1. **Identification**

   Clearly print your full name.

   ![Full Name]

2. **Program Grading**

   TA: Check the boxes below to indicate which functions of the student’s program perform properly.

   - **Forming the Peer-to-Peer Network**
     - **Peer registers periodically with peer cache**
       - Start the student peer. Use debugging output on peer cache to verify that the peer contacts the cache regularly.
       - 5 points
     - **Peer maintains 3 neighbors regularly**
       - Start 8 solution peers and then start the student peer on a different machine. Use the neighbors command on both the student and solution peers to check that the student peer has only 3 neighbors. Have one of the student peer’s neighbors exit. Use the neighbors command to verify that after 60 seconds the student peer has established a connection to a new neighbor. No exceptions should cause a thread to crash.
       - 10 points
     - **Peer accepts new neighbors**
       - Start student peer first and use the neighbors command to verify it has no neighbors. Now start a solution peer and use the neighbors command to check that the student peer now has a connection. Also check debugging output from the solution peer to verify that the solution peer initiated the connection. The student peer should not open a connection back to the solution peer if the solution peer opened the connection first (it should be able to use the peerid message from the solution peer to determine that it is already connected to it).
       - 10 points
     - **Peer treats all neighbors the same**
       - Start 3 solution peers and then start the student peer. Use the neighbors command to check whether peer has only 3 neighbors. Start 3 more solution peers and use the neighbors command to check that the student peer now has more neighbors that have connected to it. Start exiting solution peers that are neighbors of the student peer. The student peer should not create any new neighbors until it gets below 3 peers. The student peer should be able to maintain 3 neighbors unless there are fewer than 3 peers active in the network.
       - 5 points
   
   - **Searching the Peer-to-Peer Network**
     - **Peer can send a search message and receive a match**
       - Start 8 solution peers, with just one of them using zappala.ged and the others having no gedcom files. Start a student peer on a different machine and use the student peer to search for "surname=Ardiz" with a scope of 5 hops. Use debugging on solution peers to check whether the student peer sends the search to each neighbor only once. The student peer should display matching Gedcom information for Grazia Angela Ardizzone and Gregorio Ardizzone. Sending a second search for the same information returns the matching information again (requires peer to use a different ID for the search). Quitting the peer, starting the peer, and issuing the same search again works (requires peer to use a random ID in each search)
       - 10 points
     - **Peer can return a match for a search**
       - Start 7 solution peers with no gedcom files. Start student peer with zappala.ged. Use a solution peer to search for 'deathrange=1840-1910:birthrange=1800-1830:name=Giuseppe" with a scope of 5 hops. Solution peer should display Gedcom information for Giuseppe Zappala'.
       - 10 points
     - **Peer can forward search messages and return match messages over multiple hops**
       - Start 7 student peers with no gedcom files and one student peer with zappala.ged. Use the neighbors command to find a student peer that is at least two hops away from the peer with the gedcom file. Do a search for “surname=Ardiz” from this student peer with a scope of 5 hops. This peer should display matching Gedcom information for Grazia Angela Ardizzone and Gregorio Ardizzone.
       - 10 points
Peer will stop duplicate messages.
Start one solution peer and 7 student peers with no gedcom files. Use a student peer to do a search for "surname=test" with a scope of 100 hops. Use debugging output on the solution peer to verify that it does not receive excessive duplicates.

Peer will expire a search message with a scope of 0
Start 7 solution peers and one student peer, with no gedcom files on any peers. Use the neighbors command to determine a neighbor of the student peer. Send a search message from this neighbor for "surname=test" with a scope of 1 hop. Use debugging output on solution peers to verify that student peer does not forward the message to anyone.

Identifying Peers on the Network

Peer sends peerid message periodically
Start one solution peer and one student peer. Use debugging output on solution peer to verify that student peer periodically sends its peer ID. Also check the case when the student peer starts first.

Looking Up GEDCOM Information

Peer responds to family and person queries
Start one solution peer and one student peer with zappala.ged. Use the solution peer and the person command to request the person information for @I16@. The student peer should return information for Mariano Zappala. Now use the solution peer and the family command to request the family information for @I16@. The student peer should return information for families @F6@ (husband Mariano Zappala and Maria Nicotra) and @F8@ (husband Mariano Zappala and Grazia Angela Ardizzone).

Error Handling and Quitting

Peer can handle peer-cache failure
Start one solution peer and one student peer. Use debugging output from the solution peer to verify that they are connected as neighbors. Close the peer-cache and restart it. Quickly start a new solution peer before the others have registered. Use the neighbors command to verify that the student peer is eventually able to find the new solution peer and connect to it. No exceptions should cause a thread crash.

Peer can handle gedcom failure
Start one solution peer with zappala.ged and one student peer. Use the student peer to do a family query for @I16@. Exit the solution peer. Use the student peer to do a family query again. Verify that the student peer does not hang or crash due to an exception. Start the solution peer again and verify that the student peer can do a family query for @I16@ using the solution peer’s new peer ID.

Peer can quit properly
Start 7 solution peers and one student peer. Use the neighbors command on the student peer to verify that it has some neighbors. Verify that the student peer can quit and exit all threads within 10 seconds and without any errors.

3. Program Score

TA: Fill in the information below when the student agrees to turn in the program with this score. Have the student sign below and then turn the sheet in to the professor.

<table>
<thead>
<tr>
<th>TA Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Today’s Date:</td>
<td>Date Code Last Modified:</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
</tr>
</tbody>
</table>

4. Student Signature

By signing here you agree to turn in this version of your program for the above score. You may not turn in another version at a later point.
5. **Professor Grading**

The professor will fill in the following before returning this sheet to the student:

<table>
<thead>
<tr>
<th>Late Days At 10% Off:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Score:</td>
</tr>
</tbody>
</table>