COMP 2012H Honors Object-Oriented Programming and Data Structures

Assignment 4 Integer Factorization

Honor Code

We value academic integrity very highly. Please read the Honor Code section on our course webpage to make sure you understand what is considered as plagiarism and what the penalties are. The following are some of the highlights:

- Do NOT try your "luck" - we use sophisticated plagiarism detection software to find cheaters. We also review codes for potential cases manually.
- The penalty (for **BOTH** the copier and the copiee) is not just getting a zero in your assignment. Please read the Honor Code thoroughly.
- Serious offenders will fail the course immediately, and there will be additional disciplinary actions from the department and university, upto and including expulsion.

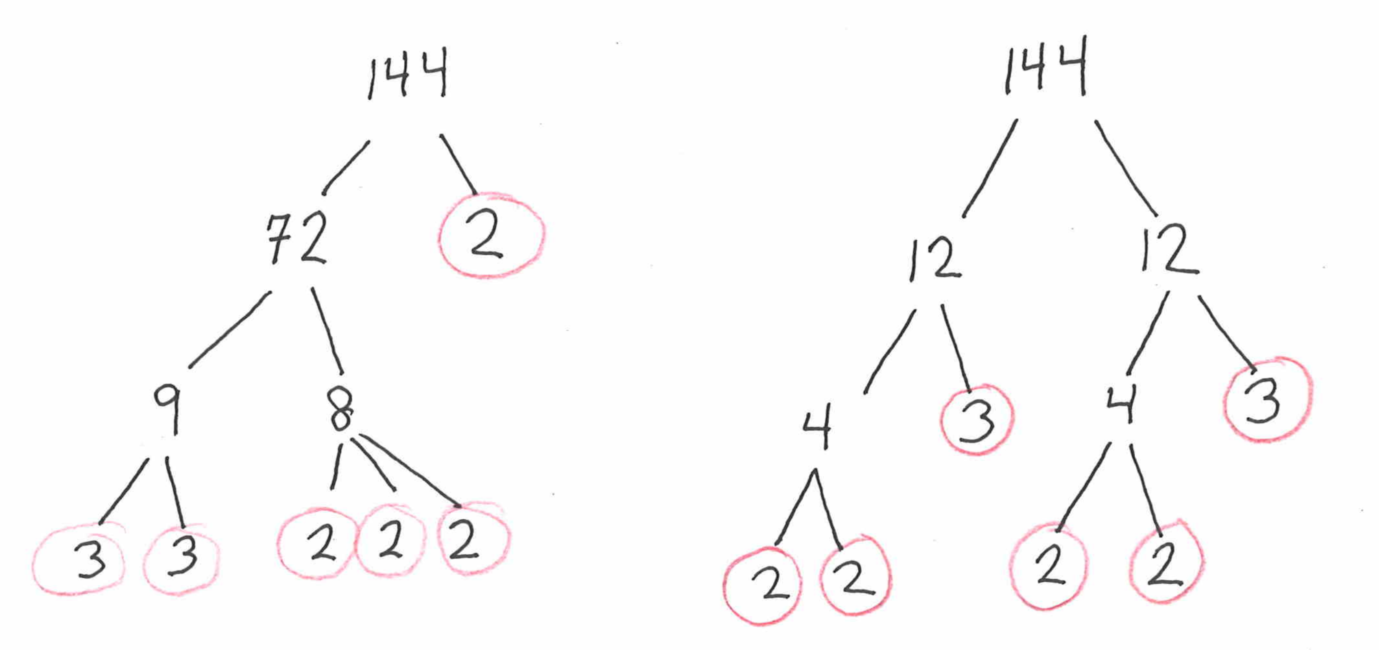
End of Honor Code

Page maintained by

SONG Sizhe
sizhe.song@connect.ust.hk
Last Modified:
09/30/2022 13:41:59

Homepage

Course Homepage



Introduction

In this programming assignment, you will have a taste of speed programming. You are going to solve a classic problem - integer factorization - which is a key in important applications such as RSA public-key encryption in a speedy manner.

Please read the FAQ section regularly, and do check it one day before the deadline to make sure you don't miss any clarification, even if you have already submitted your work by then. You can also raise questions on Piazza and remember to use the "pa4" tag.

End of Introduction

Description

Basically, we want to perform factorization for all integers in a given range and print the results.

The input arguments will be from the command line. You may read this article to learn about how you might retrieve them in C++. To provide command line arguments to your program you will have to compile and run your program in a terminal. Please feel free to do that in any terminal you are familiar with. However, if you want to time your program, you may want to use our lab 2 Linux machines. Instructions can be found in the next section: Timing your program.

Assuming the name of your program is `factorize`, we may use the following command in the terminal:

```
factorize [range_start] [range_end]
```

The **[range_start]** and **[range_end]** are the integers that specify the range. For example, `factorize 12 23` will print out the factorization results for all integers in the range [12, 23] in ascending order like so:

```
12=2*2*3
13=13
14=2*7
15=3*5
16=2*2*2*2
17=17
18=2*3*3
19=19
20=2*2*5
21=3*7
22=2*11
23=23
```

The output is pretty straightforward as you can see: each line consists of the integer being factorized at the beginning, then the equal sign, and finally the factorization result with a single `endl` character added at the end. The prime factors, which are separated by asterisks, in the factorization result must also be listed in ascending order.

Your program should output nothing else and require no user interaction. (i.e. don't need to take in any standard console input with `cin`)

For simplicity, you can assume the provided input arguments **[range_start]** and **[range_end]** are positive integers between 2 and 2^{63} - 1 inclusively (i.e. it can be stored in a `long long` variable), and that **[range_start]** is no larger than **[range_end]**.

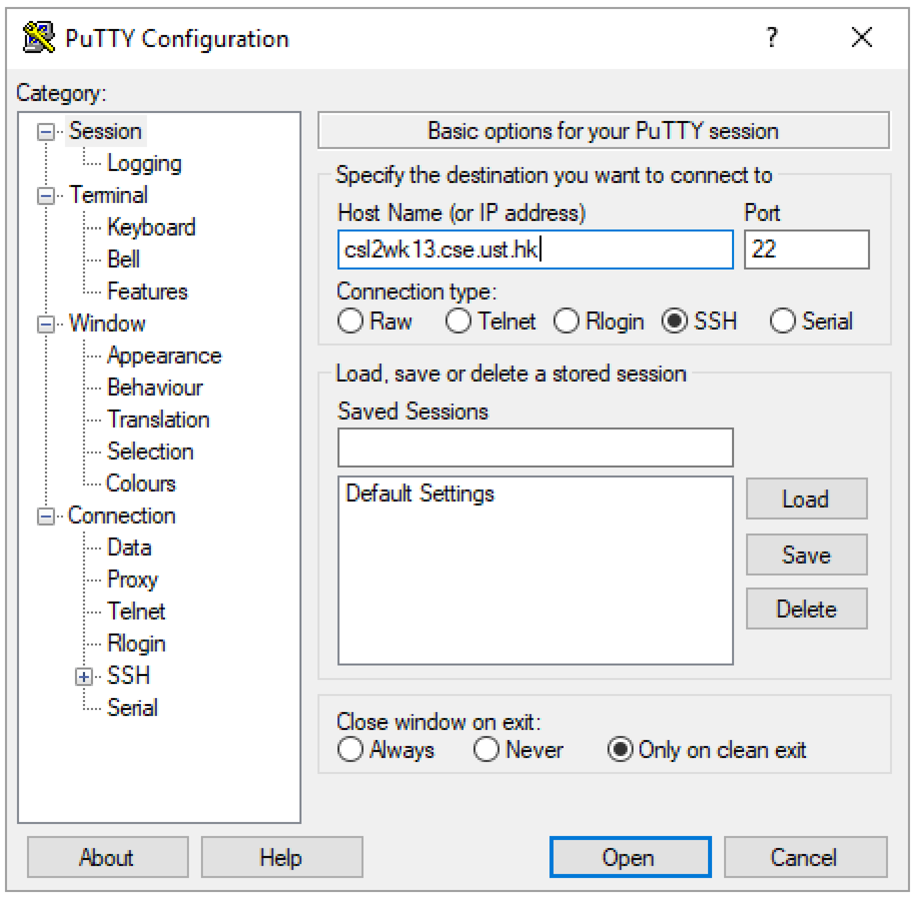
End of Description

Timing your program

We will be using the `time` Linux utility to measure the runtime performance of your program. You may use this utility on any of our CS Lab 2 (Linux Lab) machines either locally in the lab or remotely via SSH. The following shows you how using the SSH client in the Virtual Barn. You may adjust the steps accordingly if you are familiar with installing and using a SSH client on your own computer.

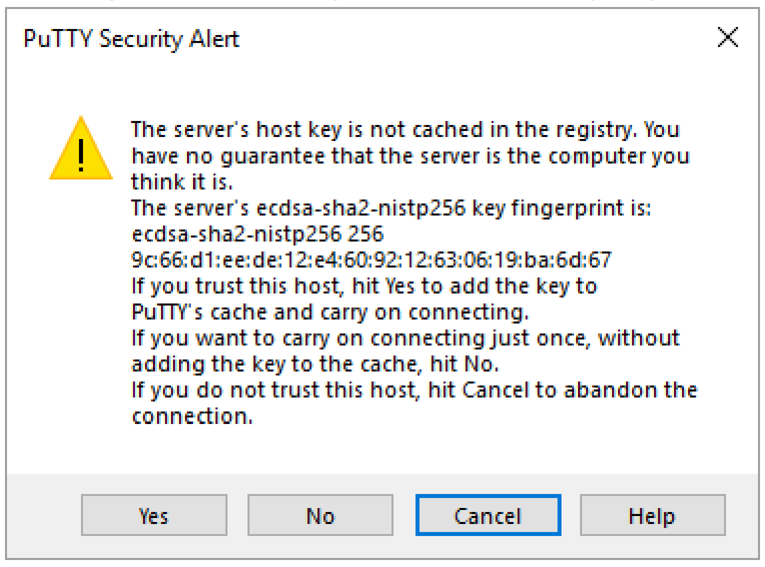
1. Remote control a Windows machine in HKUST virtual barn. After following the installation guide and user guide there to install the remote control client and connect to the HKUST network, choose the Programming Software server when you are offered a selection of different servers.
2. Copy your files to Virtual Barn. If you have no idea how, you can email yourself the files and then download them there.
3. Open `PuTTY`. There is a shortcut on your Desktop. Enter `csl2wk13.cse.ust.hk` as the Host Name. Note: If that one server doesn't work, you may try other servers such as

`csl2wk14.cse.ust.hk` and `csl2wk15.cse.ust.hk`. The server number goes up to at least 40.



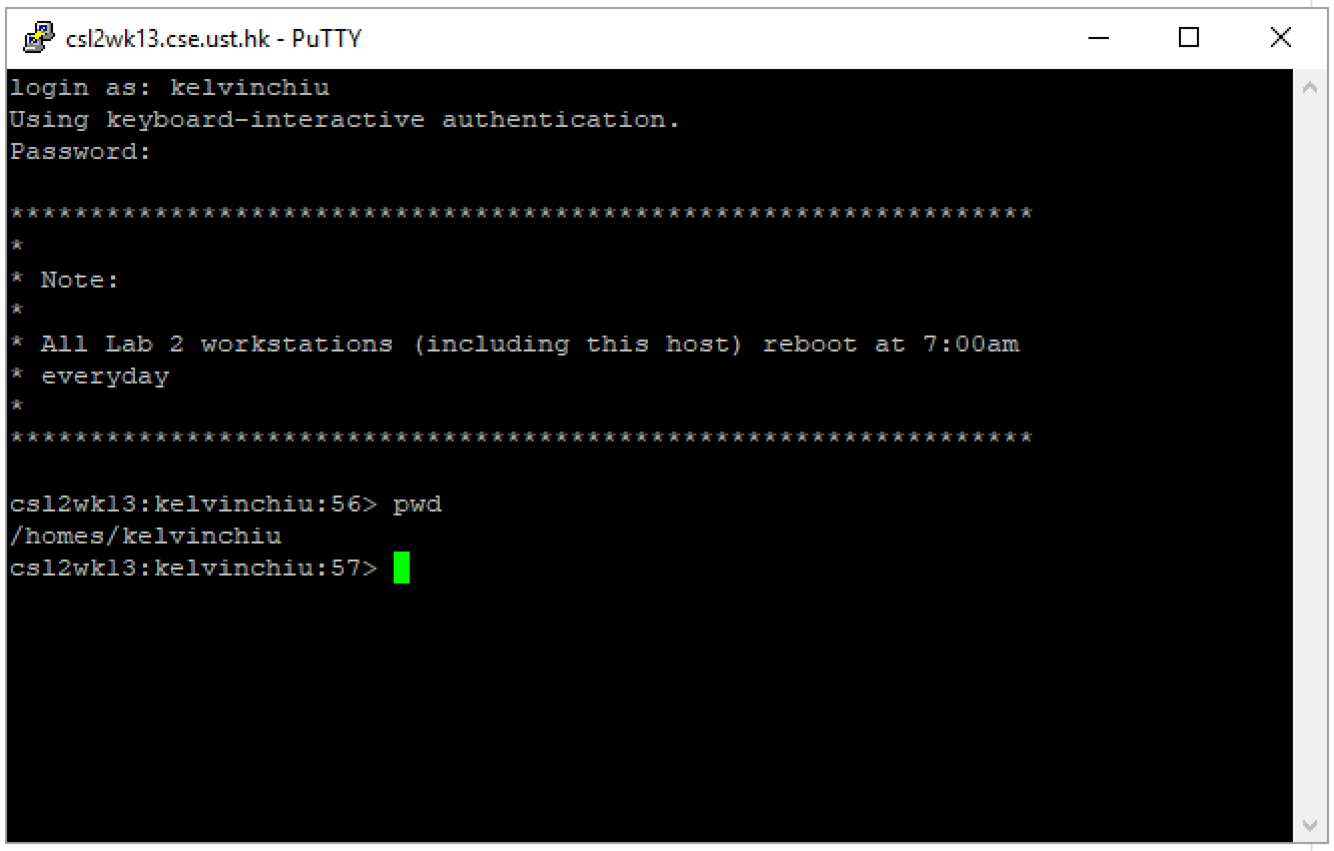
Specify the destination you want to connect to
Host Name (or IP address): csl2wk13.cse.ust.hk
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

4. Click Open. If a warning about the security key shows up, click Yes to dismiss it.



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's ecdsa-sha2-nistp256 key fingerprint is:
ecdsa-sha2-nistp256 256
9c:66:d1:ee:de:12:e4:60:92:12:63:06:19:ba:6d:67
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

5. Enter your CSD login and password. (Register one if you haven't done so in lab 1 by following the lab 1 instructions.) It may not show what you type when you are typing in the password. Don't worry, just type your password and hit the Enter key. You should login to your *home* directory. Enter the command `pwd` (Print Working Directory) to verify this. It should show `/homes/your_login`



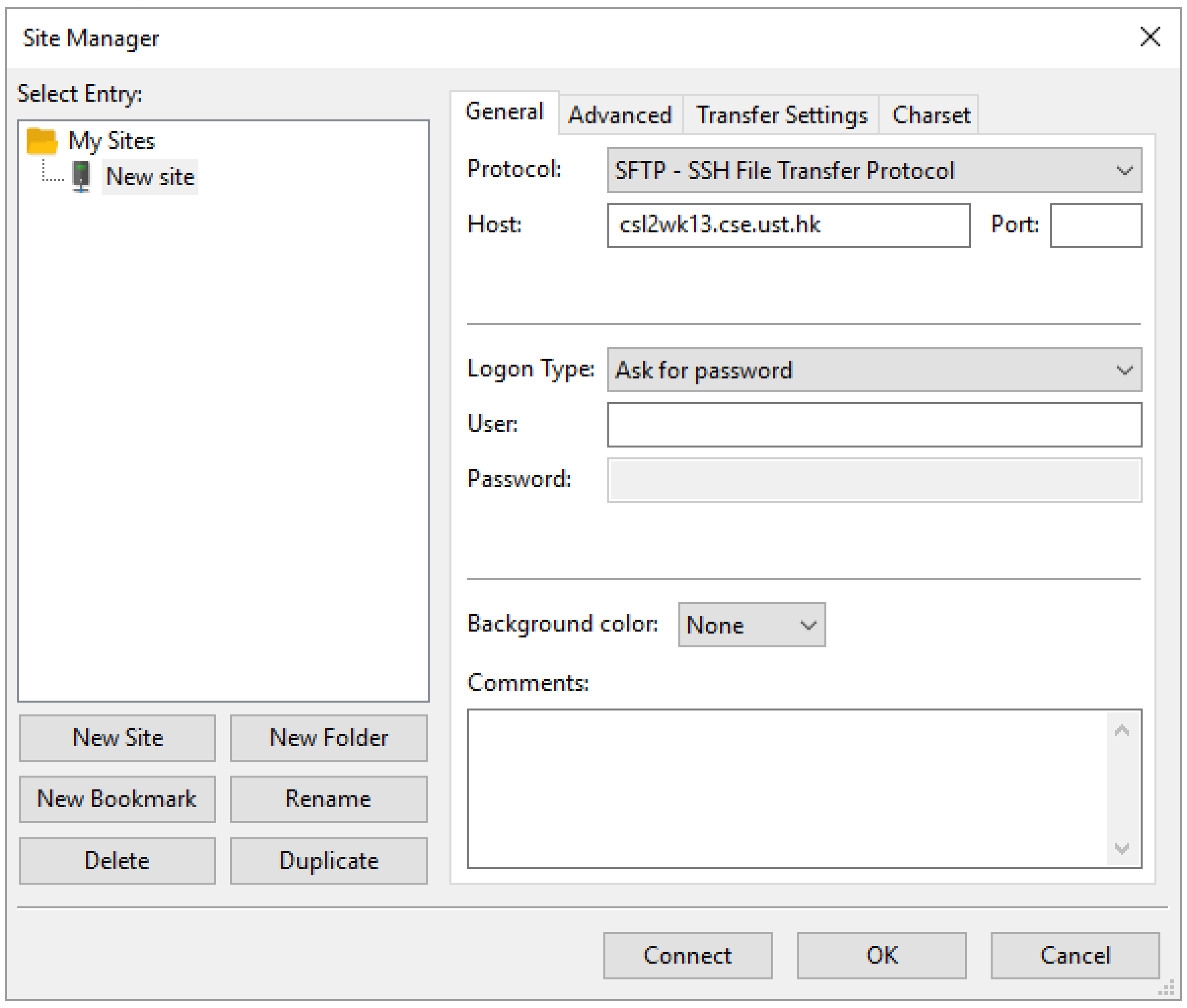
```
login as: kelvinchiu
Using keyboard-interactive authentication.
Password:

*******************************************************************
*
* Note:
*
* All Lab 2 workstations (including this host) reboot at 7:00am
* everyday
*
*******************************************************************

csl2wk13:kelvinchiu:56> pwd
/homes/kelvinchiu
csl2wk13:kelvinchiu:57>
```

6. Open `FileZilla` from Start Menu. Click "File" -> "Site Manager" -> "New Site". Choose **SFTP** for protocol according to the screenshot below. Enter `csl2wk13.cse.ust.hk` (or whatever server you used in a previous step) as Host. Choose "Ask for password" for the

Logon Type.



Protocol: SFTP - SSH File Transfer Protocol
Host: csl2wk13.cse.ust.hk Port:
Logon Type: Ask for password
User:
Password:
Background color: None
Comments:

7. Click Connect and then enter your CSD login and password.
8. If the connection succeeds, the right side should show your home directory at Linux Lab. Copy your factorize.cpp file by dragging it to the right side. That would upload the file to `/homes/your_login`. You can close `FileZilla` after doing so.
9. Go back to `PuTTY`, and compile your program using the following command:

```
g++ -Wall -Wextra -pedantic -std=c++11 -o factorize factorize.cpp
```

10. If no compilation errors occur, you should be able to time and run your program by adding `/usr/bin/time -v` before the command to be timed. For example:

```
/usr/bin/time -v factorize 9234567890 9234567899
```

Output like the following will be printed:

```
csl2wk13:yourusername:69> /usr/bin/time -v factorize 9234567890
9234567899
9234567890=2*5*53*73*238681
9234567891=3*3*3*3*3*19*127*15749
9234567892=2*2*11*17*37*333667
9234567893=191*317*152519
9234567894=2*3*907*937*1811
9234567895=5*7*263844797
9234567896=2*2*2*23*47*229*4663
9234567897=3*31*103*353*2731
9234567898=2*4617283949
9234567899=43*61*3520613
                Command being timed: "factorize 9234567890
9234567899"
                User time (seconds): 10.64
                System time (seconds): 0.00
                Percent of CPU this job got: 99%
                Elapsed (wall clock) time (h:mm:ss or m:ss): 0:10.65
                Average shared text size (kbytes): 0
                Average unshared data size (kbytes): 0
                Average stack size (kbytes): 0
                Average total size (kbytes): 0
                Maximum resident set size (kbytes): 3244
                Average resident set size (kbytes): 0
                Major (requiring I/O) page faults: 0
                Minor (reclaiming a frame) page faults: 123
                Voluntary context switches: 2
                Involuntary context switches: 5
                Swaps: 0
                File system inputs: 0
                File system outputs: 0
                Socket messages sent: 0
                Socket messages received: 0
                Signals delivered: 0
                Page size (bytes): 4096
                Exit status: 0
```

For timing purpose, we will be using the **"User time (seconds)"**. In this example, it took our unoptimized program 10.64 seconds to factorize ten integers between 9234567890 and 9234567899.

End of Timing your program

Sample program

On a lab 2 Linux machine (you can access it according to the instructions in Timing your program), you can download our unoptimized sample program with this command:

```
wget
https://course.cse.ust.hk/comp2012h/assignments/PA4/Resources/factorize
```

Note: the downloaded file may be renamed to something else if you already have a file named "factorize" in the current directory - see the output of the wget command to find out what the sample program has been renamed to.

Then you can add the executable permission to it:

```
chmod u+x factorize
```

Finally you may run it as usual:

```
factorize 12 20
```

Windows demo program is provided, but please keep in mind that we grade your codes in Linux lab, so it is recommended that you check your codes in Linux at least once before the deadline.

Sample program

Sample output

Your finished program should produce the same output as our sample output. The file names indicate the input arguments used. Please note that sample output, naturally, does not show all possible cases. It is part of the assessment for you to design your own test cases to test your program. You may use our sample program to generate more test cases for yourself. **Be reminded to remove any debugging message** that you might have added before submitting your final submission.

Sample output

Submission & Grading

Deadline: 8 October 2022 Saturday HKT 23:59.
Submit a single source file `factorize.cpp` to ZINC.

Cite External Materials

If you have referred to any external materials, please cite them in your ZINC submission. Add a comment at the beginning (line 1) of the source file. If you have already submitted, please kindly add the references (if any) and submit again. It is fine to learn ideas and algorithms from external materials, but you have to write the codes yourself.

Grading scheme

- Your program should generate the exact same output as our sample program's.
- Your program should run no significantly slower than our unoptimized sample program. That means its running time cannot be more than double of that of ours. For example, if our sample program can generate the results in 5 seconds, your program should not take more than 10 seconds to do the same.

You need to fulfill both requirements to be considered as correct for a test case. We will test your program on multiple hard test cases (very large number) for multiple times and the result (average time for one test case) will be released. This will happen outside ZINC as it doesn't support timing. In ZINC, we only provide a simple test case that factorizes 2 to 100 to help you debug.

Speed programming competition (Bonus)

For this assignment, we will also reward the students that have mastered the art of speed programming up to 10 bonus points.

- Your last submission on ZINC will be used for the ranking. No separate submission will be accepted.
- Late submission will be excluded from ranking.
- We only rank those programs with correct outputs. If your program does wrong in any test cases, it will be excluded from ranking.
- Programs running not that fast will be excluded from ranking. If your running time > 1.1 × running time of the demo program, it will not be ranked.

Hardwares for grading

Currently, the plan is to grade the submissions on CS Linux Lab machines, so:

- Please make sure your program can be compiled with this command on the above mentioned machines
  `g++ -Wall -Wextra -pedantic -std=c++11 -O3 factorize.cpp -o factorize`
- The above mentioned machines each has 8GB memories in total, and definitely we can never occupy all of them, so make sure your program's memory use is reasonable. Maybe you can test it yourself before the final submission.
- Your program will not be able to access the network.

End of Submission & Grading

Frequently Asked Questions

Q: My code doesn't work, here it is, can you help me fix it?
A: As the assignment is a major course assessment, to be fair, you are supposed to work on it by yourself and we should never finish the tasks for you. We are happy to help with explanations and advice, but we are **not allowed** to directly debug for you.

Q: Can I refer to external materials?
A: It is fine to learn ideas and algorithms from external materials, but you have to write the codes yourself. Also, please cite the materials you referred to in your submission. Add a comment at the beginning (line 1) of the source file.

End of FAQ