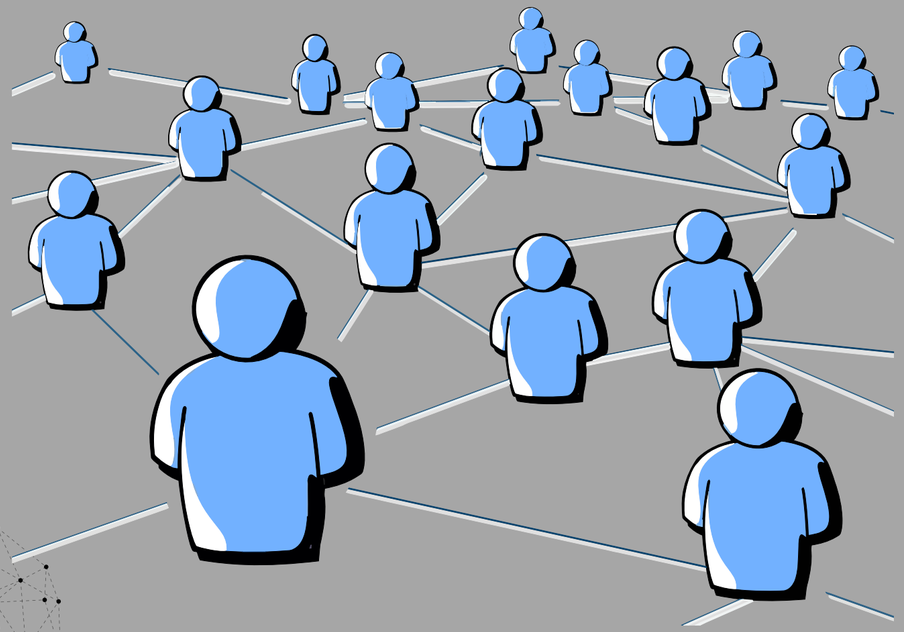
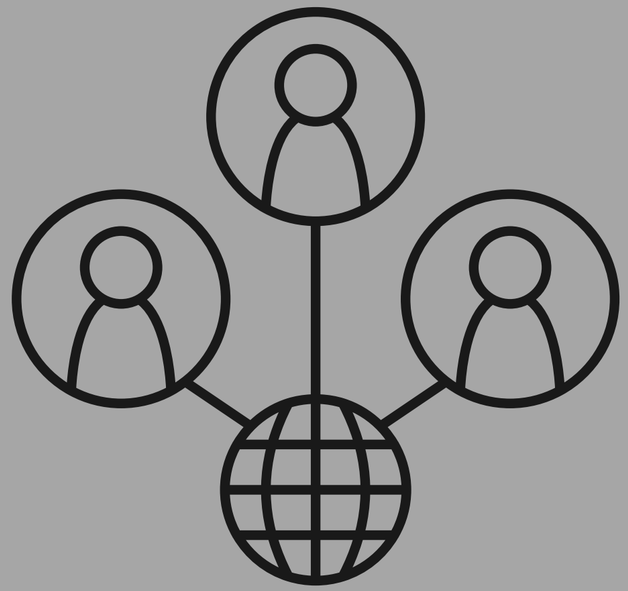
Social Network Functionalities



About our Project:

- I have created a program on Social Network Functionalities using the best suitable data structures and minimum complexity. It has functionalities like:
- Adding users to the network.
- Adding friends to the user's friend list.
- Displaying friend list.
- Finding mutual friends.
- Find a user who has highest number of friends.



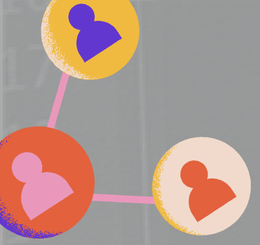
Data Structures Used:

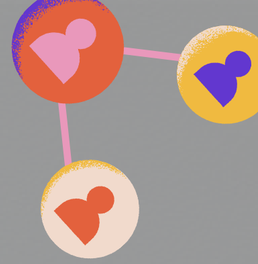
- Graph Implementation using a linked list of linked lists.
- Singly Linked List.

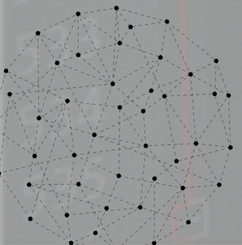
Complexity:

- Quick sort has been used to find mutual friends- O(n^2).
- To find a member with maximum friends in O(n).
- Adding members in network in O(1).
- Adding friends of the members in O(n).
- Displaying friend list in O(n).

code







```c
#include<stdio.h>

struct nodelist{
    int node;
    struct nodelist* next;
    struct adjlist* adj;
}*head=NULL;

struct nodelist* curr;

struct adjlist{
    int frd;
    struct adjlist* link;
};

struct frdct{
    int id;
    int frd;
    struct frdct * next;
}*head_frd=NULL;

void add_member(){
    struct nodelist* newnode;
    newnode = (struct nodelist*)malloc(sizeof(struct nodelist));
    int id;
    printf("Enter the ID of new member : ");
    scanf("%d",&id);
    newnode->node = id;
    if(head==NULL){
        head = newnode;
        curr = newnode;
    }else{
        curr->next = newnode;
        curr = newnode;
    }
    newnode->next=NULL;
    newnode->adj=NULL;

    printf("->Member added successfully\n");
}
```

```c
void add_edge(){
    int m1,m2;
    printf("Enter the ID of member : ");
    scanf("%d",&m1);
    printf("Enter the ID of friend member : ");
    scanf("%d",&m2);
    struct nodelist * temp = head;
    while(temp->node != m1){
        temp=temp->next;
    }
    struct adjlist * adjlistnode;
    adjlistnode = (struct adjlist*)malloc(sizeof(struct adjlist));
    adjlistnode->frd = m2;
    if(temp->adj==NULL){
        temp->adj = adjlistnode;
        adjlistnode->link = NULL;

        struct frdct * new_frdlist;
        new_frdlist = (struct frdct*)malloc(sizeof(struct frdct));
        new_frdlist->id = m1;
        new_frdlist->frd = 1;
        if(head_frd==NULL){
            head_frd = new_frdlist;
        }else{
            struct frdct * tt = head_frd;
            while(tt->next != NULL){
                tt = tt->next;
            }
            tt->next=new_frdlist;
        }
        new_frdlist->next=NULL;

    }else{
        struct adjlist * t = temp->adj;
        while(t->link != NULL){
            t=t->link;
        }
        t->link=adjlistnode;
        adjlistnode->link=NULL;

        struct frdct * tt = head_frd;
        while(tt->id != m1){
            tt = tt->next;
        }
        int f = tt->frd ;
        tt->frd = f+1;
    }
    printf("->Friend added successfully\n");
}
```

```c
struct adjlist* parition(struct adjlist* first, struct adjlist* last)
{
    struct adjlist* pivot = first;
    struct adjlist* front = first;
    int temp = 0;
    while (front != NULL && front != last) {
        if (front->frd < last->frd) {
            pivot = first;

            temp = first->frd;
            first->frd = front->frd;
            front->frd = temp;

            first = first->link;
        }
        front = front->link;
    }
    temp = first->frd;
    first->frd = last->frd;
    last->frd = temp;
    return pivot;
}


void quick_sort(struct adjlist* first, struct adjlist* last)
{
    if (first == last) {
        return;
    }
    struct adjlist* pivot = parition(first, last);

    if (pivot != NULL && pivot->link != NULL) {
        quick_sort(pivot->link, last);
    }

    if (pivot != NULL && first != pivot) {
        quick_sort(first, pivot);
    }
}

struct adjlist* last_node(struct adjlist* head)
{
    struct adjlist* temp = head;
    while (temp != NULL && temp->link != NULL) {
        temp = temp->link;
    }
    return temp;
}
```

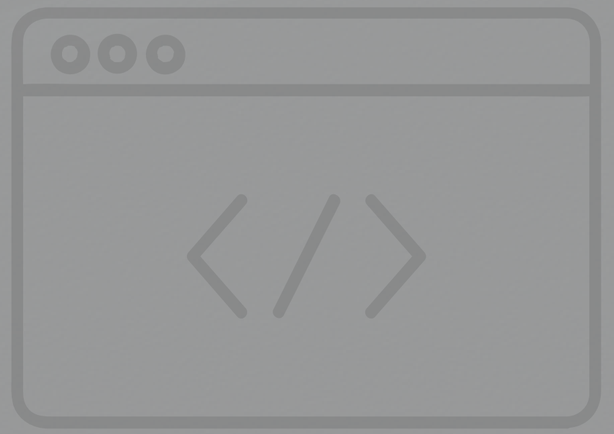
```c
struct adjlist* last_node(struct adjlist* head)
{
    struct adjlist* temp = head;
    while (temp != NULL && temp->link != NULL) {
        temp = temp->link;
    }
    return temp;
}

struct adjlist* fisrt_node(int m){
    struct nodelist* temp = head;
    while(temp->node != m){
        temp = temp->next;
    }
    return temp->adj;
}

struct nodelist* m_node(int m){
    struct nodelist* temp = head;
    while(temp->node != m){
        temp = temp->next;
    }
    return temp;
}

void sortedIntersect(struct adjlist* a, struct adjlist* b)
{
    printf("\n->Mutual friends of the two members are : ");
    while (a != NULL && b != NULL)
    {
        if (a->frd == b->frd)
        {
            printf("%d ",a->frd);
            a = a->link;
            b = b->link;
        }
        else if (a->frd < b->frd){
            a = a->link;
        }
        else{
            b = b->link;
        }
    }
    printf("\n\n");
}
```

```c
void mutual_frd(){
    int m1,m2;
    printf("Enter the ID of member 1 : ");
    scanf("%d",&m1);
    printf("Enter the ID of member 2 : ");
    scanf("%d",&m2);

    struct adjlsit* fn1,*ln1;
    fn1 = fisrt_node(m1);
    ln1 = last_node(fn1);
    quick_sort(fn1,ln1);

    struct adjlsit* fn2,*ln2;
    fn2 = fisrt_node(m2);
    ln2 = last_node(fn2);
    quick_sort(fn2,ln2);

    sortedIntersect(fisrt_node(m1),fisrt_node(m2));
}

void max_conn(){
    struct frdct * tt = head_frd;
    int ct = 0,max_id;
    while (tt->next != NULL)
    {
        if(ct < tt->frd) {ct = tt->frd;max_id=tt->id;}
        tt = tt->next;
    }
    if(ct < tt->frd) {ct = tt->frd;max_id=tt->id;}
    tt = tt->next;
    printf("\n->%d has maximum connections\n\n",max_id);
}
```

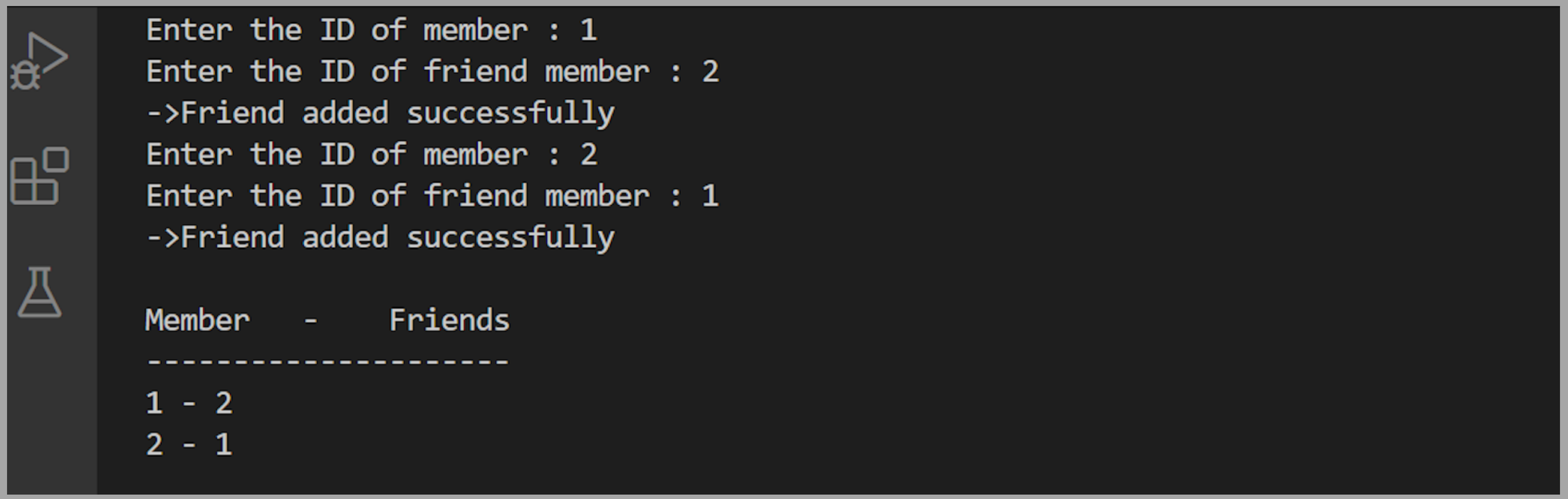
```c
void display(){
    printf("\nMember   -    Friends\n");
    printf("---------------------\n");
    struct nodelist * temp = head;
    while(temp->next !=NULL){
        printf("%d - ",temp->node);
        struct adjlist * t = temp->adj;
        while(t->link != NULL){
            printf("%d ,",t->frd);
            t=t->link;
        }
        printf("%d ",t->frd);
        temp = temp->next;
        printf("\n");
    }
    printf("%d - ",temp->node);
    struct adjlist * t = temp->adj;
    while(t->link != NULL && t != NULL){
        printf("%d ,",t->frd);
        t=t->link;
    }
    printf("%d \n\n",t->frd);
}

void display_frd_list(){
    int m;
    printf("Enter the ID of the member whose friend is to displayed : ");
    scanf("%d",&m);

    struct nodelist* temp = head;
    while(temp->node != m){
        temp = temp->next;
    }
    printf("\nFriend list - ");
    struct adjlist* t = temp->adj;
    while(t->link != NULL){
        printf("%d ",t->frd);
        t = t->link;
    }
    printf("%d \n\n",t->frd);
}
```

```c
int main(){
    printf("Working on initial network....\n");
   add_member();
   add_member();

   add_edge();
   add_edge();

   display();

   printf("-----------Welcome to Social Network functionality Program----------------\n");
   printf("You can perform the following functions in the network : \n");
   printf("1. Add a new member to the network.\n");
   printf("2. Make friend.\n");
   printf("3. Display the friend list of a member.\n");
   printf("4. Find the mutual friends of any two member\n");
   printf("5. Find a member with maximum number of connections\n");
   printf("6. Display the graph which represent the current network\n");
   printf("7. Exit");
   printf("-------------------------------------------------------------------------------\n");

    int ans = 1;
    while(ans){
        int ch;
        printf("Enter your choice : ");
        scanf("%d",&ch);

        switch (ch)
        {
        case 1:
            add_member();
            printf("For a new member you need to add atlest one friend \n");
            add_edge();
            break;
        case 2:
            add_edge();
            break;
        case 3:
            display_frd_list();
            break;
        case 4:
            mutual_frd();
            break;
        case 5:
            max_conn();
            break;
        case 6:
            display();
            break;
        case 7:
            ans = 0;
            break;
        default:
            printf("Enter a valid choice !! \n");
        }

    }

return 0;
}
```

OUTPUT



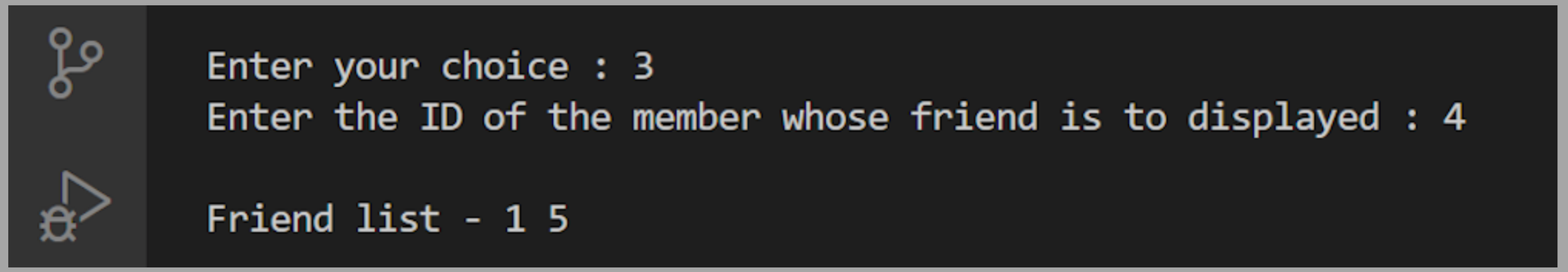
- Adding members to the network:



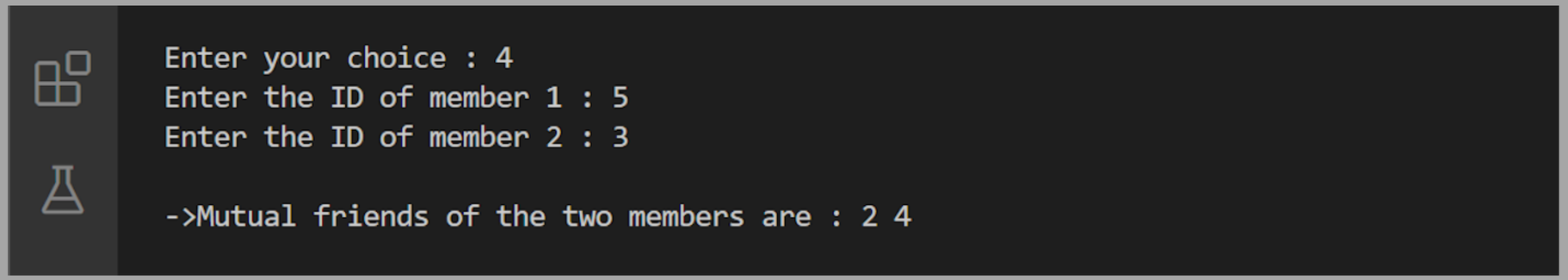
- Adding new members to the existing network:

```
-----------Welcome to Social Network functionality Program----------------
You can perform the following functions in the network :
1. Add a new member to the network.
2. Make friend.
3. Display the friend list of a member.
4. Find the mutual friends of any two member
5. Find a member with maximum number of connections
6. Display the graph which represent the current network
7. Exit--------------------------------------------------------------------------------
Enter your choice : 1
Enter the ID of new member : 3
->Member added successfully
For a new member you need to add atlest one friend
Enter the ID of member : 3
Enter the ID of friend member : 2
->Friend added successfully
Enter your choice : 1
Enter the ID of new member : 4
->Member added successfully
For a new member you need to add atlest one friend
Enter the ID of member : 1
Enter the ID of friend member : 4
->Friend added successfully
Enter your choice : 2
Enter the ID of member : 4
Enter the ID of friend member : 1
->Friend added successfully
Enter your choice : 2
Enter the ID of member : 2
Enter the ID of friend member : 4
->Friend added successfully
Enter your choice : 6
Member   -    Friends
--------------------
1 - 2 ,4
2 - 1 ,4
3 - 2
4 - 1
```

- Displaying friend list of a member:



- Finding mutual friends of any two members:



- Finding a member with maximum number of friends:

```
Enter your choice : 5

->5 has maximum connections
```

- Displaying graph representing the network:

```
Enter your choice : 6

Member   -    Friends
---------------------
1 - 2 ,4
2 - 1 ,4
3 - 2 ,4
4 - 1 ,5
5 - 2 ,3 ,4
```


Thank you!