly1 and poly2 to traverse the two polynomials respectively.

7. Compare the exponents of the two polynomials starting from the first nodes.

If exponent of poly1 > exponent of poly2

temp→coeff =poly1→coeff;

temp→expo =poly1→expo;

poly1= poly1→link;

Else If exponent of poly2 > exponent of poly1

temp→coeff=poly2→coeff;

temp→expo=poly2→expo;

poly2=poly2→link;

Else if exponent of poly1 = exponent of poly2

temp→coeff=poly1→coeff+ poly2→coeff;

temp→expo=poly1→expo;

poly1=poly1→link;

poly2=poly2→link;

8. Append the remaining nodes of either of the polynomials to the resultant linked list.

9. Display the result

10. End

**DISCUSSION AND CONCLUSION**

In this lab, we learnt about the singly linked list, its operations and application. We did insertion and deletion operations of linked list in stack and queue for their dynamic implementation. Also two polynomials were added as the application of linked list. The dynamic implementations helped us to realize the flexibility of data size in stack and queue.

Hence, in the lab, we understood and implemented algorithms of stack as linked list, queue as linked list and addition of polynomials.