Experiment #5

Unordered 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 #5:

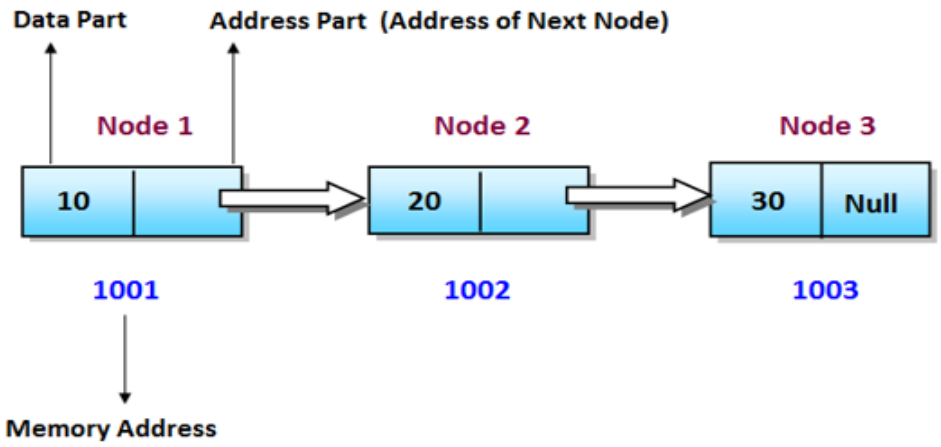
Unordered linked lists in C++ Programming Language

Objectives:

1. To introduce the students with the concept of singly linked lists
2. To implement unordered singly linked lists
3. To implement different operations on linked lists
4. To understand the advantages of linked lists implementation over array implementation.

Discussion:

A linked list is a data structure that can store an indefinite number of items. These items are connected using pointers in a sequential manner. There are two types of linked list; singly-linked list, and doubly-linked list. In a singly-linked list, every element contains some data and a link to the next element. The elements of a linked list are called the nodes. A node has two fields i.e. data and next. The data field contains the data being stored in that specific node. It cannot just be a single variable. There may be many variables presenting the data section of a node. The next field contains the address of the next node. So, this is the place where the link between nodes is established.



Operations on linked list:

1. Isempty(): to check whether the link list is empty or not.
2. Insert(): add elements to the link list as the following manner:
    a. Insert into empty link list
    b. Insert at the end of the link list (append)
    c. Insert in the middle of the link list.

3. Delete(): delete elements from the link list as the following manner:
    a. Delete from the beginning of the link list
    b. Delete from the middle of the link list
    c. Delete from the end of the link list

Unordered Singly Linked list implementation

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

#include <iostream>
using namespace std;
template <class T>
class unorderedlinklist
{
    private:
        template <class T>
        struct node
        {
          T data;
          node <T> *link;
                    } ;
                    node <T> *head;
  public:
        linklist();
        void insert( T num );
        void add_as_first( T num );
        void addafter( T c, T num );
        T del( T num );
        void display();
        T count();
       // ~unorderedlinklist();
};
```

```
template <class T>
 unorderedlinklist<T>::linklist()
{
    head = NULL;
}

template <class T>
void unorderedlinklist<T>::insert(T num)
{
    node <T>*q,*t;

  if( head == NULL )    // insert into empty list
  {
     head = new node<T>;
     head->data = num;
     head->link = NULL;

  }
  else                                                  // append
  {
       q = head;
     while( q->link != NULL )
          q = q->link;

     t = new node<T>;
     t->data = num;
           q->link= t;
     t->link = NULL;

  }
}
```

```
template <class T>
void unorderedlinklist<T>::add_as_first(T num) // insert in the beginning
{
     node <T>*q;

   q = new node<T>;
   q->data = num;
   q->link = head;
   head = q;
}

template <class T>
void unorderedlinklist<T>::addafter( T c, T num)  // insert in the middle
{
   node <T>*q,*t;
   int i;
   for(i=1,q=head;i<c;i++)
   {
        q = q->link;
      if( q == NULL )
      {
           cout<<"\nThere are less than "<<c<<" elements.";
         return;
      }
   }

   t = new node<T>;
   t->data = num;
   t->link = q->link;
   q->link = t;
}

template <class T>
```

```
T unorderedlinklist<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>
unorderedlinklist<T>::~unorderedlinklist()
{
     node <T>*q;
   if( head == NULL )
        return;

   while( head != NULL )
```

```
  {
      q = head->link;
    delete head;
    head = q;
  }
}

int main()
{
  unorderedlinklist <int>ll;
  //cout<<"No. of elements = "<<ll.count();
  ll.insert(12);
  ll.insert(13);
  ll.insert(23);
  ll.insert(43);
  ll.insert(44);
  ll.insert(50);
  ll.add_as_first(2);
  ll.add_as_first(111);
  ll.addafter(2,333);
  ll.addafter(6,666);
  ll.display();
  cout<<"\nNo. of elements = "<<ll.count();
  ll.del(333);
  ll.del(12);
  ll.del(98);
  cout<<"\nNo. of elements = "<<ll.count();
   ll.display();
  return 0;
}
```

Exercise 1:

Write a c++ program to print the contents of the link list?

Solution to Exercise 1

Output

Exercise 2:

Write a c++ program to print the number of elements in the link list?

Solution to Exercise 2

Output

Exercise 3:

Write a c++ program to find the sum of all even numbers stored in the link list?

Solution to Exercise 3

Output