Assignment No.8

Sumit Gulab Bhamare
SEComp A08
Sub - DS2

Aim: To understand & implement set operation using linked list.

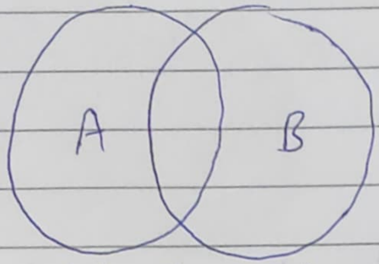
Problem definition: Second year Computer Engineering class, set A of students like Vanilla Ice-cream & set B of students like butterscotch ice-cream. Write c/c++ program to store two sets using linked list. Compute & display i.) Set of students who like either vanilla or butterscotch or both. ii) Set of students who like both vanilla & butterscotch. iii) Set of students who like only vanilla not butterscotch iv) Set of students who like only butterscotch not vanilla v. Numbers of students who like neither vanilla nor butterscotch

Learning objectives:
To understand concept of set operations & linked list.

Learning outcome:
Students will be able to analyse problems to we variants of linked list & solve various real-life problems.

Theory:
Set
Elements are the objects contained in a set. A set may be defined by a common property amongst the objects. For example, the set E of positive even integers is the set $E = \{2, 4, 6, 8, 10\}$

Two data sets A and B.

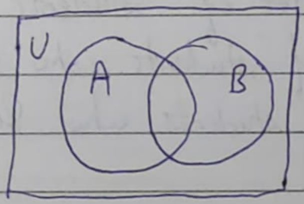


Definition (union): The union of sets A & B, denoted by $A \cup B$, is the set defined as

$$A \cup B = \{x \mid x \in A \forall x \in B\}$$

Eg. $A = \{1,2,3\}$
$B = \{1,2,4,5\}$ then $A \cup B = \{1,2,3,4,5\}$

Note that elements are not repeated in a set.



Definition (Intersection): The intersection of sets A and B, denoted by $A \cap B$, is the set defined as

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

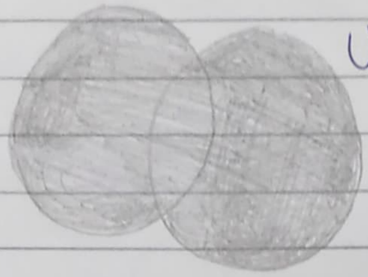
Eg. $A = \{1,2,3\}$ & $B = \{1,2,4,5\}$, then $A \cap B = \{1,2\}$

Definition (Difference): The difference of sets A from B, denoted by $A - B$, is the set defined as

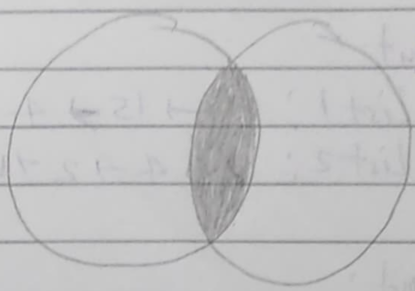
$$A - B = \{x \mid x \in A \wedge x \notin B\}$$

Eg. If $A = \{1,2,3\}$ & $B = \{1,2,4,5\}$, then $A - B = \{3\}$

Note that in general $A-B \neq B-A$

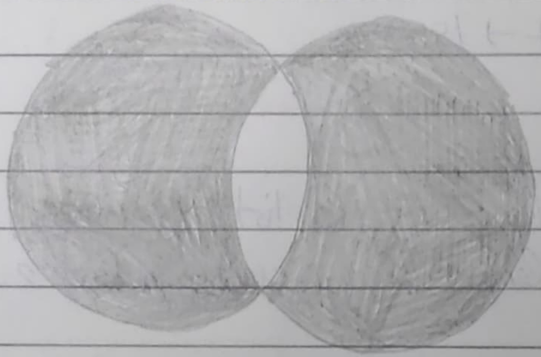


Union

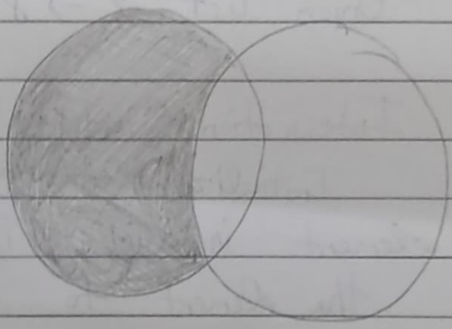


Intersection

Exclusive Or

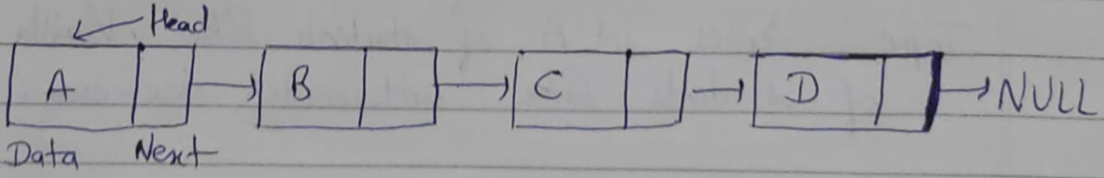


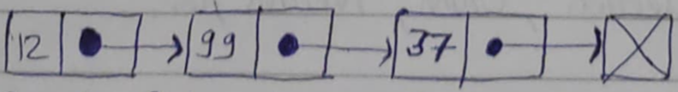
Subtraction



Singly linked list.

Singly linked list contain nodes which have a data field as well as a 'next' field, which points to the next node in line of nodes Operations that can be performed on singly linked lists include insertion, deletion & traversal.





A singly linked list whose nodes contain two fields: an integer value & a link to the next node

Union & Intersection of two linked lists

Input:
List1: 10 → 15 → 4 → 20
List2: 8 → 4 → 2 → 10

Output:
Intersection list: 4 → 10
Union list: 2 -> 8 -> 20 → 4 → 15 -> 10

Intersection (list1, list2):
Initialize result list as NULL. Traverse list1 & look for its each element in list2, if the element is present in list2, then add the element to result.

Union (list1, list2):
Initialize result list as NULL. Traverse list1 & add all of its elements to the result. Traverse list 2. If an element of list 2 is already present in result then do not insert it to result, otherwise insert.

Input: Enter set A of students like Vanilla Ice-cream & set B of students like butterscotch ice-cream.

Output: Intersection, Union, Neither nor.

Algorithm:

· Set of students who like both vanilla & butterscotch .

```
intersection()
{
    Node *cur1 = Butter;
    Node *cur2 = Vanilla;
    int found = 0;
    while (cur1 != NULL)
    {
        while (cur2 != NULL)
        {
            if (cur1-> data == cur2 -> data)
            {
                found=1;
                break;
            }
            else
                cur2=cur2 -> next;
        }
        if (found==1)
        {
            cout<< cur1 -> data<<" ";
            cur1 = cur1 -> next;
            found=0;
        }
        else
            cur1 = cur1 -> next;
        cur2= Vanilla;
    }
}
```

- Set of students who like either vanilla or butterscotch or both

```
void uni()
{
        onlyB();
        intersection();
        onlyV();
}
```

- Number of students who like neither vanilla nor butterscotch

```
void neither()
{
    cout << "\n students who like neither vanilla nor butterscotch\n";
    temp = h1;
     while (temp != NULL)
      { temp3 = head3;
         f=0;
         while (temp3 != NULL)
         {  if (temp->roll == temp3->roll)
              { f=1;   }
              temp3 = temp3 -> next;
         }
       if (f==0)
         {   cout << "\n" << temp -> roll ; }
           temp = temp -> next;
         }
}
```

Software required:- g++/gcc compiler - / 64 bit fedora.

Conclusion: We understand & implement different operations on of linked list.