CMPE 230 Systems Programming
Homework 2 (**due April 26th 23:59**)

Write a Python program that can be invoked with the following options and arguments:

```
duplicates  < -c <command> | -p >   [-f | -d]  [“...”] [<dir1> <dir2> ..]
```

The **duplicates** program will traverse the directories and look for files or directories that are exact duplicates of each other. It will then carry out an action on the files or directories in question. The explanations of the options and arguments are given in the following table:

< -c <command> | -p >	If -c is given, a command will be passed the list of files/directories which are duplicates. If -p is given , then the duplicates will be printed (a new line should be printed between the sets of duplicates)
[-f | -d]	-f means look for duplicate files, -d means look for duplicate directories. The default is duplicate files.
[“...”]	Consider only the filenames or directory names that match the Python pattern given in “...“. The default is all files or all directories
[<dir1> <dir2> ..]	The list of directories to traverse (note that the directories will be traversed recursively, i.e. directories and their subdirectories and their subdirectories etc. etc.). The default is current directory.

Note:

- Assume that directory hierarchy forms a tree.
- Duplicate will mean the contents are exactly the same (note that the names can be different). You can take sha256 hashes of files and compare the hashes in order to locate duplicate files and directories.
- To locate duplicate directories, you can use hash trees.
- You can use python2 or python3.

Grading

We are going to grade your projects with testcases by creating random files and directories. When we create example testcases we are going to share them with you on the piazza. We will also grade the documentation of the project and the comments in the code:

Documentation (written document describing how you implemented your project)	12 %
Comments in your code	8 %
Implementation and tests	80 %

Submission

You are going to submit your project as a single compressed file having python source codes and the documentation on the canvas course page.

Late Submission

If the project is submitted late, the following penalties will be applied:

- 0 < hours late <= 24 : 25%
- 24 < hours late <= 48 : 50%
- hours late > 48 : 100%

Proof of Existence

Before you submit your project, please notarize your project zip file at http://virtual-notary.org/ by using the “Notarize a document” button and save the details of the notary log. Do NOT lose the notary log and the project zip file. It is a proof that your project zip file existed during the time of submission. If for some reason something went wrong with submission, notary log and the corresponding project zip file will prove that your project zip file existed. Only the project zip file that matches the notary log will be accepted. If you show a project zip file that does not match (correspond to) the notary log, it will NOT be accepted !.