Computer Science 352 Summer 2023
Programming Assignment 9
Due 8/4/2023 by 7pm

This last PA for CS 352 has two parts. Each part will require you to write a bash-based script with a .sh extension. For the project, you should end up with the following directory structure:

```
pa9
├── cp.sh
└── sm.sh
```

Part 1 - Copy Permissions (cp.sh)

In part 1, you should write a bash script named cp.sh. The purpose of this script is to take the permission setting of one file, and replicate those same settings to every file within a specified directory. For example, say that our current working directory (CWD) is:

```
/home/yourname/
```

Within this directory, we have a file named **from.txt**. **from.txt** shows the following when displayed with stat:

```
$ stat --format="%A %n" from.txt
-rwxr----x from.txt
```

Now, say we have the **/home/yourname/other/** that has three files within. The permissions and names of these file are:

```
$ stat --format="%A %n"  /home/yourname/other/*
-r-xrwx--x /home/yourname/other/a.out
-rwxr--r-- /home/yourname/other/sample.csv
---x--xr-- /home/yourname/other/something.java
```

We can use cp.sh to make all of the files from the **other** directory have the same permission settings as **from.txt**. If we run:

```
$ cp.sh /home/yourname/other/ ./from.txt
Updated permissions for 3 file(s).
0 file(s) unchanged.
```

Then after this, if we re-run the stat command, we should see:

```
$ stat --format="%A %n"  /home/yourname/other/*
-rwxr----x /home/yourname/other/a.out
-rwxr----x /home/yourname/other/sample.csv
-rwxr----x /home/yourname/other/something.java
```

Notice how the permissions changed. Also notice how the script reports how many files it changed the permissions of, and how many it did not. This is a requirement for the script. If before running **cp.sh** the **other** directory had files with these permissions instead:

```
$ stat --format="%A %n"  /home/yourname/other/*
-r-xrwx--x /home/yourname/other/a.out
-rwxr----x /home/yourname/other/sample.csv
-rwxr----x /home/yourname/other/something.java
```

Then running that same **cp.sh** command should print out the following instead:

```
$ cp.sh /home/yourname/other/ ./from.txt
Updated permissions for 1 file(s).
2 file(s) unchanged.
```

This is because 2 out of the three files already had the same permission settings as **from.txt**. You can assume that the program will always be provided with two valid command-line arguments.

Part 2 - Search Manual (sm.sh)

In part 2, you should write a bash script named sm.sh. The purpose of this script is to search the man pages for a provided set of bash commands, and find the one that best "matches" for a particular keyword. The idea behind this script is to be used as a tool to help a developer find the appropriate tool to use when working with a UNIX system.

The mechanism of the script is actually quite simple. The first command-line argument represents a term to search for. The remaining N arguments are all expected to be names of existing bash commands. The script should search through the text of each of the bash commands in a case-insensitive way, and find the one that contains the most occurrences of the keyword.

For example:

```
$ sm.sh size grep ls du cut
Best match: du
(term appeared 25 times)
```

In this command, we specified that we wanted to search for the term "size" in the man pages of the commands grep, ls, du, and cut. After the script conducted the searches, it determined that the man page for **du** contained "size" the highest number of times (25) and it summarized this result.

Another example:

```
$ bash sm.sh process kill sed grep top ls pwd man
Best match: top
(term appeared 95 times)
```

You can assume that the program will always be provided with two valid command-line arguments. You should put the following line near the beginning of the script: **export MANROFFOPT="-rHY=0 -rLL=32760"**

Submitting Your PA

After you have completed the two parts of this PA, double check that the file structure and file / directory names match what is shown in the project overview on the first page. Also, before you submit, you should ensure that your bash scripts run correctly on Lectura with numerous test cases, not just your own computer. Also ensure you don't have any extra files in your submission.

Once you are ready to submit, zip up the **`pa9`** directory by running:

```
$ zip -r pa9.zip ./pa9
```

Then, turn this file in to PA 9 on gradescope.

There will be some public test cases on Gradescope and also some hidden test cases that will not be revealed until after grades are published. Therefore, you should test your programs thoroughly!