COMP 3500 Homework #3

Maximum Points Possible: 100
Team Assignment (1..2 members per team)

There should be no collaboration among students (teams). A student/team shouldn't share any project code with any other student. Collaborations among students in any form will be treated as a serious violation of the University's academic integrity code.

Objectives: To learn the following.

1. Process/Thread synchronization using the Mutex locks, and Semaphores.
2. Usage of POSIX PThread library.
3. Read/Write to files

Instructions:

1. This project can be submitted individually, or in teams of two members only.
2. Program must be written in C/C++ language.
3. Your program must take the input file name as command line parameters.
4. Assume that the input file only has 100 integer numbers.
5. Your program will be tested with multiple input files.
6. Perform necessary error checking of command line parameters.
7. Perform appropriate error checking at each step.

Deliverables:

1. Part-1:
    a. C/C++ file implementing Part-1 of the problem.
    b. Input file
    c. Output file from a sample run
2. Part-2:
    a. C/C++ file implementing Part-2 of the problem.
    b. Three output files from three sample runs
    c. Response to Part-2(2).
3. Part-3:
    a. C/C++ file implementing Part-3 of the problem.
    b. Output file from a sample run
    c. Response to Part-3(2).
4. Execution instructions, if any.

Problem: Distribution.

Input: Input file comprising 100 integers ranging from 0 to 99

85	47	5	8	83	97	61	30	62	91
7	56	73	90	40	49	86	98	68	68
80	46	97	11	39	60	74	11	29	64
53	77	46	76	52	5	58	49	59	56
73	24	34	91	47	2	29	72	7	20
11	82	74	53	63	89	63	51	50	8
75	21	62	4	37	92	44	42	68	22
23	33	33	6	31	75	33	9	79	92
62	96	63	6	22	10	26	39	15	27
86	57	76	97	48	46	90	68	26	21

Output: Output file comprising the distribution counts of numbers in the following 10 ranges:

Group 1: 0..9 Group 2: 10..19 Group 3: 20..29 Group 4: 30..39 Group 5: 40..49
Group 6: 50..59 Group 7: 60..69 Group 8: 70..79 Group 9: 80..89 Group 10: 90..99

Part–1 (20 Points)

1. Write a program in C/C++ that computes the distribution as follows:
    a. (5 Points) Read 100 integers from given input file
    b. (10 Points) For each integer,
        i. Identify the group (bin) that it belongs to, and
        ii. Increment the respective group count.
    c. (5 Points) Print to the output file in following format:

```
Group 1 (0..9) Count:
Group 2 (10..19) Count:
Group 3 (20..29) Count:
Group 4 (30..39) Count:
Group 5 (40..49) Count:
Group 6 (50..59) Count:
Group 7 (60..69) Count:
Group 8 (70..79) Count:
Group 9 (80..89) Count:
Group 10 (90..99) Count:
```

2. Provide following:
    a. Output from a sample run.

Part – 2 (20 Points)

1. Re-implement Part – 1 with POSIX Pthreads as follows.
    a. Read 100 integers from given input file into an array, or a list.
    b. Create 10 concurrent threads.
    c. Partition the list into 10 parts comprising 10 integers each and assign one part to each thread. Pass the starting index of sub-list as parameter. For example:
        i. Thread 6 processes the array elements [50]..[59]; hence, pass the starting index 50 as a parameter.
    d. Each thread performs following task:
        i. For each integer in the assigned sub-list:
            1. Identify the group (bin) that it belongs to, and
            2. Increment the respective group count (global variable).
        ii. Terminate the thread.
    e. Prints to the output file in the format specified in Part-1(1)(c).
2. Provide following:
    a. Provide output from three sample runs.
    b. Do you notice any differences in results? What did you observe? Provide reasoning.

Part – 3 (60 Points)

1. Re-implement Part – 2(1)(d) as follows implementing synchronization mechanisms discussed in class to ensure that each thread atomically updates the group counts.
    a. Declare and initialize 10 mutex locks, or 10 semaphores (based on the synchronization mechanism you choose). Each mutex lock / semaphore represents one group-count global variable.
    b. Each thread performs following task:
        1. For each integer in the assigned sub-list:
            1. Identify the group (bin) that it belongs to, and
            2. Acquire exclusive access to the group-count global variable by acquiring respective mutex lock / semaphore.
            3. Increment the respective group count (global variable).
            4. Explicitly release the lock.
        2. Terminate the thread.
    c. Prints to the output file in the format specified in Part-1(1)(c).
2. Provide following:
    a. Output from a sample run.
    b. Compare the outputs from Part-1 and Part-2. What did you observe? Provide reasoning.
    c. Provide a brief description of the implementation of synchronization approach.