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

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

struct node *create(struct node *start, int z)
{
    struct node *temp, *p;
    int n;
    printf("Enter the number of elements in set %d: ", z);
    scanf("%d", &n);
    start = NULL;
    if (n == 0)
        return start;
    for (int i = 0; i < n; i++)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        printf("Enter the data for node %d: ", i + 1);
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (start == NULL)
            start = temp;
        else
        {
            p = start;
            while (p->next != NULL)
                p = p->next;
            p->next = temp;
        }
    }
    return start;
};

void display(struct node *start)
{
    struct node *p;
    if (start == NULL)
    {
        printf("\nLinked list is empty");
        return;
    }
    p = start;
    printf("\nLinked list is: ");
    while (p != NULL)
    {
        printf("%d ", p->data);
        p = p->next;
```

```
    }
    printf("\n");
};

// count the number of nodes in the linked list
int count_nodes(struct node *start)
{
    struct node *p = start;
    int count = 0;
    while (p != NULL)
    {
        count++;
        p = p->next;
    }
    return count;
}

// union of two sets
struct node *union_set(struct node *start1, struct node *start2)
{
    struct node *start3, *temp, *p;
    int n1, n2, n3;
    n1 = count_nodes(start1);
    n2 = count_nodes(start2);
    n3 = n1 + n2;
    start3 = NULL;
    p = start1;
    while (p != NULL)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->data = p->data;
        temp->next = NULL;
        if (start3 == NULL)
            start3 = temp;
        else
        {
            struct node *q = start3;
            while (q->next != NULL)
                q = q->next;
            q->next = temp;
        }
        p = p->next;
    }
    p = start2;
    while (p != NULL)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->data = p->data;
        temp->next = NULL;
        if (start3 == NULL)
            start3 = temp;
```

```
        else
        {
            struct node *q = start3;
            while (q->next != NULL)
                q = q->next;
            q->next = temp;
        }
        p = p->next;
    }
    return start3;
}

// intersection of two sets
struct node *intersection_set(struct node *start1, struct node *start2)
{
    struct node *start3, *temp, *p, *q, *r;
    int n1, n2, n3;
    n1 = count_nodes(start1);
    n2 = count_nodes(start2);

    start3 = NULL;
    p = start1;
    while (p != NULL)
    {
        q = start2;
        while (q != NULL)
        {
            if (p->data == q->data)
            {
                temp = (struct node *)malloc(sizeof(struct node));
                temp->data = p->data;
                temp->next = NULL;
                if (start3 == NULL)
                    start3 = temp;
                else
                {
                    r = start3;
                    while (r->next != NULL)
                        r = r->next;
                    r->next = temp;
                }
            }
            q = q->next;
        }
        p = p->next;
    }
    return start3;
}

// set difference of two sets
struct node *set_difference(struct node *start1, struct node *start2)
```

```
{
    struct node *start3, *temp, *p, *q, *r;
    int n1, n2, n3;
    n1 = count_nodes(start1);
    n2 = count_nodes(start2);

    start3 = NULL;
    p = start1;
    while (p != NULL)
    {
        q = start2;
        while (q != NULL)
        {
            if (p->data == q->data)
                break;
            q = q->next;
        }
        if (q == NULL)
        {
            temp = (struct node *)malloc(sizeof(struct node));
            temp->data = p->data;
            temp->next = NULL;
            if (start3 == NULL)
                start3 = temp;
            else
            {
                r = start3;
                while (r->next != NULL)
                    r = r->next;
                r->next = temp;
            }
        }
        p = p->next;
    }
    return start3;
}



int main()
{
    struct node *start1 = NULL, *start2 = NULL;
    int choice;
    while (1)
    {
        printf("\n1. Create lists");
        printf("\n2. Display lists");
        printf("\n3. Union of lists");
        printf("\n4. Intersection of lists");
        printf("\n5. Set difference of lists");
```

```
        printf("\n6. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            start1 = create(start1, 1);
            start2 = create(start2, 2);
            break;
        case 2:
            display(start1);
            display(start2);
            break;
        case 3:
            display(union_set(start1, start2));
            break;
        case 4:
            display(intersection_set(start1, start2));
            break;
        case 5:
            display(set_difference(start1, start2));
            break;
        case 6:
            exit(0);
        default:
            printf("\nInvalid choice");
        }
    }
    return 0;
}
```