Set

Set

- Empty Set : S = { }
- S1 = { 1, 9, 2, 6 }
- S2 = { 2, 8, 7 }
- An element 7 is a member of S2
- An element 7 is NOT a member of S1

Operation on Set

- S1 = { 1, 9, 2, 6 } S2 = { 2, 8, 7 }
- Add a member 3 to a set S1, 7 to S2
  - S1 = { 1, 9, 2, 6, 3 } S2 = { 2, 8, 7 }
  - But, we **sort the data**, (explain later why)
- Delete a member from a set
- Is a set empty?
- Is an element a member of a set?
- Set Union S1 U S2 = { 1, 9, 2, 6, 8, 7 }
- Set Intersection S1 ^ S2 = { 2 }

Structure of A Linked List

```
typedef struct list_node {
        struct list_node *link;
        DATA_TYPE data;
} NODE;
```

We use “int” as DATA_TYPE

We use LINKED LIST for implementation of SET

Ordered List

- Set union/intersection takes N1xN2 time at worst case if not ordered
- We use ORDERED LIST of ASCENDING order
- S1 = { 1, 2, 6, 9 }
- S2 = { 2, 7, 8 }

Compare 2 Lists (1)

ADD 1

1 -> 2 -> 6 -> 9

2 -> 7 -> 8

Compare 2 Lists (2)

ADD 2

1 -> 2 -> 6 -> 9

2 -> 7 -> 8

Compare 2 Lists (3)

ADD 6

1 -> 2 -> 6 -> 9

2 -> 7 -> 8

Compare 2 Lists (4)

ADD 7

1 -> 2 -> 6 -> 9

2 -> 7 -> 8

Compare 2 Lists (5)

ADD 8

1 -> 2 -> 6 -> 9

2 -> 7 -> 8

Compare 2 Lists (6)

ADD 9

1 -> 2 -> 6 -> 9

2 -> 7 -> 8

Compare 2 Lists (7)

fin

1 -> 2 -> 6 -> 9

2 -> 7 -> 8

What to do

Write 7 functions

- int addElement(NODE **set, int i_data);
- int deleteElement(NODE **set, int *i_data);
- int isMember(NODE *set, int i_data);
- int isEmpty(NODE *set);
- NODE *setUnion(NODE *set1, NODE *set2);
- NODE *setIntersection(NODE *set1, NODE *set2);
- void deleteSet(NODE **set);

A function is given

- printSet(NODE *set);

int addElement(NODE **set, int i_data);

- Add an element (i_data) to a given set.
- All elements in the set should be **sorted in ascending order**.
- Returns 1 if the element is actually added, otherwise, return 0.
- Note that if i_data is already a member of the set, addition is not made, and returns 0.

int deleteElement(NODE **set, int *i_data);

- Delete an element given by *i_data,
- If i_data = 0, i.e., if the data pointer is NULL, delete the 1st element (smallest number) of the set.
- Return 1 if a deletion is made, otherwise, returns 0.

Membership

- int isMember(NODE *set, int i_data);

  Returns 1 if i_data is a member of the set, otherwise, returns 0.

- int isEmpty(NODE *set);

  Returns 1 if the set is empty, otherwise, returns 0.

NODE *setUnion(NODE *set1, NODE *set2);

- Returns S1 U S2 of two sets.
- Returns "(NODE *)0" if set union is empty
- Both set1 and set2 must be the same before and after calling this function.

- Function, "setIntersection" is the same except that it performs set intersection.