COMPSCI340 and SOFTENG370 Assignment 2 marking guide

Marking is to be done on the lab Linux image, as the students were told, "You may use any programming language you like (as long as it can be run on Linux in the labs by the markers)." If any extra libraries etc are needed they must be provided in the student's submission and any installation must be done via the make file (or one command line you may be asked to execute).

For some of the students the hardest part of this assignment will be submitting the correct files. They are supposed to submit source code and some sort of make or other script file which will allow an executable file called "sync" to be created. It must be called this because the test program uses this name to invoke the program. Some students may actually submit jar files. They may be using a jar file for JSON or other library routines.

Clear your marking directory for each student and dump their files along with the test_for_markers.py file into that directory before marking each submission.

If the students submitted a make file then you should be able to set everything up by typing "make". Otherwise the assignment states "You may give the marker a single line of instructions to run this makefile or script so that it completely produces a runnable version of your `sync` program in the current directory. " Unfortunately I did not state where this line of instruction should go. I hope it is obvious at the top of a source file or the A2Answers.txt file.

If there are problems e.g. the sync program is not executable please try some simple fixes but don't spend too much time on it. You might do "chmod +x sync" and see if that works. Or if the program is in Java you could use a variant of the makefile I have included for you. This makefile presumes the class name is Sync and requires a specific gson jar file. If you do have to spend time trying to make an executable file you should take off 2 marks as specified in the mark sheet.

If the program crashes running a particular test please comment that test out (at the bottom of the test_for_markers.py program) and try again for the following tests.

The output files can be in a different order than my sample output but the list of times and digests must have the most recent time/digest at the top for each file.

Some people have generated extra information in the sync files, depending on the language they used. That is fine as long as the file names, and time/digest pairs are easy to check.

Here is the test output from my solution for comparison:

```
rshe005$ python3 test_for_markers.py
**************************************
1. This should print an error (or usage) message:
Usage: sync directory1 directory2
**************************************

**************************************
2. This should show both dirA and dirB as directories (ignore others):
dirA
dirB
**************************************

**************************************
3. This should show sync file contents:
{
  "file1_1.txt": [
    [
      "2015-10-05 16:28:08 +1300",
      "a1d99931f6485fc5bd5a74e8e3b368f1d761aaa578e29da23d181f3f9e8a8ffd"
    ]
  ]
}
**************************************
```

```
**************************************
4. This should show two different sync files.
   The first appears as the last entry in the second:
{
  "file1_1.txt": [
    [
      "2015-10-05 16:28:08 +1300",
      "a1d99931f6485fc5bd5a74e8e3b368f1d761aaa578e29da23d181f3f9e8a8ffd"
    ]
  ]
}
{
  "file1_1.txt": [
    [
      "2015-10-05 16:28:08 +1300",
      "79c64b16079845e5207a1c1507492d749496b8fea3608b3588138e873ef8e0bd"
    ],
    [
      "2015-10-05 16:28:08 +1300",
      "a1d99931f6485fc5bd5a74e8e3b368f1d761aaa578e29da23d181f3f9e8a8ffd"
    ]
  ]
}
**************************************




**************************************
5. This should show two matching directories.
   Times and sizes should match in both directories:
-a.txt 64 2015-10-05 16:28:08 +1300
-b.txt 63 2015-10-05 16:28:08 +1300
-c.txt 63 2015-10-05 16:28:09 +1300
-d.txt 50 2015-10-05 16:28:09 +1300

-a.txt 64 2015-10-05 16:28:08 +1300
-b.txt 63 2015-10-05 16:28:08 +1300
-c.txt 63 2015-10-05 16:28:09 +1300
-d.txt 50 2015-10-05 16:28:09 +1300
**************************************

**************************************
6. This should show two matching directories
   with the file 'a.txt' 76 bytes long
   and the modification time later than b.txt:
-a.txt 76 2015-10-05 16:28:12 +1300
-b.txt 63 2015-10-05 16:28:09 +1300
-c.txt 63 2015-10-05 16:28:10 +1300
-d.txt 50 2015-10-05 16:28:10 +1300

-a.txt 76 2015-10-05 16:28:12 +1300
-b.txt 63 2015-10-05 16:28:09 +1300
-c.txt 63 2015-10-05 16:28:10 +1300
-d.txt 50 2015-10-05 16:28:10 +1300
**************************************

**************************************
7. This should show two pairs of matching directories
   with the second pair showing different sizes and modification times.
   a.txt may have the same time but different size.
   d.txt should be unchanged:
-a.txt 76 2015-10-05 16:28:15 +1300
-b.txt 63 2015-10-05 16:28:12 +1300
-c.txt 63 2015-10-05 16:28:13 +1300
-d.txt 50 2015-10-05 16:28:13 +1300

-a.txt 76 2015-10-05 16:28:15 +1300
-b.txt 63 2015-10-05 16:28:12 +1300
-c.txt 63 2015-10-05 16:28:13 +1300
-d.txt 50 2015-10-05 16:28:13 +1300
```

```
 --- then after the change ---

-a.txt 93 2015-10-05 16:28:15 +1300
-b.txt 80 2015-10-05 16:28:15 +1300
-c.txt 80 2015-10-05 16:28:15 +1300
-d.txt 50 2015-10-05 16:28:13 +1300

-a.txt 93 2015-10-05 16:28:15 +1300
-b.txt 80 2015-10-05 16:28:15 +1300
-c.txt 80 2015-10-05 16:28:15 +1300
-d.txt 50 2015-10-05 16:28:13 +1300
**************************************

**************************************
8. This should show the two directories.
   Then they should only contain c.txt.
   Then they should have a new version of b.txt:
-a.txt 64 2015-10-05 16:28:15 +1300
-b.txt 63 2015-10-05 16:28:15 +1300
-c.txt 63 2015-10-05 16:28:16 +1300
-d.txt 50 2015-10-05 16:28:16 +1300

-a.txt 64 2015-10-05 16:28:15 +1300
-b.txt 63 2015-10-05 16:28:15 +1300
-c.txt 63 2015-10-05 16:28:16 +1300
-d.txt 50 2015-10-05 16:28:16 +1300

 --- only c.txt ---

-c.txt 63 2015-10-05 16:28:16 +1300

-c.txt 63 2015-10-05 16:28:16 +1300

 --- new b.txt ---

-b.txt 57 2015-10-05 16:28:18 +1300
-c.txt 63 2015-10-05 16:28:16 +1300

-b.txt 57 2015-10-05 16:28:18 +1300
-c.txt 63 2015-10-05 16:28:16 +1300

 --- dirA sync file ---
{
  "a.txt": [
    [
      "2015-10-05 16:28:17 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:15 +1300",
      "421d7cd3a9189f174e4e88a720015e2d2a86adeed9819d47e6c703d18fc18cf8"
    ]
  ],
  "b.txt": [
    [
      "2015-10-05 16:28:18 +1300",
      "e5bd87cd9e408905c4c6f6776005d5c638e3e9d211da41a5d14fad3db3f24b5e"
    ],
    [
      "2015-10-05 16:28:18 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:17 +1300",
      "7bb8166f6771718eab5b0041d9cae5e5b8470a69e1804bdea4a010e46521c7d9"
    ],
    [
      "2015-10-05 16:28:17 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:15 +1300",
```

```
      "3dccaa6062aeb3f614631cfaf3c151dfc2de0b10ae0fd0634e15cd2cc3ba7814"
    ]
  ],
  "c.txt": [
    [
      "2015-10-05 16:28:16 +1300",
      "5d93224a36d2e7e3c5f4c1af723120c648da8badc16ab9d87fbfb3d819824b71"
    ]
  ],
  "d.txt": [
    [
      "2015-10-05 16:28:17 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:16 +1300",
      "2cc9853ba2b97ea606df86479da7daa49072920025467476d5a74ef404cab01f"
    ]
  ]
}

 --- dirB sync file ---
{
  "c.txt": [
    [
      "2015-10-05 16:28:16 +1300",
      "5d93224a36d2e7e3c5f4c1af723120c648da8badc16ab9d87fbfb3d819824b71"
    ]
  ],
  "d.txt": [
    [
      "2015-10-05 16:28:17 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:16 +1300",
      "2cc9853ba2b97ea606df86479da7daa49072920025467476d5a74ef404cab01f"
    ]
  ],
  "a.txt": [
    [
      "2015-10-05 16:28:17 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:15 +1300",
      "421d7cd3a9189f174e4e88a720015e2d2a86adeed9819d47e6c703d18fc18cf8"
    ]
  ],
  "b.txt": [
    [
      "2015-10-05 16:28:18 +1300",
      "e5bd87cd9e408905c4c6f6776005d5c638e3e9d211da41a5d14fad3db3f24b5e"
    ],
    [
      "2015-10-05 16:28:18 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:17 +1300",
      "7bb8166f6771718eab5b0041d9cae5e5b8470a69e1804bdea4a010e46521c7d9"
    ],
    [
      "2015-10-05 16:28:17 +1300",
      "deleted"
    ],
    [
      "2015-10-05 16:28:15 +1300",
      "3dccaa6062aeb3f614631cfaf3c151dfc2de0b10ae0fd0634e15cd2cc3ba7814"
    ]
  ]
}
```

```
************************************

************************************
9. This starts by showing the initial synchronized directories.
   file1_1.txt must be 81 bytes long:
-dirA_1
    -dirA_1_1
        -file1_1_1_1.txt 94 2015-10-05 16:28:18 +1300
-dirA_2
    -file1_2_1.txt 54 2015-10-05 16:28:18 +1300
-dirB_1
    -file2_1_1.txt 78 2015-10-05 16:28:18 +1300
-file1_1.txt 81 2015-10-05 16:28:20 +1300
-file1_2.txt 69 2015-10-05 16:28:18 +1300
-file2_1.txt 69 2015-10-05 16:28:18 +1300

-dirA_1
    -dirA_1_1
        -file1_1_1_1.txt 94 2015-10-05 16:28:18 +1300
-dirA_2
    -file1_2_1.txt 54 2015-10-05 16:28:18 +1300
-dirB_1
    -file2_1_1.txt 78 2015-10-05 16:28:18 +1300
-file1_1.txt 81 2015-10-05 16:28:20 +1300
-file1_2.txt 69 2015-10-05 16:28:18 +1300
-file2_1.txt 69 2015-10-05 16:28:18 +1300

 --- delete all files ---

-dirA_1
    -dirA_1_1
-dirA_2
-dirB_1

-dirA_1
    -dirA_1_1
-dirA_2
-dirB_1

 --- recreate two ---

-dirA_1
    -dirA_1_1
        -file1_1_1_1.txt 74 2015-10-05 16:28:20 +1300
-dirA_2
-dirB_1
-file1_1.txt 54 2015-10-05 16:28:20 +1300

-dirA_1
    -dirA_1_1
        -file1_1_1_1.txt 74 2015-10-05 16:28:20 +1300
-dirA_2
-dirB_1
-file1_1.txt 54 2015-10-05 16:28:20 +1300
************************************
```

Answers to the questions:

1.

How well would this synchronisation implementation cope with a large number of files? Explain your answer.

A big problem with this implementation is that whenever the sync command is executed every file is read again and its SHA256 digest recalculated. As the number of files gets very large this would not scale well at all. (The students don't need to provide a solution to this problem.)

2.

Several of the assumptions cannot be enforced. We rely on the user being careful. This is partly because of the limitations of most current file systems. Take two of the assumptions above (you may also specify other assumptions that I haven't explicitly mentioned), and describe ways in which the assumptions can be made unnecessary. Your solutions should make the synchronization process safer or more reliable. At least one of the solutions should mention some extra facility that must be added to the file system to make the solution possible.

I wanted this to be as open as possible, but I did want the students to think about ways in which a better synchronization method would be implemented. Remember, one of the suggestions must mention changes to the file system.
Length of the answer is irrelevant; you need to check for:

clarity – can you understand what the student is saying

usefulness – does it really make the synchronization process safer feasibility – does it make sense, could it actually be implemented

Here are some examples (the first and third require changes to the file system):

a.
Assumption:
No symbolic links will create cycles in the directories.

Fix:
Cycles in the directory structure can be allowed as long as the file system maintains unique identifiers on each directory. These identifiers must be accessible to the sync program. Then the sync program needs to keep track of any directories it has already traversed to ensure it does not go into an infinite loop. This way even recursive directory structures can be synchronized.

b.
Assumption:
You don't have to deal with deleted directories, but you do have to deal with deleted files.

Fix:
Deleted directories can be handled by adding information about the directories to the sync files. Signatures don't need to be maintained, as the contents of the directory determine whether the directory has been modified or not. The system needs to record when a directory has been deleted. As with the file deletions a unique time or number has to be associated with the deletion of a directory so that recreations of directories can be allowed. There is a real danger depending on how directory deletion is handled. If entire subtrees can be deleted with a single command then it would be safer to prompt the user when synchronizing the directories rather than just deleting all files in the matching subtree.

c.
Assumption:
No sync files will be deleted by the test programs.

Fix:
Don't keep the sync information in an ordinary file. It could be an attribute of all directories. This requires a change to the file system. When a directory is created it is allocated space for its sync data. There would have to be system calls that would allow access to the sync data area, both for reading and writing purposes. The only way the sync information could be lost would be if the directory itself is deleted.

There are other assumptions that students can work out for themselves that weren't explicitly mentioned. e.g., The user only ever syncs two directories A and B, but doesn't sync A with C and A

with B at different times.
Another interesting assumption is that the current system doesn't record if a user creates and deletes a file multiple times between syncs. Most file actions done between syncs will be ignored. One fix for this would be to have the file system maintain versioning information, automatically recording all changes and then using this information when called upon to synchronize two directories.