Page No.
Date: / /

Assignment - 9

Title : SET OPERATIONS.

Problem : To create ADT that implements SET concepts.

- Add new element
- Removes element.
- Returns true if element is present.
- Return size of set.
- Intersection.
- Union.
- Difference
- Subset.

Objective : To implement SET ADT and learn Set operations like intersection, union difference, subset.

Outcome : We will have a ready SET ADT for applications.

H/W & S/W : 64 bit OS, Fedora OS, Eclipse.

Theory :

Abstract datatype that can store unique values without any particular order. It is the complete implementation of a finite set.

Page No. _____
Date: / /

Operations on Set:

- Union (S, T)
  returns S ∪ T
- Intersection (S, T)
  returns S ∩ T
- Difference (S, T)
  returns S - T
- Subset (S, T)
  checks whether S is a subset of T

ALGORITHMS:

1. Insert

```
void insert()
{
    read d
    node * n = new node(d)
    if list is empty then
        head = n
    else
    node * temp = head
    while temp ~~= head~~ is not last element do
        temp = temp -> next
        temp -> next = n
    end if
size ++ }
```

2. Delete

```
void remove()
{
  read d;
  node *temp, *prev;
  if list is empty then
    return;
  else if head->data = d then
    head = head->next;
  else
  if d is found in list then
     prev = head
       temp = head.
  do
   if (temp->data = d) then
     prev -> next = temp->next
        delete temp
        -- size
      break
    end if
   prev = temp
    temp = temp->next
    end dowhile (temp exist)
   end if
}
```

3. Search

```
node * search (int a)
{
  node * temp = head
    if (list is empty) then
       return NULL
    do
      if temp -> data == a then
         return temp .
       temp = temp -> next .
    end do while (temp exist)
    return
}
```

4. Size

```
int retsize()
{
   return size;
}
```

5. Intersection (list b)

```
{
    list c;
  node * temp = b.head;
  while temp exists do
    c. insertnode (temp -> data);
      temp = temp -> next
  end while
  temp = c. head;
  while temp exists do
    if search (temp -> data) == NULL do
       c. removeNode (temp);
```

```
    temp = temp -> next ;
  end while
    return c
}
```

6. Union

```
list union(int b)
{
   list c;
   node * temp = b_head;
   while temp exists do
    c. insertNode (temp -> data)
    temp . temp -> next.
  end while.
  temp = head;
   while temp exists do
   if c.search (temp -> data) == NULL do
      c. insertNode (temp -> data);
   temp = temp -> next;
   end while
   return C;
}
```

7. Difference

```
list diff (list b)
{
    node * temp = head;
    while temp exists do
    c . insert Node (temp -> data)
    temp = temp -> next;
    end while;
```

```
temp = b_head;
while temp exists do
  if (c.search (temp -> data) then
      c. removeNode (temp);
  temp = temp -> Next;
  end while
return c
}
```

8. SUBSET

```
void subset (list b)
{
  count = 0;
  node * temp = b_head; while (temp exists)
if (search (temp -> data)) then
    count++
    temp = temp -> next
  end while
  if (count == b.retsize()) then
     print ("Subset");
  else
     print ("Not Subset");
  end if
}
```

Page No.
Date: / /

TEST CASES

A = {4, 1, 3, -1, 7, 9}

	I/P	O/P	Result	Analysis.
1.	Insert 5	A = {5, 4, 1, 3, -1, 7, 9}	Success	O(n)
2.	Remove 1	A = {5, 4, 3, -1, 7, 9}	Success	O(n)
3.	Contains 7	True	Success	O(n)
4.	Size	5	Success	O(n)
5.	B = {1, 2, 9, 3} Perform Union	C = {1, 2, 3, 5, 4, -1, 7, 9}	Success	O(n)
6.	Intersection	C = {1, 3, 9}	Success	O(n)
7.	Difference	D = {-1, 4, 5, 7, 9}	Success	O(n)
8.	Subset	B is not subset of A	Success	O(n)
9.	B = {-1, 7}			

Conclusion: We have understood and implemented set ADT and performed basic algebra set operations on it.