تمرین فصل 5

1.در مبحث سمافورها دیدیم که ما یک پردازنده داریم و کد های کتاب برای این حالت نوشته شده . حال با فرض اینکه ما 2 پردازنده داریم مسیله سمافور ها را بررسی کنید . توابع سمافوری signal() ,wait() را برای این حالت پیاده سازی کنید .

مساله را برای حالتی که 3 پردازنده داشته باشیم نیز بسط دهید و تنها تفاوت ها را ذکر کنید.

2.الگوریتم زیر برای ایجاد ناحیه ی بحرانی بین دو پردازندهP_i وP_j داده شده است.کدام گزینه در ارتباط با اجرای هم روند این دو پروسس درست است؟

Flag[j] = true;

While (flag[j] and turn = j) do

No-op;

Critical section

Turn = j;

Flag[j] = false;

3.سوال 5.6 قسمت مسایل.

4.در شکل 5.23 دلیل تعریف و استفاده از سمافور z چیست؟

5.دو نخ A و B قصد دارند که در طول اجرای خود ، در یک نقطه ی خاص با هم ملاقات داشته باشندو سپس به طور مستقل هر کدام اجرای خود را ادامه دهند. چگونه می توان با استفاده از سمافور به این ملاقات رسید؟ جاهای خالی شبه کد زیر را پر کنید:

Semaphore …

Void thread_A()	void thread_B()
{	{
…	…
}	}

6.تابع های سمافور wati() وSignal() می بایست به صورت اتمی اجرا شود تا مطمن باشیم که انحصار متقابل به درستی پیاده سازی می شود.فرض کنید که wait() به صورت اتمی نباشد سپس نشان دهید که انحصار متقابل نمی‌تواند رعایت شود.

7. امروز سیب زمینی سرخ کرده فرانسوی در منوی فست فود مفاخر دانشگاه صنعتی موجود است. سیب زمینی های سرخ کرده در یک سینی که میتواند به وسیله ی 7 قسمت پر شود ، فروخته می شوند. و هر کدام ازآنها با استفاده از یک جفت انبرک تحویل داده می شوند. فروشنده میتواند 7 قسمت را در یک زمان پر کند ، اما او به انبرک ها نیز نیاز دارد. هر دانشجو که به فست فود می آید دقیقا یک قسمت از سیب زمینی سرخ کرده را می خواهد.

راه حل های زیر را در نظر بگیرید. درست یا غلط بودن آن ها را مشخص کنید و در صورت نادرست بودن دلیل نادرستی را توضیح دهید.

Solution 1	Solution 2

semaphore tongs = 1, tray = 0;
semaphore fries = 0;
shared int amount = 0;
void aramark()
{
  while (1) {
  down(&tray);
  down(&tongs);
  add_french_fries();
  for (i = 0; i < 7; i++) {
  up(&fries);
  amount++;
  }
  up(&tongs);
  }
 	
}
	semaphore tongs = 1, tray = 1;
	shared int amount = 0;
	void aramark()
	{
	  while (1) {
	  down(&tray);
	  down(&tongs);
	  add_french_fries();
	  amount = 7;
	  up(&tongs);
	  }
	
	}

void student()
{
  down(&fries);
  down(&tongs)
  take_french_fries();
  amount--;
  if (amount == 0) {
  up(&tray);
  }
  up(&tongs);
 	
}
	void student()
	{
	  while (1) {
	  down(&tongs);
	  if (amount == 0) {
	  up(&tongs);
	  }
	  else {
	  take_french_fries();
	  amount--;
	  if (amount == 0) {
	  up(&tray);
	  }
	  up(&tongs);
	  break;
	  }
	  }
	
	}