```
//Union and Intersection of 2 sets represented using SLL.

#include <stdio.h>
#include <stdlib.h>

typedef struct node *Nodeptr;
struct node{
   int data;
   Nodeptr next;
};

Nodeptr getnode(){
   Nodeptr temp;

   temp = (Nodeptr) malloc(sizeof(struct node));

   return temp;

}
int IsEmpty(Nodeptr first){
   return (first == NULL ?1:0);
}
void Display(Nodeptr first){
   Nodeptr temp;

   if (IsEmpty(first)) { printf("Empty Set\n");return;}
   printf("Contents of Set : \n");

   temp = first;
   while(temp!=NULL){
      printf("%d\n",temp->data);
      temp = temp->next;
   }
}
```

```
void InsertLast(int x, Nodeptr *last){
   Nodeptr temp;

   temp= getnode();
   temp->data = x;
   temp->next = NULL;

   (*last)->next = temp;
   *last = temp;

}
```

```
 int IsMember(int item, Nodeptr first){
    Nodeptr temp;

    if (first == NULL) {printf("List is Empty\n");return 0;}

    temp=first;
    while(temp!= NULL){
       if (temp->data==item)
            return 1;

       temp=temp->next;
    }

   return 0 ;
 }
```

```
Nodeptr Create(){ //Create a singly linked list
   int item;
   Nodeptr first, last,temp;

   first = getnode(); //dummy node to make the insertion easier
   last = first;

   printf("Enter the number (unique) to be inserted (in ascending order): [-99 to end] ");
   scanf("%d",&item);
   while(item!=-99)
   {
      InsertLast(item, &last);
      printf("Enter the number to be inserted: [-99 to end] ");
      scanf("%d",&item);
   }

   temp = first;
   first= first->next;
   free(temp); //delete dummy node

   return first;
}
```

```
Nodeptr Union(Nodeptr a, Nodeptr b){

   Nodeptr first, last;

   first = getnode();
   last = first;

   //copy all elements of set A into new set representing union
   Nodeptr temp= a;
   while (temp!=NULL){
      InsertLast(temp->data, &last);
      temp=temp->next;
   }

  //Insert the elements of set B which are not present in set A into the new set
   temp= b;
   while(temp!= NULL){
      if (!IsMember(temp->data, a))        //search for temp->data in set a
         InsertLast(temp->data, &last);
      temp=temp->next;
   }

   temp = first;
   first = first->next;
   free(temp); //delete dummy node

   return first;

}
```

```
Nodeptr Intersection(Nodeptr a, Nodeptr b){
   Nodeptr first, last;;

   first = getnode();//dummy node
   last = first;

   Nodeptr temp= a;

   // Insert elements of set A which are also present in set B into the new list representing intersection
   while (temp!=NULL){
      if (IsMember(temp->data, b))
         InsertLast(temp->data, &last);
      temp=temp->next;
   }

   temp = first;
   first = first->next;
   free(temp); //delete dummy node

   return first;

}
```

```
int main(){
   Nodeptr a,b,c,d;
   printf("Singly Linked List ....\n");
   printf("Creating first Set : \n");
   a = Create();

   Display(a);
   printf("Creating second Set : \n");

   b=Create();

   Display(b);

   c = Union(a,b);
   printf("Forming Union of 2 lists \n");
   Display(c);

   d = Intersection(a,b);
   printf("Forming Intersection of 2 lists \n");
   Display(d);
   return 0;

}
```