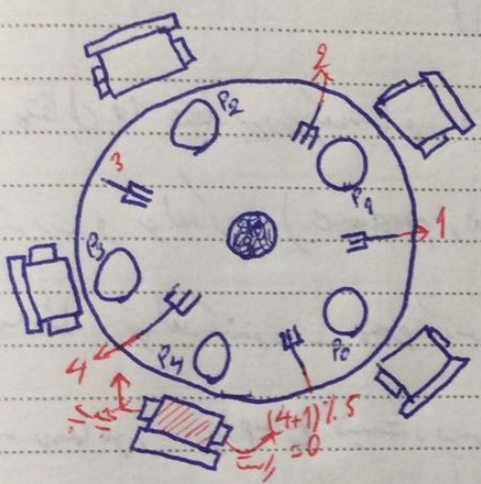
مسئله غذا خوردن فیلسوف ها:

5 فیلسوف داریم. زندگی هر فیلسوف از دو دوره متناوب خوردن و فکر کردن تشکیل شده است.

هر فیلسوف یک بشقاب ماکارونی دارد. بین هر جفت از بشقاب ها، یک چنگال قرار دارد.

هر فیلسوف برای خوردن از 2 چنگال مارنیش باید استفاده کند (عقل)



زمانی که هر فیلسوف گرسنه می شود، سعی می کند دو چنگال سمت چپ و راست خود را بردارد. اگر موفق شد، برای مدت غذای خود می خورد و سپس چنگال ها را از روی میز می گذارد و به فکر کردن ادامه می دهد.

مسئله تغذیه فیلسوفان علاوه بر استعمار متقابل (که در یکی از آن دو فیلسوف نمی توانند از یک چنگال استفاده کنند)، باید هوا بگرس نیز نیست و گرسنگی هم باشد.

هر فیلسوف ابتدا چنگال چپ و سپس چنگال راست را بر می دارد. بعد از تغذیه یک فیلسوف، دو چنگالی که استفاده می کرد را روی میز گذاشته و دیگران می توانند استفاده کنند.

```
semaphore room = 4
semaphore fork[5] = {1}
void philosopher (int i){
  while(TRUE){
    think()
    wait(room)
    wait(fork[i]) ---> left fork
    wait(fork[(i+1)%5]) ---> right fork
    eat() ---> critical-section()
    signal(fork[(i+1)%5])
    signal(fork[i])
    signal(room)
  }
}
```

پنجشنبه ۲۴
Thursday
۲۳ شهریور ۱۳۹۶
14 July 2016

سمافور room با مقدار اولیه 4، برای این است که اجازه ورود به بیش از 4 نفر داده نشود. اگر حداکثر 4 فیلسوف نشسته باشند، حداقل یک نفر به 2 چنگال دسترسی خواهد داشت

اگر از room استفاده نشود و اجازه ورود همزمان به هر 5 نفر داده شود، همه آن‌ها چنگال‌های چپ خود را برداشته و دیگر چنگال اضافی نمانده که کسی بتواند چنگال راست خود را بردارد و بنابراین بن‌بست رخ می‌دهد.

راه حل کتاب تننباوم:

```
#define LEFT  (i-1)%5
#define RIGHT (i+1)%5
typedef int semaphore
semaphore mutex=1
semaphore s[5]
int state[5]
```

در آرایه state وضعیت هر فیلسوف: فکر کردن = 0 | گرسنگی = 1 | خوردن = 2

جمعه ۲۵
Friday
۲۴ شهریور ۱۳۹۶
15 July 2016

این راحل بن‌بست ندارد.

یک فیلسوف در صورتی که هیچ یک از فیلسوفان چپ و راستش، در حال خوردن نباشند، می‌تواند غذا بخورد.

```
void philosopher(int i){
  while(true){
    think()
    take_forks(i)
    eat()
    put_forks(i)
  }}
```

```
void take_forks(int i){
  wait(mutex)
  state[i]=1
  test(i)
  signal(mutex)
  wait(s[i])
}
```

```
void put_forks(int i){
  wait(mutex)
  state[i]=0
  test(LEFT)
  test(RIGHT)
  signal(mutex)
}
```

```
void test(int i){
  if(state[i]==1
  && state[LEFT]!2
  && state[RIGHT]!2)
  {state[i]=2;
   signal(s[i])
  }
```