**ANKARA UNIVERSITY**
**COMPUTER ENGINEERING DEPARTMENT**
**BLM/COM3035**
**2022-2023**
**LAB 2**

Beş iplik (thread) oluşturan bir C programı yazınız. Programınız input.txt olarak adlandırılan girdi dosyasını satır satır okuyarak sayıların toplamını ekrana yazdıracaktır. Programınızın main fonksiyonunda dosyanın satır sayısı hesaplanıp, her bir ipliğin kaç satır okuması gerektiğine karar verilmelidir. Örneğin girdi dosyası 10 satır içeriyorsa ve 5 adet de iplik var ise iplikler her seferinde 2 satırı okumalıdır. Bu değer küsuratlı çıktığında değer yukarı yuvarlanmalıdır. Örneğin 14 satırlık bir dosya okunuyorsa ve eldeki iplik sayısının da 5 olduğu varsayıldığından her iplik operasyonunda 3 satır okunacaktır. Satır sonunda ise ilgili iplik 3 yerine 2 satır okuyarak sonucu dönmelidir. Her iplik satır okuma ve hesaplama işleminin ardından kendi toplamlarını döndürerek toplam değerini ekrana yazdıracaktır. Girdi dosyanız ve toplam sonucunun tutulacağı global değişken ortak olduğu için erişim ayarlarını yapmanız gereklidir (**semafor kullanmalısınız**). İpliklerin sıra ile çalışması zorunlu değildir. Fakat her ipliğin sürece dahil olup dosyadan okuma yapması gerekmektedir. Örnek girdi dosyası ve program çıktısı aşağıdaki gibi olmalıdır.

input.txt:
1.23
2.66
3.45
81.23
9.11
3.33
7.78
4.12
1.11
2.22

```
Thread 1 reads 1. line. The value is 1.23
Thread 1 reads 2. line. The value is 2.66
Shared Resource: 3.89
Thread 2 reads 3. line. The value is 3.45
Thread 2 reads 4. line. The value is 81.23
Shared Resource: 88.57
Thread 3 reads 5. line. The value is 9.11
Thread 3 reads 6. line. The value is 3.33
Shared Resource: 101.01
Thread 4 reads 7. line. The value is 7.78
Thread 4 reads 8. line. The value is 4.12
Shared Resource: 112.91
Thread 5 reads 9. line. The value is 1.11
Thread 5 reads 10. line. The value is 2.22
Shared Resource: 116.24
```

İplik oluşumu ve kullanımı için pthread kütüphanesindeki pthread_create ve pthread_join gibi komutları inceleyebilirsiniz. Ayrıca semafor kullanımı için sem_init, sem_wait, sem_post komutlarını inceleyebilirsiniz.

Programınızı çalıştırmak için:
gcc StudentNumber.c
./a.out

**IN ENGLISH:**

Write a C program that creates five threads. Your program will read the input file called input.txt line by line and print the sum of the numbers on the screen. In the main function of your program, the number of lines of the file should be calculated and it should be decided how many lines each thread should read. For example, if the input file contains 10 lines and there are 5 threads, the threads must read 2 lines at a time. When this value is fractional, the value should be rounded up. For example, if a 14-line file is being read and the number of threads at hand is assumed to be 5, 3 lines will be read in each thread operation. At the end of the line, the relevant thread should read 2 lines instead of 3 and return the result. Each thread will return its own sums after line reading and calculation, and print the total value to the screen. Since your input file and the global variable to hold the total result are shared, you need to set the access settings (you should use semaphore). It is not necessary for the threads to run sequentially. But each thread must be involved in the process and read from the file. The sample input file and program output should be as follows.

input.txt:
1.23
2.66
3.45
81.23
9.11
3.33
7.78
4.12
1.11
2.22

```
Thread 1 reads 1. line. The value is 1.23
Thread 1 reads 2. line. The value is 2.66
Shared Resource: 3.89
Thread 2 reads 3. line. The value is 3.45
Thread 2 reads 4. line. The value is 81.23
Shared Resource: 88.57
Thread 3 reads 5. line. The value is 9.11
Thread 3 reads 6. line. The value is 3.33
Shared Resource: 101.01
Thread 4 reads 7. line. The value is 7.78
Thread 4 reads 8. line. The value is 4.12
Shared Resource: 112.91
Thread 5 reads 9. line. The value is 1.11
Thread 5 reads 10. line. The value is 2.22
Shared Resource: 116.24
```

You can examine the commands such as pthread_create and pthread_join in the pthread library for thread creation and usage. You can also examine the sem_init, sem_wait, sem_post commands for semaphore usage.

To run your program:
gcc StudentNumber.c
./a.out