COMPUTER SOFTWARE LAB

Name: SAMBHAV R JAIN
Roll No.: 107108103
Date: __ /__ / 2010
Experiment No.: 7

PROBLEM STATEMENT:

Write a program using pointers to create a double linked list with each node containing a data field and two pointer fields. Include member functions to:

1. Add (and/or) delete a node
    a. to the left-most
    b. in the middle- (i.e.) add to the right, delete to the left of given node (info)
    c. to the right-most
2. Locate the predecessor and successor of the given node (info)

PROGRAM CODE:

```
#include<iostream>
using namespace std;
struct node
{
    int info;
    node* front;
    node* rear;
};
class dlinklist     //Double Linked List (Two-Way)
{
    private:
        node* first;
        node* last;
    public:
        node* i;
        dlinklist()
        {
            first=NULL;
            last=NULL;
        }
        void insf(int a)        //Insert as first element
        {
            node* n;
            n=new node;
            n->info=a;
            if(first==NULL)
            {
                n->front=NULL;
                n->rear=NULL;
                first=n;
                last=n;
            }
            else
            {
                n->rear=NULL;
                first->rear=n;
                n->front=first;
                first=n;
            }
        }
        void delf()                //Delete first element
        {
            if(first==NULL)
            cout<<"\nUnderflow!!\n";
            else if(first==last)
            {
                node* n;
                n=first;
                first=NULL;
                last=NULL;
                delete n;
            }
            else
            {
                node* n;
                n=first;
                first=first->front;
                first->rear=NULL;
                delete n;
            }
        }
        void insl(int a)           //Insert as last element
        {
            node* n;
            n=new node;
            n->info=a;
            if(last==NULL)
            {
                n->front=NULL;
                n->rear=NULL;
                first=n;
                last=n;
            }
            else
            {
                n->front=NULL;
                last->front=n;
                n->rear=last;
                last=n;
            }
        }
        void dell()                 //Delete last element
        {
            if(first==NULL)
            cout<<"\nUnderflow!!\n";
            else if(first==last)
            {
              node* n;
              n=last;
              first=NULL;
              last=NULL;
              delete n;
            }
            else
            {
               node* n;
               n=last;
               last=last->rear;
               last->front=NULL;
               delete n;
            }
        }
        void insbet(int a,node* j)   //Insert in between
        {
            node* n;
            n=new node;
            n->info=a;
            if(j->front==NULL)
            {
                n->front=NULL;
                n->rear=j;
                j->front=n;
                last=n;
            }
            else
            {
                n->front=j->front;
                n->rear=j;
                (n->front)->rear=n;
                j->front=n;
            }
        }
        void delbet(node* j)        //Delete in between
        {
            if(first==j)
            cout<<"\nUnderflow!!\n";
            else if(first==j->rear)
            {
                node* n;
                n=j->rear;
                first=j;
                j->rear=NULL;
                delete n;
            }
            else
            {
                node* n;
                n=j->rear;
                (n->rear)->front=j;
                j->rear=n->rear;
                delete n;
            }
        }
        node* loc(int val)
        {
            for(i=first ; (i!=NULL)&&(i->info!=val) ; i=i->front);
            return i;
        }
        void disp()
        {
            cout<<"\nLinked List: ";
            for(node* i=first ; i!=NULL ; i=i->front)
            cout<<i->info<<" ";
        }
};

int main()
{
    int ch,no;
    dlinklist l1;
    node* j;
    z:
    cout<<"\n\nChoose:\n1. Insert as first element\n2. Delete the first element\n3. Insert
as last element\n4. Delete the last element\n5. Insert in between\n6. Delete from in
between\n7. Locate predecessor and successor\n";
    cin>>ch;
```

```
 switch(ch)
 {
      case 1:
          cout<<"Enter the number to be inserted: ";
          cin>>no;
          l1.insf(no);
          l1.disp();
          goto z;
      case 2:
          l1.delf();
          l1.disp();
          goto z;
      case 3:
          cout<<"Enter the number to be inserted: ";
          cin>>no;
          l1.insl(no);
          l1.disp();
          goto z;
      case 4:
          l1.dell();
          l1.disp();
          goto z;
      case 5:
          int val;
          cout<<"Enter the number to the right of which you want to insert a new
number: ";
          cin>>val;
          j=l1.loc(val);
          if(j==NULL)
          {
                cout<<"\nElement doesnt exist in the link list!!";
                goto z;
          }
          cout<<"Enter the number to be inserted: ";
          cin>>no;
          l1.insbet(no,j);
          l1.disp();
          goto z;
      case 6:
          cout<<"Enter the element whose left number is to be deleted: ";
          cin>>val;
          j=l1.loc(val);
          if(j==NULL)
          {
                cout<<"\nElement doesnt exist in the link list!!";
                goto z;
          }
          l1.delbet(j);
          l1.disp();
          goto z;
      case 7:
          cout<<"Enter the element whose predecessor and successor are to be found: ";
          cin>>val;
          j=l1.loc(val);
          if(j==NULL)
          {
                cout<<"\nElement doesnt exist in the link list!!";
                goto z;
          }
          else if(j->front==NULL && j->rear==NULL)
          cout<<"\nNo predecessor or successor!!";
          else if(j->front==NULL)
          cout<<"\nPredecessor: "<<(j->rear)->info;
          else if(j->rear==NULL)
          cout<<"\nSuccessor: "<<(j->front)->info;
          else
          {
            cout<<"\nPredecessor: "<<(j->rear)->info;
            cout<<"\nSuccessor: "<<(j->front)->info;
          }
         goto z;
  }
  return 0;
}
```

OUTPUT:

```
Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
2
Underflow!!
Linked List:

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
1
Enter the number to be inserted: 1
Linked List: 1

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
3
Enter the number to be inserted: 4
Linked List: 1 4

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
```

```
1
Enter the number to be inserted: 2
Linked List: 2 1 4

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
4
Linked List: 2 1

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
5
Enter the number to the right of which you want to insert a new number: 1
Enter the number to be inserted: 12
Linked List: 2 1 12

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
5
Enter the number to the right of which you want to insert a new number: 1
Enter the number to be inserted: 11
Linked List: 2 1 11 12

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
```

```
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
6
Enter the element whose left number is to be deleted: 11
Linked List: 2 11 12

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
6
Enter the element whose left number is to be deleted: 12
Linked List: 2 12

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
5
Enter the number to the right of which you want to insert a new number: 13
Element doesnt exist in the link list!!

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
7
Enter the element whose predecessor and successor are to be found: 12
Predecessor: 2
```

```
Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
1
Enter the number to be inserted: 5
Linked List: 5 2 12

Choose:
1. Insert as first element
2. Delete the first element
3. Insert as last element
4. Delete the last element
5. Insert in between
6. Delete from in between
7. Locate predecessor and successor
7
Enter the element whose predecessor and successor are to be found: 2
Predecessor: 5
Successor: 12
```

RESULT:

A double linked list is created using pointers and functions are written to add and delete nodes at different positions of the list. The predecessor and successor of the given node are also printed.