```
//Created By Ritwik Chandra Pandey on 14/02/21
//183215
//Union, Intersection, Concatenation of SLL

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


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

NODE createAndAddNodes(NODE first) {
   NODE temp,q=NULL;
   int x;



   printf("Enter element : ");
   scanf("%d",&x);


   while(x!=-1){
      temp=(NODE)malloc(sizeof(struct node));
      temp->next=NULL;
      temp->data=x;

      if(first==NULL){
         first=temp;
      }else{

         q->next= temp;
      }

      q=temp;
```

```
        printf("Enter element : ");
        scanf("%d",&x);
    }


    return first;

}

NODE concatenate(NODE t1, NODE t2) {
    NODE t3;
    if(t1==NULL){
        return t2;}
    else if(t2==NULL){
        return t1;
    }else{
        t3=t1;
        while(t1->next!=NULL){

            t1=t1->next;
        }
        t1->next=t2;

    }
    return t3;



}

void print(NODE first) {
    NODE q = first;
    if (first == NULL) {
        printf("Single Linked List is empty\n");
```

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

}




NODE unionSLL(NODE l1, NODE l2) {
    NODE l3,t1,t2,q=NULL;
    l3=concatenate(l1,l2);
    for(t1=l3;t1!=NULL && t1->next!=NULL;t1=t1->next){
        for(t2=t1;t2->next!=NULL;){
            if(t1->data==t2->next->data){
                q=t2->next;
                t2->next=q->next;
                free(q);

            }else{
                t2=t2->next;
            }
        }
    }
    return l3;
}
NODE add(NODE l3,int x) {
    NODE temp;
    temp=(NODE)malloc(sizeof(struct node));
    temp->next=NULL;
    temp->data=x;
    if(l3==NULL){
        l3=temp;
```

```
    }else{
        while(l3->next!=NULL){
            l3=l3->next;
        }
        l3->next=temp;
    }
    return l3;
}


NODE intersectionSLL(NODE l1,NODE l2) {
    NODE t1=l1,t2=l2,l3=NULL;
    while(t1!=NULL && t2!=NULL) {
        if(t1->data<t2->data){
            t1=t1->next;
        }else if(t1->data>t2->data){
            t2=t2->next;
        }else{
            l3=add(l3,t1->data);
            t1=t1->next;
            t2=t2->next;
        }
    }
    return l3;
}


NODE sort(NODE first) {
    NODE t1,t2;
    int x;
    for(t1=first;t1->next!=NULL;t1=t1->next){
        for(t2=t1->next;t2!=NULL;t2=t2->next){
            if(t1->data>t2->data){
                x=t1->data;
                t1->data=t2->data;
                t2->data=x;
            }
```

```
        }
    }
    return first;
}
```

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

int main() {
    int select;
    NODE l1, l2, l3;
    l1 = l2 = l3 = NULL;
    printf("Enter list-1 elements :\n");
    l1 = createAndAddNodes(l1);
    printf("Enter list-2 elements :\n");
    l2 = createAndAddNodes(l2);
    if (l1 == NULL || l2 == NULL) {
        printf("Single Linked List is empty\n");
        return 0;
    }
    printf("\nFirst List: ");
    print(l1);
    printf("\nSecond List: ");
    print(l2);
    printf("\nSelect an option to proceed: \n");
    printf("\t\t1.Union\n\t\t2.Intersection\n\t\t3.Concatenation\n");
    scanf("%d",&select);
    switch(select){

        case 1:
    l3 = unionSLL(l1, l2);
    printf("Elements in the union list :\n");
    print(l3);
            break;
```

```
        case 2:
    l1 = sort(l1);
    l2 = sort(l2);
    l3 = intersectionSLL(l1, l2);
    printf("Elements in the intesection list :\n");
    print(l3);
            break;


        case 3:
    l3 = concatenate(l1, l2);

    printf("Elements in concatenated list :\n");
    print(l3);
            break;

    }
    return 0;
}
```