ANKARA UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLM/COM334

LAB 3

input.txt isimli bir dosyanız var. **X sayıda iplik** (thread) oluşturarak bu dosyanızı satır satır okumak ve her bir ipliğin kendi okuduğu kısımda bulunan **cümle sayısını** bulmanız isteniyor.

Bu amaçla yazdığınız **main fonksiyonunuz** sırasıyla aşağıdaki işlemleri yapmalıdır:

1. Kaç tane iplik oluşturacağını belirten **x değerini kullanıcıdan almalıdır.**

2.Dosyanın satır sayısını hesaplayıp, **her bir ipliğin kaç satır okuması** gerektiğine karar vermelidir. Bu değer tam çıkmadığında değer **yukarı yuvarlanmalıdır.**

3. İplikleri oluştururken, ilgili ipliğin hangi satırdan itibaren okuma yapacağını belirtmelidir.

4. İplikten geri gelen **okunan satır sayısı ve cümle sayısı** değerleri ekrana yazdırmalıdır.

**Her bir ipliğiniz** aşağıdaki işlemleri yapmalıdır:

1. Main fonksiyonunun atadığı satır sayısından başlayarak **belirtilen satır sayısı kadar** dosyadan okuma yapmalıdır.

2.Tahmin edileceği üzere son iplik diğerlerinden az sayıda satır okuyabilir (Yukarı yuvarlama yaptığınızdan dolayı). Bu nedenle **kaç satır okuduğunu** da main fonksiyonuna **geri döndürmelidir**. Bunun yanında soruda istenilen **cümle sayısını** da main fonksiyona **gönderecektir**.

Bütün bu işlemleri yapan bir C/C++ programı yazınız. İplik oluşumu ve kullanımı için **pthread kütüphanesindeki pthread_create** ve **pthread_join** komutlarını inceleyebilirsiniz. Programınızı çalıştırmak için (C kodu için):

gcc StudentNumber.c –lpthread

./a.out

---

You have a file named input.txt. Read this file line by line by creating **X threads**. Each thread **find the number of sentences** in its own section. For this purpose, your main function includes the following operations in order:

1. Take the value (x) from the user - **X = Number of threads** .

2. Calculate the number of lines of the file and decide **how many lines each thread** should read. The value should be **rounded up** when this value is not an integer

3. When creating the threads, it must be indicated **the begining line of thread**

4. **The number of scanned lines** and **number of sentences** returned from the thread must be **printed.**

Each of your threads should do the following:

1. Started from the number of lines assigned by the Main function.

2. The last thread can scan fewer lines than the others (because you round up). Therefore, it should return **how many lines th thread scans** to the main function. In addition, it will send the **number of sentences**.

Write a **C / C ++** program that does all this. You can examine the **pthread_create** and **pthread_join** commands in **pthread library**. To run your program (for C code):

gcc StudentNumber.c –lpthread

./a.out