152116023 İŞLETİM SİSTEMLERİNE GİRİŞ LAB.(B) **24/25 BAHAR**

Hafta 11, Laboratuvar Haftası 4, Uygulama 3: *Thread* Yaratma ve Çalıştırma

Çalışmanızı *ESUZEM*'deki ilgili alana yüklemeyi unutmayınız!
Uygulama prosedürünü terminal kullanarak gerçekleştiriniz.

Prosedür:
Belirli frekanslarda eş zamanlı çalışan iş yüklerinin organizasyonu sağlanacaktır. Bu doğrultuda bu iş yüklerini sembolize edecek şekilde çeşitli metinler *thread*ler kullanılarak ekrana basılacak ve sayılacaktır. Kodu yazınız.

1. Programın çalışması için gerekli kullanıcı parametreleri terminal çalıştırması sırasında argüman olarak gönderilmelidir (*Hint*: `./Uyg3A 2600 2 ...`).
   Hint: `int dummyint = atoi(char); //char* to integer in c`
   - İlk argüman, programın toplam çalışma süresini belirler. Genel olarak süre birimi kodlayıcı tarafından belirlenebilir; kontroller buna göre yapılmalıdır. Örnek çalıştırma kodunda süre birimleri **ms** olarak ele alınmıştır.
   - İkinci argüman, eş zamanlı olarak kaç işlem yükünün çalıştırılacağını belirler. Bu değere göre göre programın çalışması için gerekli toplam argüman sayısı netleşir.
   - Üçüncü ve dinamik olarak şekillenecek diğer argümanlar ise farklı iş yüklerinin parametrelerini sembolize eder.
     Örnek1: `./Uyg3A 4900 2 workA 470 workB 750`
     Örnek2: `./Uyg3A 5300 3 "+" 70 "*" 75 "." 99`
2. Bu doğrultuda algoritma, çalıştırılma sırasında elde edilen ikinci argüman olan *n* adet eş zamanlı iş yükü üzerinden kurgulanmalıdır. Gerekli kullanıcı verisi programa aktarıldıktan sonra işlemler başlar. Çalışma süresine ulaşıldığında *thread*lerin tekrar işlemleri de sonlanır.
   Hint: Eş zamanlı işlem sayısı kullanıcı tarafından girileceğinden oluşturulacak *thread* sayısı da dinamik olmalıdır.
3. Örnek ekrana yazdırma fonksiyonu *pseudo* kod parçacığı:
   Hint: *Thread*in bağlandığı fonksiyon?

```
#include <time.h> //usleep() kullanimi icin gerekli
/** @brief        : Hint: Kod parcacigi yalnızca islemin yapilisini ozetler.
*                   Entegrasyon için degisiklikler gerekebilir.
* @param *prTxt   : hangi metnin basilacagi bilgisi tutulur.
* @param seconds  : isleminin frekans bilgisi tutulur.
* @return         : -    */
void printCharMultipleTimes(char *prTxt, double seconds)
    while(true)
            count++;
            fflush(stdout);                     //anlik degisimler icin buffer temizlenmesi
            printf("%s", prTxt);
            usleep((int)(seconds*1000000));     //seconds=1 icin yaklasik 1 saniye bekler
    end while
end func
```

   *Thread*leri yaratırken *thread*in fonksiyonu içerisinde kullanılacak bilgiler argüman paslama ile gönderilecektir.

4. Her bir işlem yükünün bilgileri karma bir veri yapısında tutulabilir. Örnek *struct*:

```
struct multiplePrintData{
    pthread_t id;
    int threadid;
    char *printText;
    double waitDuration;
};
```

Yüklenmesi Gereken Dosyalar:

kaynak kod dosyası	**<OgrNo>_uyg<#><Şube>.<dil>**
çalıştırma sonuçları çıktı dosyası	**<OgrNo>_uyg<#><Şube>_output.txt**

Puanlama Sistemi:

	QUIZ	PERFORMANS	
Term.Argüman	25	10	kod düzeni
Thread işl.	40	10	yorum satırı
Doğru çıktı	15		

Uygulamalar performans ve uygulamanın doğru kısımlarına göre değerlendirilir. Yüklenmesi gereken dosyalar "**<OgrNo>_<Ders><Şube><AkademikYıl><GUZ/BHR/YAZ>_<UYG/HW><#>.zip**"(**Örnek: 152120151028_IsSisLabB2425BHR_UYG3.zip**) isimlendirme formatında sıkıştırılarak yükleme alanına yüklenir. Yükleme hatalarına ceza puanı uygulanır ve yüklenmeyen çalışmalar geçersiz sayılır.