To start the program, use the following commands:

Compile the program: **make**

Run the program: **make run**

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/HW8$ make
javac Person.java
javac SocialNetworkGraph.java
javac Main.java
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/HW8$ make run
java Main
Person added: john doe (Age: 25, Hobbies: [reading, hiking, cooking], Timestamp: Sun May 26 23:31:45 TRT 2024)
Person added: jane smith (Age: 22, Hobbies: [swimming, cooking], Timestamp: Sun May 26 23:31:45 TRT 2024)
Person added: alice johnson (Age: 27, Hobbies: [hiking, painting], Timestamp: Sun May 26 23:31:45 TRT 2024)
Person added: bob brown (Age: 30, Hobbies: [reading, swimming], Timestamp: Sun May 26 23:31:45 TRT 2024)
Person added: emily davis (Age: 28, Hobbies: [running, swimming], Timestamp: Sun May 26 23:31:45 TRT 2024)
Person added: frank wilson (Age: 26, Hobbies: [reading, hiking], Timestamp: Sun May 26 23:31:45 TRT 2024)
Friendship added between john doe and jane smith
Friendship added between john doe and alice johnson
Friendship added between jane smith and bob brown
Friendship added between emily davis and frank wilson
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option:
```

Starts my programme with this data for testing :

```
// Adding some people for demonstration
network.addPerson(name:"John Doe", age:25, Arrays.asList(...a:"reading", "hiking", "cooking"));
network.addPerson(name:"Jane Smith", age:22, Arrays.asList(...a:"swimming", "cooking"));
network.addPerson(name:"Alice Johnson", age:27, Arrays.asList(...a:"hiking", "painting"));
network.addPerson(name:"Bob Brown", age:30, Arrays.asList(...a:"reading", "swimming"));
network.addPerson(name:"Emily Davis", age:28, Arrays.asList(...a:"running", "swimming"));
network.addPerson(name:"Frank Wilson", age:26, Arrays.asList(...a:"reading", "hiking"));

// Adding friendships for demonstration
network.addFriendship(name1:"John Doe", name2:"Jane Smith");
network.addFriendship(name1:"John Doe", name2:"Alice Johnson");
network.addFriendship(name1:"Jane Smith", name2:"Bob Brown");
network.addFriendship(name1:"Emily Davis", name2:"Frank Wilson");
```

1. **Add Person:**

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 1
Enter name: Sehmus Acar
Enter age: 50
Enter hobbies (comma-separated): running,swimming
Person added: Sehmus Acar (Age: 50, Hobbies: [running, swimming], Timestamp: Sun May 26 23:04:52 TRT 2024)
```

2. **Remove Person:**

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 2
Enter name: Sehmus Acar
Person removed: Sehmus Acar (Age: 50, Hobbies: [running, swimming], Timestamp: Sun May 26 23:04:52 TRT 2024)
```

3. **Add Friendship:**

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 3
Enter first person's name: jane smith
Enter second person's name: john doe
Friendship added between jane smith and john doe
```

4. **Remove Friendship:**

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 4
Enter first person's name: jane smith
Enter second person's name: john doe
Friendship removed between jane smith and john doe
```

5. **Find Shortest Path:**

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 5
Enter start person's name: alice johnson
Enter end person's name: bob brown
Shortest path: alice johnson -> john doe -> jane smith -> bob brown
```

6. **Suggest Friends:**

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 6
Enter person's name: alice johnson
Enter maximum number of friends to suggest: 3
Suggested friends for alice johnson:
jane smith (Age: 22, Hobbies: [swimming, cooking], Timestamp: Sun May 26 23:28:39 TRT 2024) (Score: 1.0, 1 mutual friends, 0 common hobbies)
frank wilson (Age: 26, Hobbies: [reading, hiking], Timestamp: Sun May 26 23:28:39 TRT 2024) (Score: 0.5, 0 mutual friends, 1 common hobbies)
bob brown (Age: 30, Hobbies: [reading, swimming], Timestamp: Sun May 26 23:28:39 TRT 2024) (Score: 0.0, 0 mutual friends, 0 common hobbies)
```

7. **Count Clusters:**

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 7
Cluster 1:
jane smith
john doe
bob brown
alice johnson
Cluster 2:
emily davis
frank wilson
Number of clusters found: 2
```

1. Shortest Path Algorithm

This algorithm finds the shortest path between two people using the Breadth-First Search (BFS) method. BFS works by exploring the graph layer by layer.

Steps:

1. Start with the first person (start).
2. Create a queue and add the first person to it.
3. Take the person at the front of the queue and visit all their friends.
4. If one of these friends is the target person (end), stop and return the path.
5. Add friends who haven't been visited to the queue and keep track of the previous person.
6. Repeat step 3 until the queue is empty.
7. If the target person is found, there's a path. If not, there's no path.

2. Count Clusters Algorithm

This algorithm counts the connected groups (clusters) of people in the social network using BFS or Depth-First Search (DFS). A cluster is a group of people who are all connected, directly or indirectly.

Steps:

1. Pick an unvisited person and use BFS/DFS to visit everyone connected to them.
2. This group of people forms a cluster. Increase the cluster count.
3. Mark the people as visited.
4. Find another unvisited person and repeat from step 1.
5. Keep going until everyone is visited.
6. Return the total number of clusters.

3. Friend Suggestion Algorithm

This algorithm suggests new friends by looking at a person's friends and their friends.

Steps:

1. Choose the person who needs friend suggestions (target person).
2. Find this person's friends and their friends.
3. For each potential new friend, score them based on mutual friends and shared hobbies.
4. Sort the potential friends by their scores and suggest the top ones.