EE-3233

Systems Programming for Engineers

Lab Assignment 7

This assignment has a high difficulty level, start early.

In this assignment you must create a Python3 script, with a **`main()`** entry point:

```
if __name__ == "__main__":
      main()
```

that creates five threads to divide a workload in five. In the script you must include a global List (**`glist`**) that is initialized with all the numbers from 0 to 499. Then, each thread will be given as a target an **`accumulate(i: int)`** function, where the **`i`** argument depicts a starting index that the function will use to identify its workload. For example:

Function Invocation	Numbers Processed	Expected work (acc)
`accumulate(0)`	0, 1, 2, …, 99	0 + 1 + 2 + … + 99
`accumulate(1)`	100, 11, 12, …, 199	100 + 111 + 112 + … + 199

Here, the accumulate function will add all its corresponding values from **`glist`** and put the result in a local **`acc`** variable. To make its **`acc`** value visible to the rest of the program, you can create a global accumulate list (**`alist`**) and initialize it to zeros:

```
alist = [0] * 5
```

Since the workload is being distributed across five threads, **`alist`** must be able to contain five numbers. Therefore, the accumulate function can save its result by executing this line of code towards the end of its definition:

```
alist[i] = acc
```

Once all the threads have completed their own work, your main thread will then add all the values in **`alist`** to calculate the total accumulated value, which should be 124750.

Monitoring

Put this line of code towards the end of the **`accumulate()`** function to monitor the work it has done:

```
print(f"Accumulated value in thread [{tid} -> {i}] is {acc}")
```

Where **tid** is a variable that contains the thread ID. To get it, simply run:

```
tid = threading.get_native_id() # If working in the VM.
```

Or:

```
tid = threading.get_ident() # If working with an online interpreter, etc…
```

In your main thread, after you have joined all the other threads and calculated the total accumulated value, put this line of code to see the result:

```
print(f"Total is: {total}")
```

Where **total** is a local variable that holds the value **alist[0]** + **alist[1]** + …

Expected Output

```
Accumulated value in thread [3410 -> 0] is 4950
Accumulated value in thread [3411 -> 1] is 14950
Accumulated value in thread [3412 -> 2] is 24950
Accumulated value in thread [3413 -> 3] is 34950
Accumulated value in thread [3414 -> 4] is 44950
Total is: 124750
```

Deliverables

You must upload the script with your solution, named with the following format:

<name-of-file>_<first-name>_<last-name>.py

For example:

acc_andres_hernandez.py

Report-like submissions in PDF format will no longer be accepted.

Grading

- Your solution will be evaluated with a script to check if your solution is correct.
- Partial marks will be granted for cases where the output is partially correct.

Furthermore, following the syllabus, your script will be evaluated against MOSS (http://theory.stanford.edu/~aiken/moss/) to detect similar code.