Project Grading Policy

- Each project group will record a video demo, preferably 5-8 minutes in duration.
- If the video file is not very big, you can upload it to Canvas. Otherwise, you can upload the video to onedrive@UF. Make sure you give access right to view your video. When uploading your source files on Canvas, please include a readme file, which should contain an URL for your video file (if the video file is not on Canvas).
- The video file should be dated no later than the project submission deadline.
- If your program can't work fully correctly, you need to show us what requirements your program can meet and what requirements your program can't, so we can give you as much grade as possible. You also need to document who has done what in the readme file and explain your situation during the demo. This helps each of you to receive partial credits based on your code.
- Each group only needs to make one submission. Please write down your group members in the readme file.

Since we grade your project based on your video, you are expected to show your program meet the following rubric:

1. Start the peer processes: 35%
   a) In your video, you should clearly show that your program will read the Common.cfg and PeerInfo.cfg to correctly set the related variables.
   b) In your video, you should clearly show that your program will let each peer make TCP connections to all peers that started before it.
   c) When a peer is connected to at least one other peer, it starts to exchange pieces as described in the protocol description section. A peer terminates when it finds out that all the peers, not just itself, have downloaded the complete file.
2. After connection: 30%
   a) Handshake message: Whenever a connection is established between two peers, each of the peers of the connection sends to the other one the handshake message before sending other messages.
   b) Exchange bitfield message.
   c) Send 'interested' or 'not interested' message.
   d) CORRECTLY send *k* 'unchoke' and 'choke' messages every *p* seconds.
   e) Set optimistically unchoked neighbor every '*m*' seconds.
3. File exchange: 30%
   a) Send 'request' message.
   b) Send 'have' message.
   c) Send 'not interested' message.
   d) Send 'interested' message.
   e) Send 'piece' message.
   f) Receive 'have' message and update related bitfield.
4. Stop service correctly. 5%

Please note that the best way to let us understand your program is to print a clear log. During the video, you need to SHOW and EXPLAIN to us that your program meets all these requirements ONE BY ONE. As an example, your program may produce the following log when you start the peers.

P1:
At t1, P1 Start, set variables to xxx, bitfield to xxx.
At t3, TCP connection is built between P1 and P2
At t5, TCP connection is built between P1 and P3

P2:
At t2, P2 Start, set variables to xxx, bitfield to xxx.
At t3, TCP connection is built between P1 and P2
At t6, TCP connection is built between P2 and P3

P3:
At t4, P3 Start, set variables to xxx, bitfield to xxx.
At t5, TCP connection is built between P1 and P3
At t6, TCP connection is built between P2 and P3