ANKARA UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

COM/BLM3035 Operating Systems/İşletim Sistemleri

Fall 2024-25

LAB2

BLM3035

Soru: Semaphore Kullanarak Çoklu İş Parçacıklı (Multithreaded) Öğrenci Not Analizi Sistemi

Bir girdi dosyasında saklanan öğrenci notlarını işleyen çoklu iş parçacıklı bir program tasarlayın. Program, her öğrencinin ortalama notunu hesaplamalı, geçip geçmediğini belirlemeli ve sonuçları bir çıktı dosyasına yazmalıdır. Ayrıca, her soru için geçen öğrenci sayısı, en yüksek ve en düşük notlar ve genel olarak geçen öğrenci sayısı gibi genel istatistikleri de hesaplamalıdır. **Semaphore'lar iş parçacığı senkronizasyonu için kullanılacak ve dosya okuma ve yazma işlemleri gerçekleştirilecektir.**

Kurallar: Kopya kontrolü yapılacaktır. Kodunuz tamamen kendinize özgü olmalıdır. %85 üstünde kalan herkes kopya olarak nitelendirilecektir. Geç gönderim kabul edilmeyecektir. Output hatası -%25 puan olarak değerlendirilecektir.

Gereksinimler:

1. Girdi Dosyası Formatı (input.txt**):**

- İlk satır öğrenci sayısını (N) belirtir.
- İkinci satır, öğrenci başına girilecek not sayısını (M) belirtir.
- Takip eden her satır, bir öğrenci kimliği ve ardından M notunu içerir.

Örnek:

3

4

101 78 85 90 45

102 45 60 55 30

103 88 92 95 90

2. Çıktı Dosyası Formatı (results.txt**):**

- Her öğrenci için: öğrenci kimliği, ortalama notu ve geçti/kaldı durumu.
- Sonunda: aşağıdaki genel istatistikler:
  - Her soru için geçen öğrenci sayısı.
  - Genel olarak geçen toplam öğrenci sayısı.
  - En yüksek ve en düşük notlar.

Örnek:

101 74.50 Passed

102 47.50 Failed

103 91.25 Passed

--- Overall Statistics ---

Number of students passing each question:

Question 1: 2 students passed.

Question 2: 3 students passed.

Question 3: 2 students passed.

Question 4: 1 students passed.

Total number of students who passed overall: 2

Highest grade: 95

Lowest grade: 30

3. Uygulama Detayları:

- İş parçacıkları kullanarak her öğrencinin notlarını paralel olarak işleyin.
- Semaphore kullanarak istatistiklerin (örneğin, en yüksek/en düşük not, geçen öğrenci sayısı) güvenli bir şekilde güncellenmesini sağlayın.
- Girdi dosyasına bağlı olarak belleği dinamik olarak ayırın.

4. Görevler:

1. Girdi dosyasını okuyarak N (öğrenci sayısı) ve M (not sayısı) değerlerini alın.
2. Her öğrenci için şu işlemleri gerçekleştirin:
    - Ortalama notu hesaplayın.
    - Geçti/kaldı durumunu belirleyin (geçti: ortalama >= 60).
3. Küresel istatistikleri güncelleyin:
    - Her soru için geçen öğrenci sayısı.
    - En yüksek ve en düşük notlar.
    - Genel olarak geçen öğrenci sayısı.
4. Sonuçları ve küresel istatistikleri çıktı dosyasına yazın.

5. Derleme ve Çalıştırma:

- Kodunuzu bir dosyaya Ogrenci_No.c (örneğin, 18888888.c) adıyla yazın.
- Aşağıdaki komutlarla programınızı derleyin ve çalıştırın:

gcc 18888888.c -o a.out -lpthread

./a.out

- Çıkış dosyanızı results.txt olarak kaydetmek için programınızı çalıştırırken şu komutu kullanın: ./a.out.
- Çıktınızı doğru yüklediğinizden emin olmak için komutu çalıştırarak karşılaştırma yapabilirsiniz: **diff -w results.txt results1.txt**

COM3035

Question: Multithreaded Student Grade Analysis System with Semaphores

Design a **multithreaded program** that processes student grades stored in an input file. The program should calculate each student's average grade, determine if they passed or failed, and write the results to an output file. Additionally, it should compute global statistics, such as the number of students passing each question, the highest and lowest grades, and the total number of students passing overall. **You will use semaphores for thread synchronization and perform file reading and writing operations.**

Rules: Duplicate checks will be made. Your code must be completely unique to you. Anyone above 85% will be considered a duplicate. Late submissions will not be accepted. Output errors will be evaluated as -25% points.

Requirement

1. Input File Format (input.txt):

- The first line specifies the number of students (N).
- The second line specifies the number of grades per student (M).
- Each subsequent line contains a student's ID followed by their M grades.

Example:

3

4

101 78 85 90 45

102 45 60 55 30

103 88 92 95 90

2. Output File Format (results.txt**):**

- For each student: their ID, average grade, and pass/fail status.
- At the end: global statistics, including:
  - Number of students passing each question.
  - Total number of students passing overall.
  - The highest and lowest grades.

Example:

101 74.50 Passed

102 47.50 Failed

103 91.25 Passed

--- Overall Statistics ---

Number of students passing each question:

Question 1: 2 students passed.

Question 2: 3 students passed.

Question 3: 2 students passed.

Question 4: 1 students passed.

Total number of students who passed overall: 2

Highest grade: 95

Lowest grade: 30

3. Implementation Details

- Use **threads** to process each student's grades in parallel.
- Use **semaphores** to safely update global statistics (e.g., highest/lowest grade, passing counts).
- Dynamically allocate memory based on the input file.

4. Tasks:

- Parse the input file to read N (number of students) and M (number of grades per student).
- Compute for each student:
    - Average grade.
    - Pass/fail status (pass if average >= 60).
- Update global statistics:
    - Number of students passing each question.
    - Highest and lowest grades.
    - Total students passing.
- Write results and global statistics to the output file.

5. Compilation and Execution:

- You should write your code in a file named *Student_Id.c* (e.g., 18888888.c).
- Compile it using the command: ***gcc 18888888.c -o a.out -lpthread*** , and run it with the command: ***./a.out*** , making use of the provided input files.

- This task challenges you to implement **multithreading**, manage **semaphore synchronization**, and perform efficient **file I/O operations** for real-time data processing.
- To save your own output name as **result.txt,** use the command: ***./a.out.***
- Ensure that you have uploaded your output correctly by using the command: ***diff -w results.txt results1.txt*** .
- Due to thread execution, the order of the outputs such as: **'101 84.33 Passed , 102 53.33 Failed ... '** may vary. These discrepancies will be disregarded.