DUE DATE: **By the deadline specified on Canvas**

CSE-381: Systems 2

<u>Exercise #1</u>

Max Points: 20

You should save/rename this document using the naming convention MUid.docx (example: `raodm.docx`).

<u>Objective</u>: The objective of this exercise is to review the following concepts from CSE-278

1. Basics of working and programming under GNU/Linux from CSE-278
2. Review using NetBeans for remote programming
3. Review the CSE department's programming style requirements
4. Recap the use of debugger
5. Review basics of File I/O and command-line arguments
6. Understand the procedure for submitting programs

Fill in answers to all of the questions. For some of the questions you can simply copy-paste appropriate text from the Terminal/NetBeans window into this document. You may discuss the questions with your neighbor or your instructor.

Name: RAObot

Part #0: Setup NetBeans

Estimated time: 15 minutes

Note: In order to access the remote GNU/Linux server **`os1.csi.miamioh.edu`** setup for this course, you will need to use VPN software. Download and install VPN client from Miami's VPN website. You will also need to approve VPN access via Duo for 2-factor authentication. You will need to approve your VPN access via Duo quickly for the VPN connection to be established successfully.

Exercise: Complete this part of the exercise via the following steps:

1. In case you have not already done so (you may have used NetBeans in CSE-278), Download and install NetBeans version 8.2 from https://netbeans.org/images_www/v6/download/community/8.2/ . It may be best to download the distribution titled "`All`".

2. Play videos on setting up "*Basic setup of NetBeans Plugins*" from Video Demonstrations page on Canvas. Follow the procedure shown in the video to configure NetBeans. Note: This course will use the following Linux server (accessible via VPN) throughout this course. You should memorize the name of this server: **`os1.csi.miamiOH.edu`**.

DUE DATE: **By the deadline specified on Canvas**

Part #1: Review basic Linux `bash`-shell skills

Estimated time: 20 minutes

Exercise: This course will use the following Linux server throughout this course. You should memorize the name of this server: **`os1.csi.miamiOH.edu`**. For off-campus access, you will need to use Virtual Private Network (VPN) due to security concerns. You can get VPN software for free from: http://miamioh.edu/vpn. VPN also need 2-factor authentication via Duo.

Exercise: Complete this part of the exercise via the following steps:

1. First review the video titled "*Using the built-in NetBeans terminal*" via https://youtu.be/vW2vJkwebfs and open a terminal on the Linux server for this course, **`os1.csi.miamioh.edu`**. If you have trouble logging onto the `os1.csi.miamioh.edu`, seek help. Confirm that you are logged onto os1 via the `hostname` command run as the bash shell prompt (`$`) as and pressing the ENTER (↵) key, shown below:

```
raodm@os1:~$ hostname↵
os1
raodm@os1:~$
```

2. Double check your login – When you log onto the Linux machine, you will start off in a default directory called your **home** directory. You should create all your files and save your work off sub-directories under your home directory. To figure out what your home directory is, you need to use the `pwd` (present working directory) command (that is, type `pwd` at the shell (`$`) prompt and press ENTER key, which is indicated by ↵) as shown below:

```
raodm@os1:~$ pwd ↵
```

Note: The home directory will be in the format `/home/MUID` (where `MUID` is your Miami University unique ID), example: `/home/raodm`. In addition, note the following important terminology associated with directory hierarchies:

- **Absolute path**: In Linux, paths always start with a `/` (forward slash or just slash, *i.e.*, the division sign) indicating the root directory. Example: `/home/raodm`, `~/`, or `/usr/bin/ls` etc.
- **Relative path**: Paths that **do not start** with a `/` are relative paths. Relative paths indicate directory and file structures with respect to `pwd` (present working directory). Examples: `../cse278` or `../` or `../../courses/csex43/exercises` or `cse278/exercises` etc.

3. Let's protect our home directory to avoid potential eavesdropping, by restricting privileges via the following `chmod` (change mode) command below. We will study this `chmod` command in a bit more detail later on in this course.

```
$ chmod g-rwx,o+rx-w ~/ ↵
```

Now, briefly recollect the use of the first 8 commonly used GNU/Linux commands from page #3 in the `CommonMethodsAndCommands.pdf` file in the `Handouts` folder on Canvas. The first 8 commands in that document are: `exit`, `cd`, `pwd`, `ls`, `mkdir`, `rmdir`, `cp`, `scp`. By now you should have memorized these 8 commands from CSE-278, if not, you are missing important skills, not just for this course, but for your future career.

4. Briefly practice some of the Linux commands (from the table above) to perform the following tasks:
   a. Check to see your present working directory
   b. Change the PWD to your home directory (hint: `~/`)
   c. Create a directory named `cse381` in your home directory. Verify that the directory has been created by listing the files.
   d. Copy a file named `/usr/share/common-licenses/GPL` to the newly created `cse381` directory. Show the command that can you have used to confirm that you have successfully copied the file below:

I used the `ls` command to ensure that the file has been copied as in:

```
raodm@os1:~/courses/cse381_spring20$ ls -l
~/courses/cse381_spring20/
total 40
drwxr-xr-x 2 raodm uuidd  4096 Jan 25 19:48 exercises
-rw-r--r-- 1 raodm uuidd 35147 Jan 25 20:36 GPL
```

Provide brief responses to the following questions:

5. What is the server-name or hostname of the Linux server being used for this course?

The hostname of the Linux server is: `os1.csi.miamioh.edu`

6. Assume you would like to access the server when you are not on-campus. What extra software do you first need to run before trying to access the server while off-campus?

For off-campus access, Virtual Private Network (VPN) software needs to be run first. VPN also requires 2-factor authentication via Duo. VPN software can be downloaded for free from: http://miamioh.edu/vpn.

7. Your friend is running a program and it is generating the following error – "`Input file a.txt not found`". Your friend is perplexed. Searching the Internet did not seem to

yield useful solutions. What suggestions can you offer to your friend to help troubleshoot this issue?

1. First check to ensure you are on the correct Linux server using `hostname` command.
2. Use `pwd` command to ensure you are in the correct directory that you "think" you are in. If not, use `cd` command to first get to the correct working directory.
3. Use the `ls` command to list files in the directory and check to ensure `a.txt` is present.
4. If the file had to be copied over, check to ensure you have copied the file to the correct directory. Maybe you specified a wrong destination path in `scp` command.

If you are writing a program, step through the debugger to ensure you are passing the correct path to the specified file.

Part #2: Recap basics of HTTP protocol from CSE-278

Estimated time: 20 minutes

Background

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web (WWW). In HTTP a "client" (typically a Browser) send a request to a web-server requesting information, typically a file. If the request is valid, the server response with information. HTTP is a relatively straightforward multiline textual protocol. A key aspect of HTTP are headers, that have the following characteristics:

- HTTP headers consists of multiple lines. Each line is terminated with `"\r\n"` (aka CR-LF – Carriage-Return and Line-feed).
- Request and response have 1 or more HTTP headers starting with the second line.
- Each header line is in the format "`Name: Value`" (*i.e.*, "Name Colon Value" pair). The "name" typically are predefined values, such as: `"Host"`, `"Connection"`, etc.
- The headers are terminated by 1-blank line (i.e., just `"\r\n"`)

Exercise: In this part of the exercise you are expected to observe HTTP request and response headers from a browser. Several browsers (such as: Google `Chrome`) include "Developer tools" that permit you to view the actual HTTP request and response that the browser is used. It is useful to be able to view the headers for understanding the HTTP protocol and troubleshoot problems. **However, bear in mind every browser has a different way to show the same information.**

1. Launch Google `Chrome` (on your local machine) and open a new tab.
2. Open `Developer Tools` (opens in the bottom of browser window) using the following keyboard shortcut (you can use browser menus as well):
    1. Mac: Cmd+Opt+I
    2. Windows: Ctrl+Shift+I
3. Next switch to the `Network` tab in `Developer Tools`.

4. Now access the following URL (You may have to press Shift and the "↻ **Reload**" button if you have already accessed the URL): http://ceclnx01.cec.miamioh.edu/~raodm/exercise1_numbers.txt. You should see a HTTP request in the `Network` tab. Stop recording by clicking the stop button (●) at the top-left corner of the `Network` tab.
5. From the Network tab copy-paste the request headers (click view source) in the space below:

```
GET /~raodm/exercise1_numbers.txt HTTP/1.1
Host: ceclnx01.cec.miamioh.edu
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36
DNT: 1
Accept: text/html, ,application/xml;q=0.9,image/webp,image/apng,*/*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _ga=GA1.2.486049154.1559230408
If-None-Match: "21-59029c1df6131"
```

6. From the above request headers briefly respond to the following questions on how the URL has been converted to a GET request:
   1. What was the host name in this example? Where did the host name (i.e., server name) get placed in the HTTP request?

      The host name in this example was `ceclnx01.cec.miamioh.edu`. It is always placed as the value for the HTTP header named `Host`.
   2. What was the file that we requested? Where did the file we requested get placed?

      The file that we requested was `~raodm/exercise1_numbers.txt`. Path to this file is always placed in the 1st line after the word "GET"
   3. Now, given the URL `http://www.users.miamioh.edu/raodm/nums.txt`, convert it to a simple HTTP GET request by filling-in the blanks below: Connection (you will use similar operations in C++ code further below):

```
GET  /raodm/nums.txt  HTTP/1.1\r\n
Host: os1.csi.miamioh.edu\r\n
Connection: Close\r\n
```

7. Next, from the Network tab copy-paste just the response headers (click view source) in the space below:

```
HTTP/1.1 200 OK
Date: Mon, 19 Aug 2019 23:32:37 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Thu, 15 Aug 2019 15:45:04 GMT
ETag: "21-59029c1df6131"
Accept-Ranges: bytes
Content-Length: 33
Keep-Alive: timeout=5, max=94
Connection: Keep-Alive
Content-Type: text/plain
```

8. From the above response headers observe:
   1. The first line is an HTTP status code in the form "`HTTP/1.1 200 OK`".
   2. Note how the "`Server:`" header shows the name to be "Apache". Apache is one of the most popular web-servers developed in C/C++.
   3. You should also observe a "`Content-Type`" header with value "`text/plain`" indicating that the file is a plain text file.

Part #3: Setting up C++ projects in NetBeans

Estimated time: 15 minutes

Background: In this course it is highly recommended to use `NetBeans` 8.2 (using version 8.2 is important) for C++ programming. NetBeans is already installed on the lab computers. If-and-when you install it on your computer, you will need to configure it as shown in the `InstallNetBeansPlugin.pdf` in Canvas → `Files` → `Handouts` folder.

Exercise: Complete this part of the exercise via the following steps:

1. If you have not used `NetBeans` before, review the following video (< 8 minutes) demonstrating the process of creating, compiling, and running a C++ program in NetBeans: https://youtu.be/421zFJmrLkw (video available on Canvas as well).

2. Create a remote `NetBeans` project called **`exercise1`** on the Linux server **`os1.csi.miamioh.edu`**. Ensure you use `Miami University C++ Project to create` your `NetBeans` project. You don't need to create a main file (as you are given starter code for this exercise).

3. Double check you have created the project correctly.
   a. **Double check to ensure that the project is really on the server by ensuring that the server name is shown adjacent to the project name.**
   b. Note the directory where `NetBeans` created your remote project using your project's properties (`NetBeans` → Right-click-on-project → `Properties` → `General`) and ensure it is on the server. You will need this directory information in the next step.
4. Download the supplied starter files to your local computer and copy the files to your `NetBeans` project directory via the following steps:
   a. Note where the files are being downloaded to your local computer. Typically, they are stored in the `~/Downloads` folder.
   b. Start a Windows `PowerShell` terminal on your local computer. On a Mac you can start a local Mac `Terminal`.
   c. From the local terminal, secure-copy the starter files to your remote project via the following `scp` command. You should be familiar with `scp` from your CSE-278 course. If not, you should invest time to learn how to use it. It is one of the most important tools in the industry today. **Note**: Each one of the following commands are typed on 1 line. The commands appear wrapped onto more than 1 line merely because of page width.

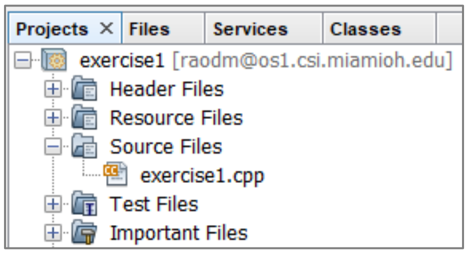
```
BEN006-Instr ~/ $ cd ~/Downloads
```

```
BEN006-Instr ~/ $ scp exercise1.cpp
MUID@os1.csi.miamioh.edu:~/NetBeansProjects/exercise1
```

Troubleshooting: Having issues with `scp`? Here are a few things to keep in mind:

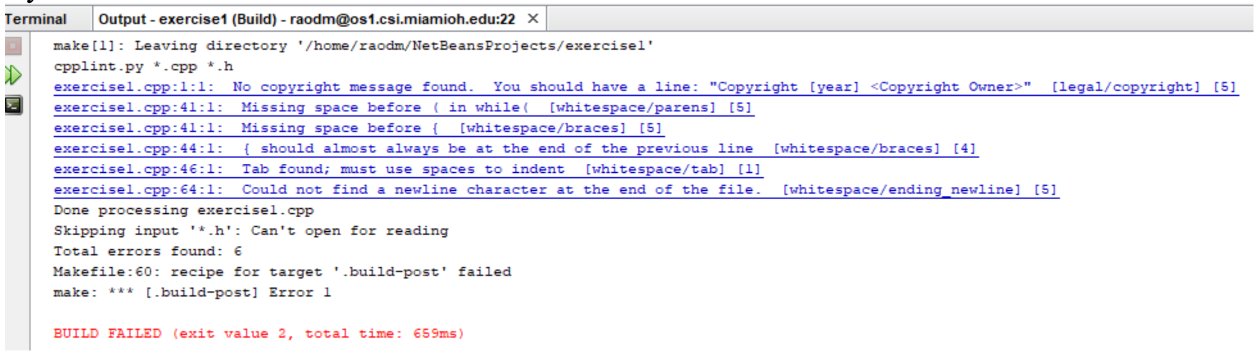
- **`scp` should be run on your local computer and not on the Linux server!**
- First form a mental model that you are working with 2 completely different computers.
- Remember syntax: `scp SourceFilePath LoginID@server.edu:DestFilePath`
- The files you are copying are in two different directories on the 2 computers. Use `ls` command to double check if files are present in the source directory from where you are trying to copy.
- Ensure you have your destination directory on the Linux server is correctly specified. Use the path from your `NetBeans` project settings.
- **Warning: Be cautious when copying files to ensure you accidentally don't overwrite files and lose your work. If you overwrite files, then you basically have to start over!**

5. Next, in your `NetBeans` project, add the files you have copied to the server to your project via: Right-mouse-click on `NetBeans` Project and use Add Existing Item… menu option to add the two files to your project. You may have to drag-n-drop `exercise1.cpp` into the `Source Files` category. After this step your project should appear as shown in the adjacent screenshot.



6. For future use remember these 4 steps: ❶ Create `NetBeans` Project using <u>Miami University C++ project</u> ❷ Download starter files to local computer, ❸ `scp` to server, ❹ Add files copied to Linux server to `NetBeans` project.

7. If you have setup your project, then the starter code should compile but will produce the style errors shown below:



```
make[1]: Leaving directory '/home/raodm/NetBeansProjects/exercise1'
cpplint.py *.cpp *.h
exercise1.cpp:1:1:  No copyright message found.  You should have a line: "Copyright [year] <Copyright Owner>"  [legal/copyright] [5]
exercise1.cpp:41:1:  Missing space before ( in while(  [whitespace/parens] [5]
exercise1.cpp:41:1:  Missing space before {  [whitespace/braces] [5]
exercise1.cpp:44:1:  { should almost always be at the end of the previous line  [whitespace/braces] [4]
exercise1.cpp:46:1:  Tab found; must use spaces to indent  [whitespace/tab] [1]
exercise1.cpp:64:1:  Could not find a newline character at the end of the file.  [whitespace/ending_newline] [5]
Done processing exercise1.cpp
Skipping input '*.h': Can't open for reading
Total errors found: 6
Makefile:60: recipe for target '.build-post' failed
make: *** [.build-post] Error 1

BUILD FAILED (exit value 2, total time: 659ms)
```

Part #3: Understanding and fixing style issues

Estimated time: 15 minutes

Background: In your jobs, you will be developing software for other programmers to use, and not just for yourself. Hence, you have to get into the habit of "*thinking about others*" – a key requirement being adhering to coding styles. So, in preparation for your careers, the CSE programming style guidelines will be strictly enforced in this course. The style guidelines are pretty simple (and the same rules apply for Java program as well!)

- First, keep in mind different people use different devices (smartphones, tablets, terminals, etc.). Hence, all technology companies (including Google, Facebook, Amazon, etc.) mandate the following style so that source code renders consistently on all devices:
    - Must use only spaces for indentation. Use 4-spaces. **Lucky for you, when you click the "Tab" key in `NetBeans` it actually uses 4-spaces**.
    - No line should be longer than 80-characters (see thin vertical red line in NetBeans).
    - None of the methods should be longer than 25-lines of code (does not include comments or blank lines). This is to ensure good program structure, ease readability, and understandability of methods. If you have methods longer than 25-lines of code, that is an indication that you are lacking structured thinking process. So, you have to get into the practice of developing good problem-solving skills.

  Again, keep in mind you are developing code for other people and you have to demonstrate to companies you can develop good, well-styled code to get good jobs.
- There must be spaces between words (just like in English) – so, "`if(a>b){`" is wrong style. Instead, it should be typed as "`if (a > b) {`" (note spaces between words).

Exercise: In this part of the exercise, you are expected to learn about style issues and fix style errors in the starter code.

1. read the style error messages in the output window and try to fix them. Each time you fix a line, compile your program and ensure you have fixed the problem correctly.

2. As part of the TODO, you are required to add documentation to the process method to explain in full detail what the lines of code are doing.

3. Once you have fixed the style issues and added documentation, you should be able to run the program and produce the following output:

```
raodm@os1:~/NetBeansProjects/exercise1$ ./exercise1
<html>
  <body>
    <h2>Analysis results</h2>
    <p>Number of values: 13</p>
    <p>Sum of numbers: 83</p>
    <p>Average value: 6.38462</p>
  </body>
</html>
```

4. Briefly state (in your own words) why we will be practicing "Programming style guidelines" in this course (*i.e.*, why can't you post ugly code on GitHub for your job interview and hope to make a good "first impression"?)

   We all get only one chance to make a "first impression". Program style is very important in jobs because we will be developing programs for others to use. Good style makes it easy for others to read and understand our program. So, employers pick people who can consistently demonstrate good programming style – it is not enough that I claim I can do good style (I can claim anything I want) but proving I can do good style is what is important and that comes through practice. So, in this course I am glad we will be practicing good style.

Part #4: Debugging a program

Estimated time: 15 minutes

Background: The debugger is a very important tool that you should learn how to use. It will come in handy in future laboratory exercises and homework. Consequently, you should gain familiarity with using the debugger.

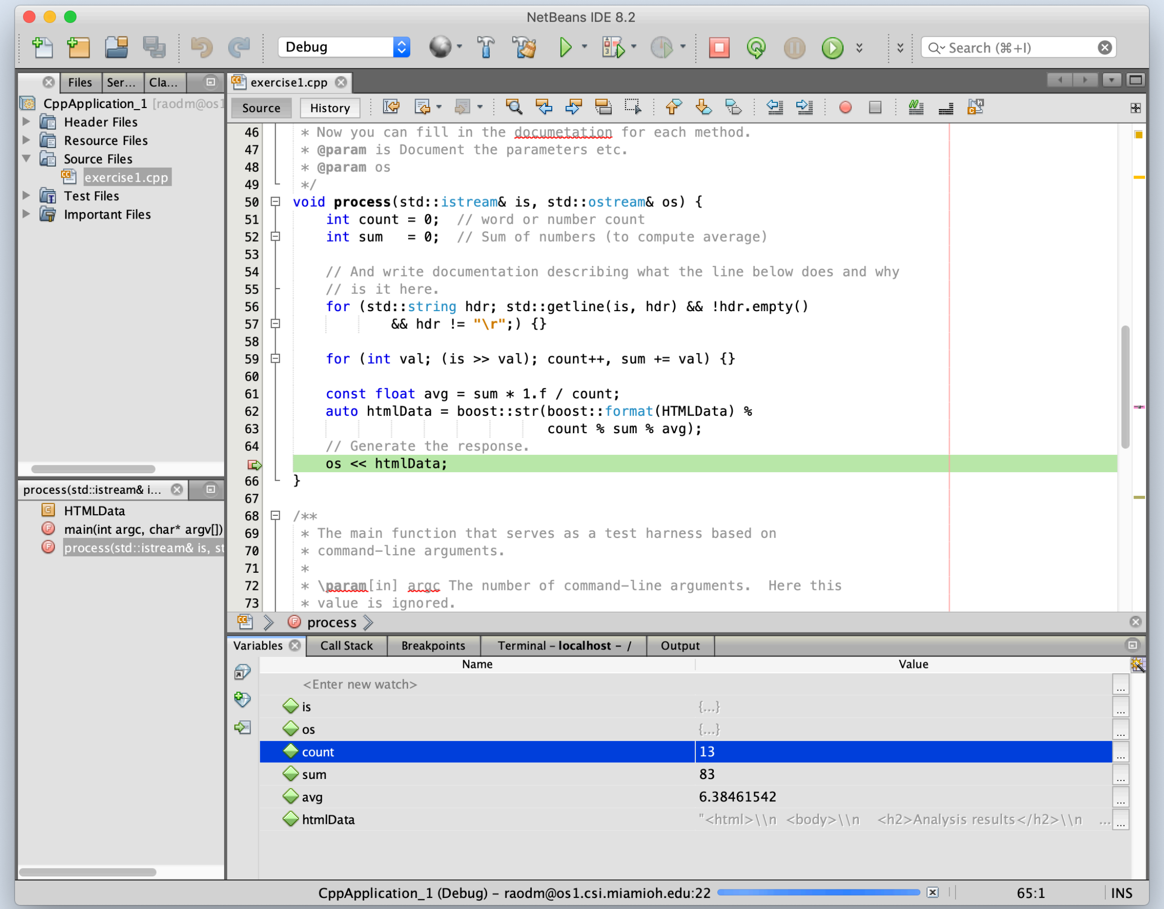
Exercise: Complete this part of the exercise using the following procedure:

1. If you have not used the debugger in NetBeans, first review its usage via the following video: https://youtu.be/O7UYNut1oks (video also available on Canvas)
2. In order to use the debugger, you first need to set a breakpoint. Set a breakpoint in `exercise1.cpp` , in the `process` method, at the line "`os << htmlData;`".
3. Run the program via the debugger.
4. Your friend is trying to use the debugger. However, your friend is getting the warning – `&"warning: GDB: Failed to set controlling terminal: Operation not permitted\n"`. Your friend is very confused. What should your friend do?

   Obviously, my friend did not watch the videos posted by the instructor. I would point my friend to the spot in the video where the instructor says to "ignore this warning message" as it is a spurious warning.

5. Observe how the call stack looks.
6. Observe the local variables. Try to look at the details of the different variables.
7. Finally, make a screenshot of your whole `NetBeans` window showing the following:
   a. The current line of code
   b. The stack trace tab should be visible
   c. The variables tab should be visible

Place screenshot of debugger showing all of the above 3 pieces of information here



```
 * Now you can fill in the documetation for each method.
 * @param is Document the parameters etc.
 * @param os
 */
void process(std::istream& is, std::ostream& os) {
    int count = 0;  // word or number count
    int sum   = 0;  // Sum of numbers (to compute average)

    // And write documentation describing what the line below does and why
    // is it here.
    for (std::string hdr; std::getline(is, hdr) && !hdr.empty()
            && hdr != "\r";) {}

    for (int val; (is >> val); count++, sum += val) {}

    const float avg = sum * 1.f / count;
    auto htmlData = boost::str(boost::format(HTMLData) %
                               count % sum % avg);
    // Generate the response.
    os << htmlData;
}

/**
 * The main function that serves as a test harness based on
 * command-line arguments.
 *
 * \param[in] argc The number of command-line arguments.  Here this
 * value is ignored.
```

Name	Value
<Enter new watch>	
is	{...}
os	{...}
count	13
sum	83
avg	6.38461542
htmlData	"<html>\\n <body>\\n <h2>Analysis results</h2>\\n ...

CppApplication_1 (Debug) – raodm@os1.csi.miamioh.edu:22   65:1   INS

Part #5: Review Java → C++ translation guide

Estimated time: 10 minutes

Background: Many of you should be comfortable with programming in Java. C++ and Java are syntactically very close. Importantly, the logic (or approach to problem-solving) in the two languages is the same! However, the API (i.e., classes and methods) are different. The biggest difference between Java & C++ arises in input-output (I/O) operations. However, with practice C++ becomes as easy as Java.

Exercise: In order to help you transition between Java and C++, there is a handy Java → C++ translation guide on Canvas. Briefly review this guide via the following steps:

1. View the translation guide on Canvas→ Files → Handouts → JavaCppTranslationGuide.pdf.
2. Now, let's use the translation guide to understand the `substr` method in C++. In C++, sub-string method `substr(int index, int len)` the 2nd parameter is the number of characters to return (**because that is how the CPU works**). Ensure you understand

substring by completing the table – for each row, fill-in either a call to `substr` to obtain the desired output or indicate output from the `substr` method call.

```
std::string str = "0123456789";
```

Call to substr method in C++	Desired/Expected output
`str.substr(3, 4)`	"3456"
`str.substr(4, 1)`	"4"
`str.substr(8, 2)`	"89"
`str.substr(7)`	"789"

3. Now, to really ensure we understand `substr` in C++, let's write a 1-liner method in C++ that is equivalent to Java's substring method that takes 2 index positions.

```
/**
 * See the Javadoc for Java.lang.String.substring method to
 * implement this method to return a substring from the given
 * string str.
 */
std::string substring(const std::string& str,
                      size_t beginIndex, size_t endIndex) {
    // Write the 1-line C++ code to return substring similar to
    // how Java would return.

    return str.substr(beginIndex, endIndex - beginIndex);
}
```

Part #7: Submit to Canvas via CODE plug-in

Estimated time: 5 minutes

Background

Most (if not all) programming homework/project are required to be submitted via the CSE department's CODE plug-in. The plug-in has been designed to facilitate learning and teaching in programming centric courses such as this one. The CODE plug-in is setup to ensure: (1) your program compiles, (2) passes style checks, and (3) correctly operates for test cases. This eliminates issues with accidental incorrect submissions, incorrect solutions, etc. while promoting higher quality learning and outcomes for all.

Exercise

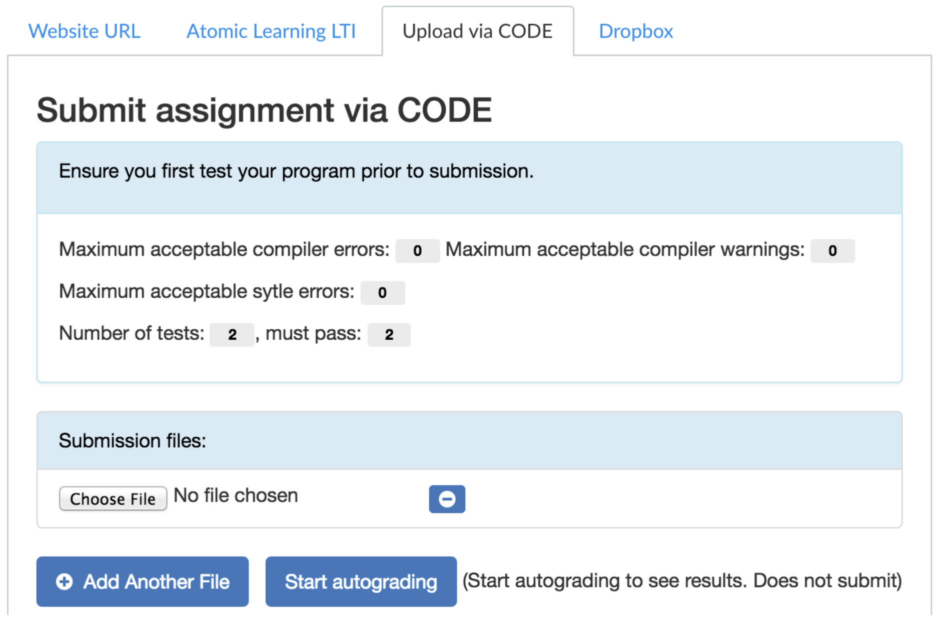
In this part of the exercise, you will be submitting the necessary files via the Canvas CODE plug-in.

1. If this is your first time using the CODE plug-in then review the submission process via the following brief video demonstration -- https://youtu.be/P2bWUt5KqbU.
2. Save this MS-Word document as a PDF file – **Do not upload Word documents. Only submit PDF file**.

3. Use `scp` from a `PowerShell` terminal to copy `exercise1.cpp` from the Linux server to your local machine. For convenience here is an example `scp` command that you can adapt for your use:

```
scp        muid@os1.csi.miamioh.edu:~/NetBeansProjects/exercise1/exercise1.cpp
~/Downloads
```

4. Practice submitting solutions via the CODE plug-in by submitting the following files:
   a. The `exercise1.cpp` starter code
   b. Your lab notes for this exercise saved as a PDF file.
5. Upload the files using the "`Upload via CODE`" tab shown in the screenshot below:



Submit assignment via CODE

Ensure you first test your program prior to submission.

Maximum acceptable compiler errors: 0 Maximum acceptable compiler warnings: 0

Maximum acceptable sytle errors: 0

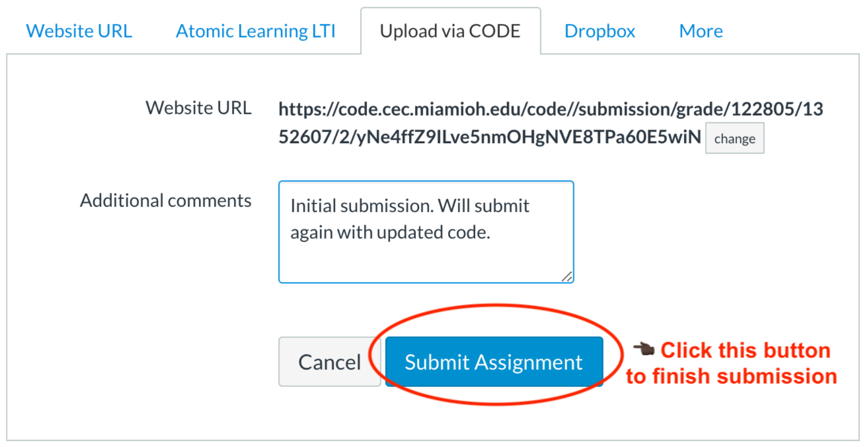
Number of tests: 2 , must pass: 2

Submission files:

No file chosen

(Start autograding to see results. Does not submit)

Ensure you actually **submit** the URL generated by CODE plug-in in the final step as shown in the Video demonstration and in the screenshot below:



Website URL https://code.cec.miamioh.edu/code//submission/grade/122805/1352607/2/yNe4ffZ9ILve5nmOHgNVE8TPa60E5wiN

Additional comments: Initial submission. Will submit again with updated code.