دانشکده ی مهندسی کامپیوتر

دانشگاه صنعتی امیرکبیر

درصورت داشتن سوال درمورد این تمرین، سوال خود را با موضوع تمرین ۸ با ایمیل زیر درمیان بگذارید:

osfall2020@gmail. com

تمرین هشتم درس سیستم عامل

مهلت تحویل ساعت ۲۳:۵۹ روز ۳۰ آبان ۹۹

تمرینات را انفرادی حل کرده و در سایت مودل (courses.aut.ac.ir) با قالب زیر بارگزاری نمایید:

StudentID_Name_Last Name

۱- توضیح دهید که چگونه می توان یک monitor را با استفاده از semaphore پیاده سازی کرد.

۲- فرض کنید یک نانوایی وجود دارد که در مقابل آن دو صف تشکیل شده. در هر لحظه تنها **یک نفر** می تواند در مقابل نانوا قرار گرفته و از او نان بگیرد. با استفاده از سمافورها ، راه حلی ارائه کنید تا اولا ، بیش از یک نفر در مقابل نانوا قرار نگیرد و ثانیا هیچ یک از دو صف دچار قحطی نشود.

همچنین می توانید پیاده سازی های توابع signal و wait را مطابق زیر فرض کنید:

```
wait(semaphore *S) {
  S->value--;
  if (S->value < 0) {
    add this process to S->list;
    block();
  } }
```

```
signal(semaphore *S) {
  S->value++;
  if (S->value <= 0) {
    remove a process P from S->list;
    wakeup(P);
  } }
```

۳- برای مشکل readers writers راه حل صفحه بعد بر پایه monitor ارائه شده.

- ابتدا پس از مطالعه کد بخش های شروع خواندن و نوشتن (BeginRead , BeginWrite) ، توابع پایان خواندن و نوشتن (EndRead , EndWrite) را تکمیل کنید.
- آیا این راه حل اجازه خواندن همزمان بیشتر از ۱ خواننده را می دهد ؟ دلیل پاسخ خود را شرح دهید.
- به نظر شما برتری این پیاده سازی نسبت به حل آن با سمافور چیست ؟

Monitor Readers_Writers :

```
int Active_Readers = 0;  // Number of active readers
int Waiting_Readers = 0; // Number of waiting readers
int Active_Writers = 0;  // Number of active writers
int Waiting_Writers = 0; // Number of waiting writers

Condition Read = NULL; //condition variables for synchronization
Condition Write = NULL; //condition variables for synchronization
```

```
Void BeginRead()
{
        if (Active_Writers == 1 || WaitingWriters > 0)
        {
                Waiting_Readers += 1;
                Wait(Read);
                Waiting_Readers -= 1;
        }
        Active_Readers += 1;
        Signal(Read);
}
```

```
Void EndRead()
{
        ?
}
```

```
Void BeginWrite()
{
        if (Active_Writers == 1 || Active_Readers > 0)
        {
                Waiting_Writers += 1;
                wait(Write);
                Waiting_Writers -= 1;
        }
        Active_Writers = 1;
}
```

```
Void EndWrite()
{
        ?
}
```