ENGG1340 Computer Programming II
COMP2113 Programming Technologies
Module 1 Checkpoint Exercise

Name: Shaheer Ziya
University ID: 3035946760

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 1.1

Now, let's try to answer the following questions. Although you haven't been taught the meaning of the following commands, you can display the manual page of these commands and learn their meanings by yourself

1. Why do we need to learn command line although we can use a GUI to control a computer?
2. What is the meaning of `ls -t`?
3. What is the meaning of the command **`pwd`**?
4. What is the meaning of the command **`rm`**?
5. What is the meaning of the command **`mv`**?
6. Suppose that the **fileA** does not exist in your present working directory, what is the meaning of the command `touch fileA`?
7. What is the meaning of the command **`tar`**?
8. What is the command for creating an archive **files.tar** from two files named **fileA** and **fileB**?

Ans:

1. Fluency with the bash/CLI is absolutely crucial for working with systems that do not have the processing power or even the hardware to display graphical interfaces. For example, in micro-controllers inside robotic/mechatronic devices.
   Additionally, learning the command line "will" absolutely make dealing with data and text much easier. (Or so I have heard)

2. The command `ls -t` lists the contents of the current directory (`./`) and sorts them by modification time, with the most recently edited file first.

```
-t      sort by modification time, newest first
```

```
h3594676@academy11:~$ ls -l -t
total 4
drwxr-xr-x 3 h3594676 ext 3 Jan 25 20:05 Documents
drwxr-xr-x 2 h3594676 ext 6 Jan 24 22:40 Music
drwxr-xr-x 2 h3594676 ext 2 Jan 23 20:49 Downloads
drwxr-xr-x 2 h3594676 ext 2 Jan 20 14:06 Pictures
drwxr-xr-x 2 h3594676 ext 2 Jan 20 14:06 Public
drwxr-xr-x 2 h3594676 ext 2 Jan 20 14:06 Templates
drwxr-xr-x 2 h3594676 ext 2 Jan 20 14:06 Videos
drwxr-xr-x 2 h3594676 ext 2 Jan 20 14:06 Desktop
```

3. The command `pwd` stands for print working directory. Using this command in the shell prints out the path of the directory the user currently is in (as can be seen in the figure below)

```
h3594676@academy11:~$ pwd
/student/21/ext/h3594676
```

4. The command `rm` stands for remove and is used to remove objects in the shell. `rm "filename.ext"` deletes the file (if the file exists and is in the current directory). Adding the path of the file instead of just the filename accomplishes the same task (assuming the file exists and also if the user has write permissions I guess). To remove non-empty directories, we can use the command `rm -rf` where the `-r` flag recursively goes down the subdirectories and deletes everything while the `-f` flag ensures we aren't bombarded with confirmation messages everytime the shell wants to delete a file. On the other hand, to remove empty directories we can make use of the `rmdir "directory_name"` command.
5. The `mv` command stands for move. We can use it to move files between directories and can also use it to rename files/directories. To my understanding the command works as follows:

   Let the command be `mv "target" "destination"`

   Where target is a file/directory and destination is a file/directory or a path to one. If the destination exists and is a directory then the target will be moved inside this directory. If the destination doesn't exist then the object will be renamed to destination. (That's the gist of it at least)
6. Technically the `touch` command is used to update the access/modification date of a file, however, it can be used to create new files if files of the same name don't exist. So, using `touch fileA` will create a new file named "fileA" in the working directory.
7. The `tar` command stands for tape archive and is used to archive a collection of files and directories into a highly compressed archive called tar.
8. `tar -cf "archive_name.tar" "/path of fileA" "/path of fileB"`

The `-c` flag is used to create the archive, while the `-f` flag is used to give a name to the tar file. The remaining parts are the paths to fileA and fileB or simply their names if they are in the current working directory.

Checkpoint 1.2a

Assume we have logged in Ubuntu and started a bash shell. The current directory is the home directory, i.e., ~ . We want to perform the following tasks sequentially. For each of the tasks below, to accomplish it.

1. Create a new subdirectory "***assignments***" under ~.
2. Create a new subdirectory "***assignment 1***" under "***assignments***". (Note that we are creating one subdirectory "***assignment 1***" but not two subdirectories "assignment" and "1")
3. Remove the directory "***assignments***" and all its subdirectories.

Ans:

1. `mkdir assignments`
2. `cd assignments`
   `mkdir "assignment 1"`
3. `rm -rf assignments`

Checkpoint 1.2b

[Self-learning question] - You need to search for the information on the Internet to answer this question.

There is another way to modify the permission, which is called the Absolute mode.

a. Explain the meaning of `chmod 666 hello.txt`
b. Explain the meaning of `chmod 700 hello.txt`
c. What is the `chmod` command, in absolute mode, to set the following permission for *hello.txt*?

User permissions			**Group Permission**			**Other permission**		
r	w	x	-	w	-	r	-	-

d. The administrator says that "One does not simply 777 their entire server", explain what the problem is if we `chmod 777` for all the files.

Ans:

a) `chmod 666 hello.txt` enables the owner, group and others to read and write on the hello.txt file but not the execute permissions for that file. 6 (octal) is 110(binary). The format used by the binary number is rwx. That is read, write and execute permission respectively, where 1=permitted, 0=not permitted

b) This command gives the user read, write and execute permissions while all others (the group and others) are given no permissions.

c) Continuing with the explanation from (b), we want 111_2 for the user, 010_2 for the group and 100_2 for others, which converted to octal is simply 7_8, 2_8, 4_8. Thus the command we want is `chmod 724`

d) If we `chmod 777` all the files on our server, that is equivalent to allowing all users (owners, group members & others) to read, **write**, and execute all files present on the server. Which is an extremely bad if we have some naughty students on the server who think overwriting a year's worth of source code for some important project with quotes from Monty Python is funny :)

Checkpoint 1.3

Now you may have a doubt: I understand how **`diff`** works, but why is the output claimed to be the difference between the two files?

Consider the two files below:

`$cat question1A`	`$cat question1B`
`Apple`	`Boy`
`Boy`	`Cat`
`Cat`	`Egg`
`Dog`	
`Egg`	

Note that file **question1B** is created by removing "Apple" and "Dog" from the file **question1A**.

A. What will be the output if we execute the following command (Please try to think about the output before trying it in the shell)? Please explain your answer.

```
$diff question1A question1B
```

B. What will be the output if we execute the following command (Please try to think about the output before trying it in the shell)? Please explain your answer.

```
$diff question1B question1A
```

Ans:

a) To convert question1A into question1B we need delete the first line and fourth line of question1A

So, the output should be:

```
1d0
```

```
< Apple
```

```
4d2
```

```
< Dog
```

1d0 signifies that after deleting line 1 of 1A we the fill will be in sync starting from line 0 of 1B (where the line to be deleted read Apple). While 4d2 means that after deleting the fourth line of 1A (which reads Dog) the file will be sync with 1B starting from its second line.

b) This time we want to make question1B into question1A we need to add the first line of 1A to before the first line of 1B (which is line 0), so we will get:

```
0a1
```

```
> Apple
```

Then we want to add the line Dog (4th line in 1A) after the second line in 1B to make it like in 1A, so we have:

```
2a4
```

```
> Dog
```

Checkpoint 1.4

This is a challenging exercise! You need to understand the shell commands and the techniques introduced in the previous sections to work on this task.

The following C++ program *gen4.cpp* reads in a 4-character string from the input and generates all possible permutations from the 4 characters.

```
//gen4.cpp
#include <iostream>
#include <string>
int main() {
    std::string s;
    std::cin >> s;
    for (int i = 0; i < s.length(); i++) {
        for (int j = 0; j < s.length(); j++) {
                for (int k = 0; k < s.length(); k++) {
                        for (int l = 0; l < s.length(); l++) {
                                if (i != j && i != k && i != l && j != l && j != k && k != l) {
                                        std::cout << s[i] << s[j] << s[k] << s[l] << std::endl;
                                }
                        }
                }
        }
    }
    return 0;
}
```

To compile *gen4.cpp*

```
$ g++ gen4.cpp -o gen4
```

The input of the program should be stored in the file *gen4_input.txt* with the following content.

```
lopo
```

gen4_input.txt

1. Give **ONE** command (one line of command(s)) to run the *gen4* with *gen4_input.txt* as input and redirect the result to a file named *gen4_output.txt*.

   Hints:

   ```
   $ [your_command]
   $ cat gen4_output.txt
   lopo
   loop
   lpoo
   lpoo
   …
   $ wc gen4_output.txt
   24 24 120 gen4_output.txt
   ```

   *The output file should contain all permutations of the letters ‘l’, ‘o’, ’p’, and ‘o’. There should be 24 permutations in total.

   Ans: `$ ./gen4 < gen4_input.txt > gen4_output.txt`

2. Give **ONE** command to sort the words in *gen4_output.txt* in alphabetical order, and then also remove the adjacent duplicate lines and finally store the result in a file named *sort_uniq.txt*.

   Hints: Consider the command **`uniq`**

   ```
   $ [your command]
   $ cat sort_uniq.txt.
   loop
   lopo
   lpoo
   olop
   olpo
   oolp
   oopl
   oplo
   opol
   ploo
   polo
   pool
   ```

```
$ wc sort_uniq.txt
12 12 60 sort_uniq.txt
```
*There should be 12 unique words total.

Ans: `sort gen4_output.txt | uniq > sort_uniq.txt`

3. Give **ONE** command to check the spelling in *sort_uniq.txt* and store the misspelled words into another file named *misspell.txt*.

   Ans: `spell sort_uniq.txt > misspell.txt`

4. Now *sort_uniq.txt* contains all distinct generated words, and *misspell.txt* contains all misspelled words. The differences between the two files are the meaningful 4-character words. Give **ONE** command to return the correctly spelled words as shown below:

```
$ [your command]
< loop
< polo
< pool
```
Hints: Consider the command **`diff`** and **`grep`**.

Ans: `diff sort_uniq.txt misspell.txt | grep “p”`

Checkpoint 1.5

Consider the file *question1.txt*.

```
2011111111,John,M,98
2011222222,Marry,F,85
2011333333,Sally,F,85
2012111111,Kit,M,86
2012222222,Ben,M,97
2012333333,Smitty,F,92
2012444444,Jolly,F,93
2012555555,Ken,M,100
```

Figure 1 question1.txt

1. Give ONE command to return the lines that contain the record of Kit

   Hints:

```
$ [Your command]
```

```
2012111111,Kit,M,86
```

Ans: `grep "Kit" question1.txt`

2. Give ONE command to find the students with UID begin with "**2012**" (i.e., To find the lines that begin with 2012)

   Hints:

```
$ [Your command]
2012111111,Kit,M,86
2012222222,Ben,M,97
2012333333,Smitty,F,92
2012444444,Jolly,F,93
2012555555,Ken,M,100
```

   Ans: `grep "^2012" question1.txt`

3. Give ONE command to return the lines that contain the record of the students who are both:
   - UID start at **2012**, and
   - Name starts with the characters **J** or **S**

   Hints:

```
$ [Your command]
2012333333,Smitty,F,92
2012444444,Jolly,F,93
```

   Ans: `grep "^2012.*[JS]" question1.txt`