1. **Can I make changes to the starter code?**
   You can make small and minor changes to the starter code as long as you are not changing its functionality.

2. **How do I verify whether the file transferred from the BIMDC directory is identical to the one in the received directory?**
   You can use the diff linux command to check differences between 2 files.
   For example, if you were to check the difference between file "9.csv" that exists under both the received and the BIMDC directory, you would run the following command in the shell:

   diff received/9.csv BIMDC/9.csv

   The absence of an output upon running this command means that the files are identical. If an output exists, it indicates the difference between the two files.

   Additionally, say you were to compare the first 1000 lines of the two files, you would include the head command as follows:

   diff <(head -n 1000 received/9.csv) <(head -n 1000 BIMDC/9.csv)

3. **Do I need to manually create the received directory to store my files?**
   You can create this folder manually or during program execution.

4. **When requesting a new channel, the document says to use the "-c" flag. Does this "-c" flag have any value associated with it?**
   No, the c flag does not take in any value associated with it. It is simply an indication for the client to request the server for a new channel. Upon a successful new channel request, the server creates a new channel and returns its name. The client can join this new channel using this returned name.

5. ```
   filemsg fm(0, 0);
   string fname = "teslkansdlkjflasjdf.dat";
   int len = sizeof(filemsg) + fname.size() + 1;
   char buf2[len];
   memcpy(buf2, &fm, sizeof(filemsg));
   strcpy(buf2 + sizeof(filemsg), fname.c_str());
   chan.cwrite(buf2, len);
   ```

   I don't really understand this piece of the starter code well. What exactly is going on?
   The objective of this code is to send a message to the server asking for the size of the file "`teslkansdlkjflasjdf.dat`". To do this - we construct a filemsg object with offset and length parameters set to 0 and append the filename to this filemsg object; we then send this (filemsg object + filename) across a character buffer to the server. Note
   ```

that the size of the buffer (len) is `sizeof(filemsg) + fname.size() + 1`; we add 1 for the *'\0'* character which indicates the end of the character array (this is implicitly added for you, you must not explicitly add this character to the end of the filename).

The purpose of memcpy is to copy over the filemsg object to the character buffer such that it occupies the first (sizeof(filemsg)) bytes in the buffer. We then use strcpy to copy over our filename to occupy the rest of the buffer. Note the `buf2 + sizeof(filemsg)` parameter in strcpy - this indicates that we want to copy our string starting from the byte after the end of the filemsg object (so that we do not overwrite the filemsg object present in the buffer itself).

6. **I see strange/random characters in the file transferred to the received/ directory. What could be the reason for this?**
   It's likely that you're writing the receive data as a string rather than using the `fwrite(...)` system call to transfer over binary files. Another reason could be that you are putting in an incorrect offset or length parameter.

7. **Should I handle incorrect / invalid inputs in my code?**
   No, you can assume all inputs are valid.

8. **When running the server as a child process of the client, how do I pass the command line arguments (specifically -m for the common buffer capacity) to the server?**
   You must use `execvp(...)` for this functionality.