Experiment #6

Ordered Singly Linked Lists

Student's Name:	
Semester:	Date:

Assessment:

Assessment Point	**Weight**	**Grade**
Methodology and correctness of results		
Discussion of results		
Participation		
Assessment Points' Grade:		

Comments:

Experiment #6:

Ordered linked lists in C++ Programming Language

Objectives:

1. To introduce the students with the concept of ordered singly linked lists
2. To implement ordered singly linked lists
3. To implement different operations on ordered linked lists

Discussion:

Ordered singly linked list has the elements sorted in ascending or descending way. This kind of link lists can be created using two ways:

1. Post creation: where the unordered link list is created, then a sorting function can be used to sort the elements of this link list.

2. Upon creation: where the link list is created in an ordered fashion.

Ordered Singly Linked list implementation

```
// ordered singly link list implementation using templates
// programmed by Dr.Aryaf Aladwan

#include <iostream.h>
template <class T>
class orderedlinklist
{
    private:
       template <class T>
       struct node
       {
         T data;
         node <T> *link;
                  } ;
                  node <T> *head;
```

```
  public:
        orderedlinklist();
        void insert( T num );
        T del( T num );
        void display();
        T count();
        ~orderedlinklist();
};

template <class T>
 orderedlinklist<T>::orderedlinklist()
{
    head = NULL;
}

template <class T>
void orderedlinklist<T>::insert(T num)
{
    node <T>*q,*t,*n;
    n= new node <T>;
          n->data=num;

  if( head == NULL )    // insert into empty list
  {
     head=n;
     head->link = NULL;

  }
  else if(num<head->data)
  {

           q=head;
           while(q->link!=NULL)
```

```
                q=q->link;
        n->link=head;
        head=n;
        q->link=NULL;
  }
  else
  {
        q=head;
        t=head->link;
        while(t!=NULL && num>t->data)
        {
                q=t;
                t=t->link;
        }
        q->link=n;
        n->link=t;
  }
}

template <class T>
T orderedlinklist<T>::del( T num )
{
  node <T>*q,*r;
  q = head;
  if( q->data == num )   // delete from the beginning
  {
    head = q->link;
    delete q;
    return 0;
  }
  r = q;
```

```
while( q!=NULL )
  {
      if( q->data == num )
    {
        r->link = q->link;
       delete q;
       return 0;
    }

    r = q;
    q = q->link;
  }
  cout<<"\nElement "<<num<<" not Found.";
}

template <class T>
void orderedlinklist<T>::display()
{
    node <T>*q;
  cout<<endl;

  for( q = head ; q != NULL ; q = q->link )
      cout<<endl<<q->data;
}

template <class T>
T orderedlinklist<T>::count()
{
    node <T>*q;
  int c=0;
  for( q=head ; q != NULL ; q = q->link )
      c++;
```

```
   return c;
}

template <class T>
orderedlinklist<T>::~orderedlinklist()
{
     node <T>*q;
   if( head == NULL )
        return;
   while( head != NULL )
   {
        q = head->link;
      delete head;
      head = q;
   }
}

int main()
{
   orderedlinklist <int>ll;
   cout<<"No. of elements = "<<ll.count();
   ll.insert(9);
   ll.insert(7);
   ll.insert(8);
   ll.insert(4);
   ll.insert(8);
   ll.insert(1);
   ll.insert(0);
   ll.insert(10);
   ll.insert(4);
   ll.insert(9);
   ll.display();
   ll.display();
```

```
  cout<<"\nNo. of elements = "<<ll.count();
  ll.del(5);
  ll.del(12);
  ll.del(98);
  cout<<"\nNo. of elements = "<<ll.count();
   ll.display();
  return 0;
}
```

Exercise 1:

Write a c++ program to sort the elements of unordered link list?

Solution to Exercise 1

Output

Exercise 2:

Write a c++ program to print the elements of ordered link list in reverse order?

Solution to Exercise 2

Output