Assignment no - 9

Title - SET Operations

Problem statement - To create ADT that implements SET concepts ① Add new element
② Remove element
③ Return true if element is present
④ Return size of set
⑤ Insertion
⑥ Union
⑦ Difference
⑧ Subset

Objective - To implement SET ADT and learn set operations like intersection, union, difference, subset

Outcome - We will have a ready SET ADT for application

H/W and S/W requirements - DELL optiplex 3010 MT, keyboard, Monitor, Fedora 20, Eclipse

Theory - SETs
Abstract data type that can store unique values without any particular order. It is the complete implementation of a finite set

* Operations
① Union (s, T)
returns S ∪ T
② Intersection (s, T)
returns S ∩ T

③ Difference (S, T)
returns S-T

④ Subset (S, T)
checks whether S is a subset of T

Algorithm

① Insert

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

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
    endif
    prev = temp
    temp = temp → next
    end dowhile (temp exist)
    endif
    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 double(temp exist)
    return
}
```

④ Size

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

⑤ Insertion (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 exist do
    if search (temp → data == NULL) do
      C.removeNode (temp);
    temp = temp → Next;
  end while
    return c
}
```

⑥ Union

```
list union (list 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
endwhile
return()
}
```

⑦ Difference

```
list diff(list b)
{
    node * temp = head;
    while temp exists do
    {insertNode(temp→data)
     temp = temp→next;
     endwhile
     temp = b.head;
    while temp exists do
    if (c.search(temp→data)) then
       c.removeNode(temp);
    temp = temp→next;
    endwhile
    return c
}
```

⑧ seeSubset

```
void subset (list b)
{ count = 0;
  node *temp = b.head;
  if (search (temp -> data)) then
     count ++;
  temp = temp -> next
  endwhile
   if (count == b.retSize() then
    print "subset"
   else
     print ("Not subset")
  endif
}
```

Test cases

	Description	Expected	Actual	Result
①	A{4,1,3,-1,7,9}	A={4,1,3,-1,7,9}	same	success
②	Insert 2	A={2,4,1,3,-1,7,9}	same	success
③	Remove 1	A={2,4,3,-1,7,9}	same	success
④	Contains 7	True	same	success
⑤	size	5	same	success
⑥	B={1,2,9,3} Union	c={1,2,3,4,-1,7,9}	same	success
⑦	Intersection	C={1,2,3,9}	same	success
⑧	Difference	D={-1,4,7}	same	success
⑨	subset	B is a subset of A	same	success
⑩	B={-1,7} subset			

Conclusion - We have understood implementation of SET ADT & perform basic algebraic set operations on it.