TUTORIAL NO. – 7

**SET OPERATIONS USING LINKED LIST**

**PROBLEM STATEMENT /DEFINITION:-**

Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream. Write C/C++ program to store two sets using linked list. Compute and display

1. Set of students who like either vanilla or butterscotch or both.
2. Set of students who like both vanilla and butterscotch.
3. Set of students who like only vanilla not butterscotch.
4. Set of students who like only butterscotch not vanilla.
5. Number of students who like neither vanilla nor butterscotch.

**OBJECTIVE:-**

* To understand concept of SETs and its primitive operations.
* To understand use of Linked list for implementing SET operations.

**OUTCOME:-**

* To implement primitive operations on SLL/ DLL.
* To apply SLL/DLL in maintaining ice cream parlour data.

**S/W PACKAGES AND HARDWARE APPARATUS USED:-**

Operating Systems (64-Bit) 64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit) Latest Open source update of Eclipse Programming frame work, TC++.

**REFERENCES:-**

* + C++ by B. Stroustrup
  + Fundamental of Data Structure in C++

**STEPS:-**

Refer to algorithm

**Theory:-**

Sets are containers that store unique elements following a specific order.In a set, the value of an element also identifies it (the value is itself the key, of type T), and each value must be unique. The value of the elements in a set cannot be modified once in the container (the elements are always const), but they can be inserted or removed from the container.

Internally, the elements in a set are always sorted following a specific strict weak ordering criterion indicated by its internal comparison object (of type Compare).set containers are generally slower than unordered\_set containers to access individual elements by their key, but they allow the direct iteration on subsets based on their order.

Sets are typically implemented as binary search trees.

template < class T, // set::key\_type/value\_type

class Compare = less<T>, // set::key\_compare/value\_compare

class Alloc = allocator<T> // set::allocator\_type

> class set;

**Algorithm:-**

Step1: Start

Step2: Create a function to get union of two set using linked list.

Step3: Insert all elements of list 1 to resulting list

Step4: Insert those elements which are not present in result list.

Step5: Return result.

Step6: Create function to get intersection of two sets in linked list

Step7: Traverse list1 and search each element of it in list2.

Step8: If the element is present in list 2, then insert the element to result.

Step9: Return result

Step10: End

Complexity:

Time complexity is O(n).