

ENGG1340 Computer Programming II
COMP2113 Programming Technologies
Module 2 Checkpoint Exercise

Name: XXXXXXXXXX
University ID: XXXXXXXXXX

Instructions:

For each single question or each group of questions in the Checkpoint exercise, please type your answer right after the question in this Word document. Please refer to the example below.

Checkpoint 0:

What is the meaning of the command “date”?

Ans: The “date” command prints the current date of the current machine

Checkpoint 2.1 (Please answer in this document and submit to Moodle)

What is the output of the following shell script? Please explain your answer.

```
#!/bin/bash

a=100
b=99
if [ $a \> $b ]
then
    echo "$a is larger"
else
    echo "$b is larger"
fi

if [ $a -gt $b ]
then
    echo "$a is larger"
else
    echo "$b is larger"
fi
```

Note: If you are using your own Linux, you may need to grant execute permission to run it.

Solution

99 is larger

100 is larger

We get this output because 100 is sorted before 99 so `$a /> $b` evaluates to **False**, hence the commands in the else clause are executed (1 is sorted before a 9). Whereas the `-gt` command compares the numeric magnitude of `$a` and `$b`, and since 100 is larger in magnitude than 99 we get the result in the second line of the output.

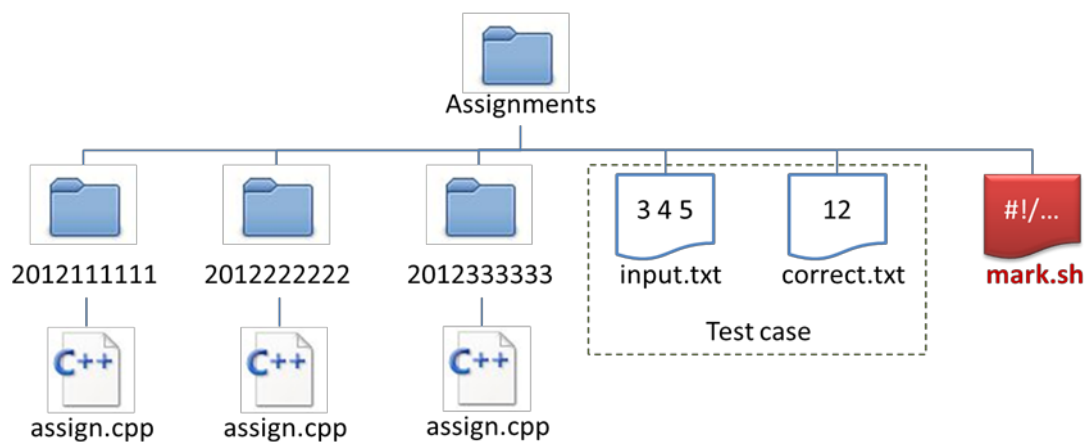
Checkpoint 2.2: Shell Script to mark all assignments (Please submit and evaluate your answer in VPL)

Let's upload the module2 files to the Linux server and browse to `Assignments/`

```
$ cd Module2/  
$ cd Assignments/  
$ ls  
2012111111  2012222222  2012333333  input.txt  correct.txt  mark.sh
```

Let's write a script to mark all the assignments in a class!

1. Suppose that the submission of each student is organized in the directory `Assignments/UID`, where UID is the University number of the student.
2. In this example, we have three students with UID 2012111111, 2012222222 and 2012333333.
3. The test cases `input.txt` and `correct.txt` are located in the Assignments directory.
4. The shell script `mark.sh` is under the Assignments directory, to be updated by you.
5. Evaluate your answer on Moodle. Test case files and student submissions are included in the evaluation environment. You only need to submit your updated `mark.sh`.
6. When locating a file or directory in `mark.sh`, use the relative path instead of the full path.



Consider the unfinished shell script *mark.sh*

```
#!/bin/bash
allStudents=Q1
for student in $allStudents
do
    if Q2
    then
        cd $student
        if Q3
        then
            g++ assign.cpp -o assign.o 2>error.txt
            if Q4
            then
                ./assign.o < ../input.txt > output.txt
                result=`diff output.txt ../correct.txt`
                if Q5
                then
                    echo "$student: Wrong answer."
                else
                    echo "$student: Correct!"
                fi
            else
                echo "$student: Cannot be compiled."
            fi
        fi
        cd ..
    fi
done
```

Question1:

- What should be the code in **Q1**? (Reminder, spacing is critical!)
- Hints: First we need to get the names of the files and directories under our current directory, what is the command to do this? How we can execute shell command in shell script and store the result in the variable `allStudents`?

Question 2:

- However, the command in **Q1** also returns files under the Assignments directory, such as the test case files *input.txt* and *correct.txt*, as well as the shell script *mark.sh*.
- We need an if statement, what should be the condition in **Q2**?
- Hints: How can we check if `$student` is a directory?

Question 3:

- After we have identified the student's directories and browse into them one by one using the `cd $student` command, we need to check if the student has submitted *assign.cpp* or not.
- What is the condition to be filled in **Q3**?

Question 4:

- If the student has submitted *assign.cpp*, we then need to compile the *assign.cpp*, and report on screen if it cannot be compiled.
- We need to check if *assign.cpp* can be compiled or not.
- What is the condition to be filled in **Q4**?
- Hints: if the compilation is successful, what file will be created after executing the line `g++ assign.cpp -o assign.o 2> error.txt`?

Question 5:

- If the student has submitted *assign.cpp*, and *assign.cpp* can be compiled, we then need to check if the output file *output.txt* is the same as the test case *correct.txt* or not.
- The `result=`diff output.txt ../correct.txt`` command will store some value in the variable `$result` if *output.txt* is different from *correct.txt*.
- Therefore, if the length of the string `$result` is NOT EQUAL to 0, then there is something output by the `diff` command, that means *assign.o* returns wrong answer.
- What is the condition to be filled in **Q5**?

Upload your *mark.sh* to the Moodle page “**Checkpoint 2.2: Shell Script to mark all assignments**” under Module 2 and **evaluate** it.

You can refer to “**Quick VPL Guide**” on Moodle if you don't know how to use the evaluation system.

Checkpoint 2.3: Shell Script to analyze web log (Please submit and evaluate your answer in VPL)

Suppose the web server of your company website that produces a log file which captures every page access in the following format:

ip_address date_YYYY/mm/dd time_HH:MM:SS page_accessed

Here is a sample content of such file:

192.168.83.173	2014-09-04	10:05:17	/products.html
192.168.130.175	2014-09-04	10:09:13	/products/296.html
192.168.130.175	2014-09-04	10:25:01	/index.html
192.168.138.244	2014-09-04	10:29:37	/products.html
...			

You are asked to write a shell script, **getProductAccess.sh**, which captures the product pages under **/products/** accessed by each IP address. The script takes one command line argument, which is the filename of a log file. It will read the content of the log file specified, find and print the product pages each IP address accessed. The output should be sorted by IP address and then by the product page.

- Sorting is done **alphabetically, ascendingly** for both IP address and product page, i.e., 192.168.0.11 will be ordered before 192.168.0.2.
- Only pages starting with **/products/** should be printed.
- You can assume that the input file will strictly follow the above format. I.e.:
 - Each page access is recorded on a separated line.
 - Each value (*ip_address*, *date*, *time* and *page_accessed*) are separated by tabs.
 - The values will not consist of spaces or tabs.
- The script does not need to handle cases where the input file is empty, or if there is not enough argument specified.

Sample Run

Assume we have *access-2014-09-04.log* in the current directory, with the following content:

192.168.83.173	2014-09-04	10:05:17	/products.html
192.168.130.175	2014-09-04	10:09:13	/products/296.html
192.168.130.175	2014-09-04	10:25:01	/index.html
192.168.138.244	2014-09-04	10:29:37	/products.html
192.168.83.173	2014-09-04	11:00:00	/products/004.html
192.168.130.175	2014-09-04	11:01:20	/products/296.html
192.168.138.244	2014-09-04	11:22:49	/products.html
192.168.138.244	2014-09-04	11:23:00	/index.html
192.168.83.173	2014-09-04	12:19:55	/products/560.html
192.168.130.175	2014-09-04	12:24:24	/products/004.html

If we run the script like this:

```
$ ./getProductAccess.sh access-2014-09-04.log
```

The script will print:

```
192.168.130.175    /products/004.html
192.168.130.175    /products/296.html
192.168.83.173     /products/004.html
192.168.83.173     /products/560.html
```

Note that IP 192.168.130.175 accessed /products/296.html multiple times, but there should only be one corresponding line in the output.

Hints:

- You may need to use the commands “cut”, “grep”, “sort” and/or “uniq” in your script.

Upload your answer to the Moodle page “**Checkpoint 2.3: Shell Script to analyze web**” under Module 2 and **evaluate** it.

Checkpoint 2.4: Working with Git and GitHub

GitHub is an online service where you can store your projects and share your code with others easily. Later, we plan to request students commit their code and changes to GitHub mandatorily for the course project. Study the steps below to learn how to work with your local git repository with GitHub.

Create a GitHub account

1. Go to <https://github.com/> to create a personal user account.

Create a personal access token

You should create a personal access token (PAT) to use in place of a password with the command line. When Git prompts you for your password, enter your personal access token instead of your password.

For example, you can do something like this with your token when working with GitHub:

```
$ git clone https://github.com/username/repo
Username: your_username
Password: your_token
```

To create a personal access token, go to <https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token> and follow its procedures.

Note:

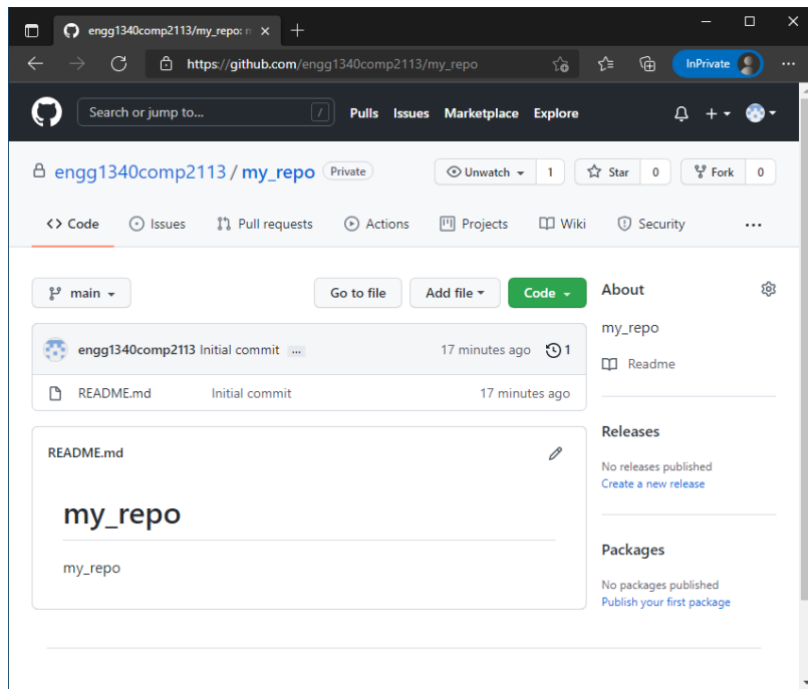
- In step 6, give your token a name whatever you like.
- In step 8, select all scopes and permissions if you are not sure what they are. (Less security but more convenience)
- Treat your token as like password and keep it secret!

Create a new repository on GitHub

Go to <https://docs.github.com/en/get-started/quickstart/create-a-repo#create-a-repository> and follow its procedure to create a **repository** in “GitHub.com”.

After the creation of a new repository, your browser will be directed to the main page of your new repository, which the URL has the formal of *https://github.com/<username>/<repo>*, where *<username>* is the name of your GitHub account and *<repo>* is the name of your new repository.

The screenshot below shows the main page of a new repository (Initialized with adding a README.md)



Push your files to GitHub

There are two main ways to connect your local git repository to GitHub.

Option 1: If you are starting from scratch and do not have a local repo, clone your GitHub repository to your local computer from the command line.

```
$ git clone https://github.com/username/repo
$ cd repo
```

- Replace the URL and directory name with your repository URL and name.

Then work on the project and finally push it to GitHub

```
$ git add .
$ git commit -m "first commit"
$ git push
```

Option 2: If you have already created a repository locally, push your existing repository from the command line:

```
$ git remote add origin https://github.com/username/repo
$ git branch -M main
$ git push -u origin main
```

- Replace the URL with your repository URL.

Upon success, you will see your project files on GitHub.

Note: There is no need to submit anything for this checkpoint.

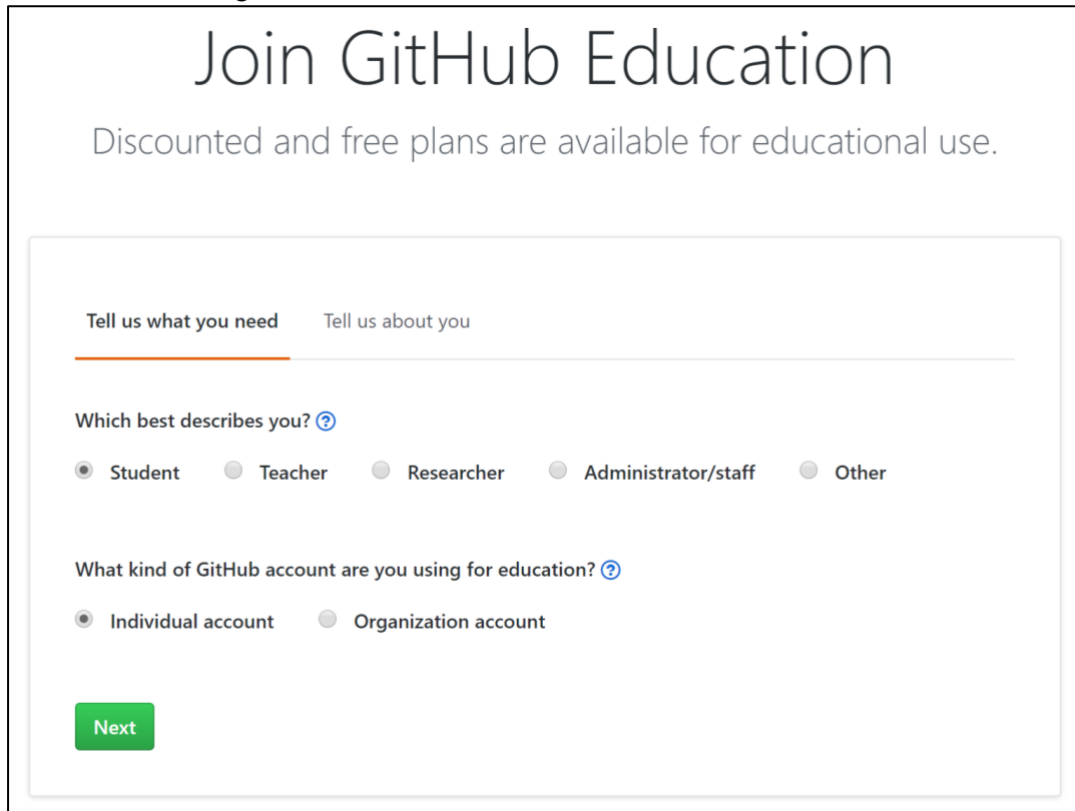
Checkpoint 2.5: GitHub Education

GitHub has an education plan to allow university students to upgrade their account to GitHub Pro for free while they are a student. All students can join it at <https://education.github.com/>.

To join GitHub Education, you must have a personal GitHub account at <https://github.com/> first.

Here are some steps to join the GitHub Education (For reference only):

1. Sign in to https://education.github.com and click “Get benefits”. In the application form, specify you are a **student** and creating an **individual account**.

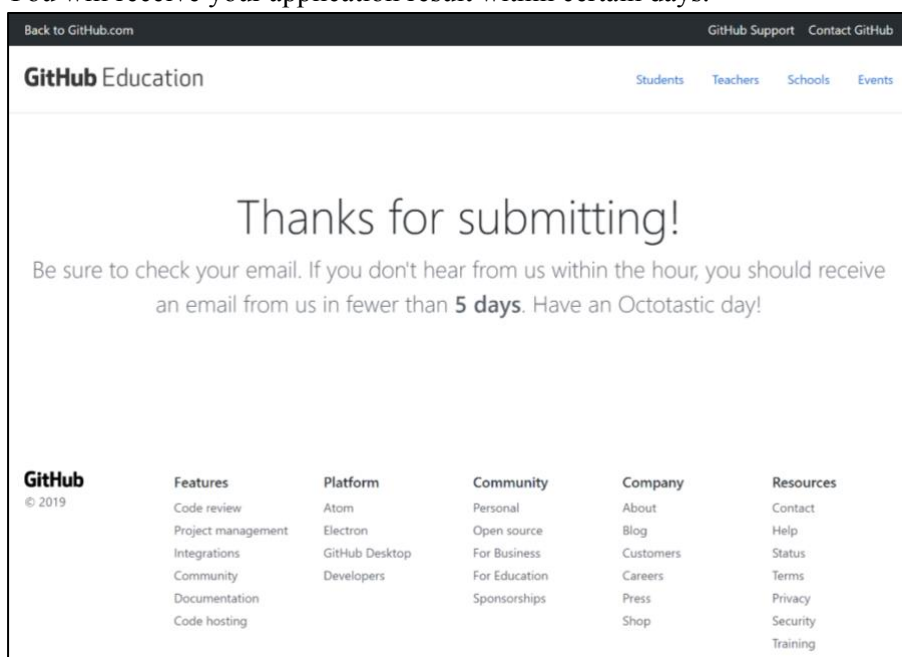


The screenshot shows the 'Join GitHub Education' page. At the top, it says 'Join GitHub Education' and 'Discounted and free plans are available for educational use.' Below this is a form with two tabs: 'Tell us what you need' (active) and 'Tell us about you'. The first question is 'Which best describes you?' with five radio button options: Student (selected), Teacher, Researcher, Administrator/staff, and Other. The second question is 'What kind of GitHub account are you using for education?' with two radio button options: Individual account (selected) and Organization account. At the bottom left of the form is a green 'Next' button.

2. Add your HKU email address and then click “Add”. A verification email will be sent to your HKU mailbox. Click the link in the email to verify your identity. Then click “Submit request”

The screenshot shows the GitHub Education application form. At the top, there are two tabs: "Tell us what you need" and "Tell us about you", with the latter being selected. Below the tabs, there is a "Name" field with a placeholder "[your name]". Underneath, the question "What e-mail address do you use for school?" is followed by a "Pro-tip" indicating that a school-issued email address is preferred. There are two email input fields: the first contains a placeholder email "@hotmail.com", and the second contains a placeholder "@hku.hk" with a green "Add" button next to it. Below these fields is a button labeled "+ Add an email address". Further down, the question "How do you plan to use GitHub?" is followed by a text input field containing "To finish my coursework.". At the bottom, a note states: "Please note, your request cannot be edited once it has been submitted, so please verify your details for accuracy before sending them to us." Below this note is a green "Submit request" button.

3. You will receive your application result within certain days.



Note: There is no need to submit anything for this checkpoint.