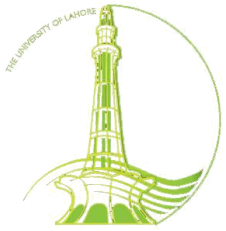

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

DATA STRUCTURE AND ALOGRITHUM

Lab Report

Name: Jassim Bashir
Registration #: SEU-f16-101

Lab Report #: 06
Dated: 5-21-2018
Submitted To: Mr. Usman Ahmed

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

Experiment # 1
DOUBLE LINK LIST

Objective
To understand the meaninig and implementation of double link list.

Software Tool
1.
DEV C++

1 Theory

Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly linked list. Link Each link of a linked list can store a 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. LinkedList A Linked List contains the connection link to the rst link called First and to the last link called Last. Doubly Linked List Representation

As per the above illustration, following are the important points to be considered. Doubly Linked List contains a link element called rst and last. Each link carries a data eld(s) and two link elds called next and prev. Each link is linked with its next link using its next link. Each link is linked with its previous link using its previous link. The last link carries a link as null to mark the end of the list. Basic Operations Following are the basic operations supported by a list. 1. CREATE NEW NODE 2. ADD AT BEGINNING 3. ADD AFTER POSITIO 4. DELETE 5. DISPLAY 6. COUNT 7. REVERSE 8. QUIT

Figure 1: Time Independent Feature Set

2 Task

2.1 Procedure: Task 1

The minimum number of moves required to solve a Tower of Hanoi puzzle is 2n - 1, where n is the number of disks.

```
#i n c l u d e <i o s t r e a m >
#i n c l u d e <c s t d i o >
#i n c l u d e <c s t d l i b >
/
  Node  D e c l a r a t i o n
  /
u s i n g  namespace  s t d ;
s t r u c t  node
f
      int  i n f o ;
      s t r u c t n o d e  next ;
      s t r u c t n o d e  prev ;
g  s t a r t ;
/
  C l a s s  D e c l a r a t i o n
```

```
 /
class double_llist
{
    public:
        void create_list(int value);
        void add_ begin(int value);
        void add_ after(int value , int position);
        void delete_ element(int value);
        void search_ element(int value);
        void display_ dlist();
        void count();
        void reverse();
        double_ llist()
        {
            start = NULL;
        }
};
/
  Main :      Co n at in s Menu
 /
int main ()
{
    int choice, element, position;
    double llistdl;
    while (1)
    {
        cout<<endl<<""<<endl;
        cout<<endl<<" Operations on Doubly linked list "<<endl;
        cout<<endl<<""<<endl;
        cout<<" 1    . Create Node"<<endl;
        cout<<" 2  . Add at begining "<<endl;
        cout<<" 3  . Add after position "<<endl;
        cout<<" 4  . Delete "<<endl;
        cout<<" 5 . Display "<<endl;
        cout<<" 6 . Count"<<endl;
        cout<<" 7 . Reverse "<<endl;
        cout<<" 8 . Quit "<<endl;
        cout<<" Enter your choice : ";
        cin>>choice;
```

```
switch ( choice )
f
case 1:
     cout<<" Enter the element: ";
     cin >>e l e m e n t;
     dl.create_list(element);
     cout<<e n d l;
     break ;
case 2:
     cout<<" Enter the element: ";
     cin >>e l e m e n t;
     dl.add_begin(element);
     cout<<e n d l;
     break ;
case 3:
     cout<<" Enter the element: ";
     cin >>e l e m e n t;
     cout<<" I n s e r t Element after postion: ";
     cin >>p o s i t i o n;
     dl.add_after(element, p o s i t i o n);
     cout<<e n d l;
     break ;
case 4:
     if (start == NULL)
     f
          cout<<" List empty, nothing to delete"<<endl;
          break ;
     g
     cout<<" Enter the element for deletion: ";
     cin >>e l e m e n t;
     dl.delete_element(element);
     cout<<e n d l;
     break ;
case 5:
     dl.display_dlist();
     cout<<e n d l;
     break ;
case 6:
     dl.count();
     break ;
```

```
            case 7:
                if (start == NULL)
                f
                    cout<<" List empty, nothing to reverse"<<endl;
                    break;
                g
                dl.reverse();
                cout<<endl;
                break;
            case 8:
                exit(1);
            default:
                cout<<"Wrong choice"<<endl;
            g
        g
        return 0;
g

/
 Create Double Link List
 /
void double_llist::create_list(int value)
f
        struct node *s, *temp;
        temp = new (struct node);
        temp->info = value;
        temp->next = NULL;
        if (start == NULL)
        f
            temp->prev = NULL;
            start = temp;
        g
        else
        f
            s = start;
            while (s->next != NULL)
                s = s->next;
            s->next = temp;
            temp->prev = s;
        g
```

```
g
/
 Insertion at the beginning
 /
void double _llist::add _begin(int value)
f
      if (start == NULL)
      f
            cout<<" First Create the list."<<endl;
            return ;
      g
      struct node temp ;
      temp = new ( struct node ) ;
      temp >prev = NULL;
      temp >info = value;
      temp >next = start;
      start >prev = temp ;
      start = temp ;
      cout<<" Element Inserted "<<endl;
g

/
 Insertion of element at a particular position
 /
void double _ llist::add_after(int value , int pos)
f
      if (start == NULL)
      f     cout<<" First Create the list."<<endl;
            return ;
      g
      struct node tmp , q ;
      int i ;
      q = start;
      for (i = 0 ; i < pos1 ; i ++)
      f
            q = q >next ;
            if (q == NULL)
            f
```

```
                cout<<" There are less than ";
                cout<<pos<<" elements. "<<endl;
                return ;
            }
        }
        tmp = new (struct node);
        tmp->info = value;
        if (q->next == NULL)
        {
            q->next = tmp;
            tmp->next = NULL;
            tmp->prev = q;
        }
        else
        {
            tmp->next = q->next;
            tmp->next->prev = tmp;
            q->next = tmp;
            tmp->prev = q;
        }
        cout<<" Element Inserted "<<endl;
    }

/
  Deletion of element from the list
 /
void double_llist::delete_element(int value)
{
    struct node *tmp, *q;
    /* first element deletion */
    if(start->info == value)
    {
        tmp = start;
        start = start->next;
        start->prev = NULL;
        cout<<" Element Deleted "<<endl;
        free(tmp);
        return;
    }
    q = start;
```

```
        while ( q >next >next != NULL)
        f
                / Element d e l e t e d  in between /
                i f  ( q >next >i n f o == v a l u e )
                f
                        tmp = q >next ;
                        q >next = tmp >next ;
                        tmp >next >prev = q ;
                        cout<<" Element D␣e l e t e d "<<e n d l ;
                        f r e e ( tmp ) ;
                        return ;
                g
                q  = q >next ;
        g
         / l a s t  e l e m e n t       d e l e t e d /
        i f  ( q >next >i n f o == v a l u e )
        f
                tmp = q >next ;
                f r e e ( tmp ) ;
                q >next = NULL;
                cout<<" Element D␣e l e t e d "<<e n d l ;
                return ;
        g
        cout<<" Element  "≤<value<<"  not␣found "<<e n d l ;

g

/
 D i s p l a y  e l e m e n t s  o f   Doubly  Link     L i s t
 /
v o i d  d o u b l e     _ l l i s t : : d i s p l a y     _ d l i s t ( )
f
        s t r u c t  node   q ;
        i f   ( s t a r t == NULL)

        f       cout<<" L i s t ␣empty , n o t h i n g ␣t o ␣ d i s p l a y "<<e n d l ;
                return ;
        g
        q  = s t a r t ;
        cout<<"The  ␣Doubly ␣Link ␣L i s t ␣i s ␣ : "<<e n d l ;
        while ( q != NULL)
```

```
        f
            cout<<q >i n f o <<" <␣> " ; ␣
            q  = q >next ;
        g
        cout<<"NULL"<<e n d l ;
    g

/
     Number  o f     e l e m e n t s  in Doubly  Link  L i s t
     /
    v o i d d o u b l e␣ l l i s t : : count ( )
    f
        s t r u c t  node   q = s t a r t ;
        int  c n t = 0 ;
        while ( q != NULL)
        f
            q  = q >next ; c
            n t++;
        g
        cout<<"Number  o␣f  ␣e l e m e n t␣s  a r e␣  "<<cnt<<e n d l ;
    g

/
     R e v e r s e  Doubly  Link  L i s t
     /
    v o i d d o u b l e ⊥l i s t : : r e v e r s e ( )
    f
        s t r u c t  node   p1 ,    p2 ;
        p1 = s t a r t ;
        p2 = p1 >next ;
        p1 >next = NULL;
        p1 >prev = p2 ;
        while ( p2  != NULL)
        f
            p2 >prev = p2 >next ;
            p2 >next = p1 ;
            p1 = p2 ;
            p2 = p2 >prev ;
        g
        s t a r t = p1 ;
```

```
        cout<<" L i s t ␣Reversed "<<e n d l ;
g
```

3 Conclusion

in this lab we perform the basics function of double link list insertion deletion insertion at any n postion display reverse etc