# CFRS 510 HW3

## Purpose:

Using Python, traverse a directory and its subdirectories to locate and compute the md5 hash of each JPEG file.

## Procedure:

1. Download and extract the contents of the cfrs510hw3.rar file from BB, keeping the file structure intact.
2. Create a Python script (.py or .ipynb extension) that will:
    - **a.** Request from the user a directory to investigate (and its subfolders)
    - **b.** Traverse the given path looking for JPEG files.
    - **c.** Once identified, check if any data has been appended to the image file.
        - **i.** If so, decode the appended data (you may assume that it is base64 encoded).
    - **d.** Store the file name, md5 hash (minus any appended data) and the file's creation, access and modification time stamps and appended message (if applicable) in a dictionary.
    - **e.** Save the dictionary to output.txt
    - **f.** Print the location of the output file and the time the script completed to the console.

# Requirements:

- Module (python script) must be titled *hw3LastNname.py* (with "LastName" being your last name).
- Module must contain the following functions:
    - **md5HashFunc** – this function should **take in content to be hashed** and **return** the content's md5 hash value.
    - **b64decode** – this function should take base64 encoded content and return the **string** representation of the content.
    - **main** –
        - Ask user for directory path (root directory) to investigate.
        - Traversed the directory and its subdirectories looking for files that are .jpg format based on the file header (not the extension).
        - If a jpg file is found:
            - Check for appended data.
                - If appended data found, decode the information.
            - Compute the md5 hash (**NOT to include the appended data**).
            - Get file's creation, last modified, and access times.
        - The filename, hash and timestamps are stored in a dictionary.
        - Store dictionary contents in an **output.txt file, along with the hashed file's creation, last modified, access times, and decoded message (if any).** *Hint: use a list and keep items pertaining to each file in the same order*
    - **if __name__ == '__main__'**
- Gracefully handle any potential errors (e.g. imports, file paths, etc.).
- Print to the console:
    - Time script completes execution.
    - Path and name of the **output file** that contains the hashes.

## Grading Rubric – 10 points possible

-3 Program crash (to include unhandled exceptions)
-3 Incorrect output (to include handled exceptions – meaning no try/except without naming the specific exception you intend to catch)
-2 Late submission
-2 Lack of comments (to include docstrings)
-1 Incorrect function names
-1 Wrong script name