بسمه تعالی

نمونه تمرینات دو فصل همزمانی و بن بست

۱- شبه کد زیر برای حل مسئله تولید کننده و مصرف کننده با بافر نامحدود داده شده است.

الف) عملکرد این شبه کد را برای سه اصل رعایت انحصار متقابل، انتظار محدود و شرط پیشرفت بررسی کنید.

ب) اگر جای دو دستور wait(s) و wait(n) را عوض کنیم، آیا در عملکرد شبه کد برای رعایت سه اصل فوق مشکلی ایجاد می شود؟ توضیح دهید.

Program Producer Consumer Int n=0 ; s=1	

Void Producer (Void){
Int item;
While (True){
Produce_item(&item);
Wait (s);
Enter_Item(Item);
Signal (s);
Signal(n) ;
}
}
	Void Consumer (Void){
	Int item;
	While (True){
	Wait (N);
	Wait(S);
	
	Remove_Item(&Item);
	Signal (S);
	Consum_Item(Item);
	}
	}

۲- شبه کد زیر برای ایجاد انحصار متقابل بین تعداد دلخواهی فرایند پیشنهاد شده است. عملکرد این شبه کد را برای سه اصل رعایت انحصار متقابل، انتظار محدود و شرط پیشرفت بررسی کنید.

```
While(True) {
While (lock!=0);
Lock=1;
Critical_section();
Lock=0;
 Noncritical_Section();
}
```

۳- نسخه ای از الگوریتم نانوایی به صورت زیر برای حل مسئله ناحیه بحرانی داده شده است. ضمن تشریح عملکرد این الگوریتم، مشکل آن را توضیح دهید.

```
boolean choosing[n];
int number[n];
while (true) {
   choosing[i] = true;
   number[i] = 1 + getmax(number[], n);
   choosing[i] = false;
   for (int j = 0; j < n; j++){
     while (choosing[j]) { };
     while ((number[j] != 0) && (number[j],j) < (number[i],i)) { };
   }
   /* critical section */;
   number [i] = 0;
   /* remainder */;
}
```

۴- برنامه زیر یک راه حل نرم افزاری برای مسئله انحصار متقابل دو فرایند است. مثال نقصی پیدا کنید که نادرستی این راه حل را نشان دهد. پیشنهاد خود برای حل آن مشکل چیست؟

```
Boolean blocked [2];
int turn;
Void p (int id)  {
      While (true)   {
Blocked [id]=true;
While (turn !=id)
  {
        While (blocked[1-id])
  {
turn=id
        }
    }
/* critical section*/
Blocked[id]=false;
/* remainder*/
         }
      }
Void main( )
     {
Blocked[0]=false ;  blocked[1]=false;  turn=0;
Parbegin (p(0) , p(1));
  }
```

۵- حل مسئله خوانندگان – نویسندگان با استفاده سمافور و تبادل پیام به روش های زیر پیشنهاد شده است. ضمن تشریح عملکرد این الگوریتم ها، مشکلاتی همچون بن بست، گرسنگی و انتظار نامحدود را در این دو الگوریتم بررسی نمایید.

اولویت خوانندگان

```
/* program readersandwriters */
int readcount;
semaphore x = 1,wsem = 1;
void reader()
{
    while (true){
     semWait (x);
     readcount++;
     if(readcount == 1)
         semWait (wsem);
     semSignal (x);
     READUNIT();
     semWait (x);
     readcount;
     if(readcount == 0)
         semSignal (wsem);
     semSignal (x);
    }
}
void writer()
{
    while (true){
     semWait (wsem);
     WRITEUNIT();
     semSignal (wsem);
    }
}

void main()
{
    readcount = 0;
    parbegin (reader,writer);
}
```

اولویت نویسندگان

```
/* program readersandwriters */
int readcount,writecount;
void reader()
{
    while (true){
      semWait (z);
          semWait (rsem);
               semWait (x);
                     readcount++;
                     if (readcount == 1)
                           semWait (wsem);
                     semSignal (x);
               semSignal (rsem);
          semSignal (z);
          READUNIT();
          semWait (x);
               readcount--;
               if (readcount == 0) semSignal (wsem);
          semSignal (x);
    }
}
void writer ()
{
    while (true){
      semWait (y);
          writecount++;
          if (writecount == 1)
               semWait (rsem);
      semSignal (y);
      semWait (wsem);
      WRITEUNIT();
      semSignal (wsem);
      semWait (y);
          writecount;
          if (writecount == 0) semSignal (rsem);
      semSignal (y);
    }
}
void main()
{
    readcount = writecount = 0;
    parbegin (reader, writer);
}
```

تبادل پیام

```
void reader(int i)
{
   message rmsg;
      while (true) {
         rmsg = i;
         send (readrequest, rmsg);
         receive (mbox[i], rmsg);
         READUNIT ();
         rmsg = i;
         send (finished, rmsg);
      }
}
void writer(int j)
{
   message rmsg;
   while(true) {
      rmsg = j;
      send (writerequest, rmsg);
      receive (mbox[j], rmsg);
      WRITEUNIT ();
      rmsg = j;
      send (finished, rmsg);
   }
}
```

```
void controller()
{
   while (true)
   {
      if (count > 0) {
         if (!empty (finished)) {
            receive (finished, msg);
            count++;
         }
         else if (!empty (writerequest)) {
            receive (writerequest, msg);
            writer_id = msg.id;
            count = count - 100;
         }
         else if (!empty (readrequest)) {
            receive (readrequest, msg);
            count--;
            send (msg.id, "OK");
         }
      }
      if (count == 0) {
         send (writer_id, "OK");
         receive (finished, msg);
         count = 100;
      }
      while (count < 0) {
         receive (finished, msg);
         count++;
      }
   }
}
```

۶- روش تناوب صریح یا Strict Alternation در حل مشکل ناحیه بحرانی را تشریح کرده و عیب آن را با تعریف یک سناریو بیان نمایید.

۷- مشکل وارونگی اولویت یا Priority Inversion Problem در حل مشکل ناحیه بحرانی چیست؟ تشریح نمایید.

۸- دو فرآیند زیر به صورت همروند اجرا می شوند، تمام دستورات این فرآیند ها یکپارچه هستند و مقدار اولیه متغیرها ی x و y صفر و مقدار سمافور دودویی Mutex یک است. بعد از اجرای کامل دو فرآیند متغیرهای x و y چه مقادیری می‌توانند داشته باشند. ۳ حالت را بررسی کنید.

P_1	P_2
X=1	X=x+2
Wait (mutex)	Wait (mutex)
Y=y+x	Y=y-1
X=2	X=x-y
signal (mutex)	signal (mutex)

۹- در صورتی که دو کد زیر برای حل مسئله کلاسیک همزمانی بافر محدود با ظرفیت n باشد و اگر مقدار سمافور های x ،y، z به ترتیب برابر n،1 ،0 باشد آن گاه در صورت اجرای همروند دو فرآیند تولید کننده و مصرف کننده چه اتفاقی می افتد؟ تعداد تولید و مصرف قطعات را بررسی کنید.

مصرف کننده	تولید کننده

While (1) {
Wait(z)
Wait(y)
حذف قطعه از بافر
Signal(y)
استفاده از قطعه
}
	While (1) {
	تولید قطعه
	Wait(x)
	Wait(y)
	اضافه کردن قطعه به بافر
	Signal(y)
	Signal(z)
	}

۱۰- در سیستمی چهار منبع Tape=6، Plotter=3 ، Printer=4 و CD ROM=2 موجود است که وضعیت برخی از ماتریس های آن به صورت زیر است. اگر دو فرایند A و D هر کدام درخواست منابع Tape=2 و Printer=1 باید عمل تخصیص انجام شود یا خیر؟

Process	Tape drives	Plotters	Printers	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Printers	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still needed

۱۱- این پروتکل پیشگیری از بن بست را در نظر بگیرید: **" هر فرایند قادر است منابعی را درخواست کند که شماره آن منبع از تمامی شماره های منابع در اختیار فرایند بزرگتر است."** این پروتکل کدام شرط لازم بن بست را نفی می کند؟ چرا؟

۱۲- در سیستمی n فرایند همزمان و 4n واحد از یک منبع قابل استفاده مجدد وجود دارد. اگر هر فرایند درخواست x واحد ($x \leq 4n$) از منبع را نماید، برای اینکه در این سیستم بن بست رخ ندهد، حداکثر مقدار x باید چقدر باشد؟

۱۳- سیستمی با مجموع ۱۵۰ واحد حافظه به صورت زیر به سه فرایند تخصیص داده شده است.

فرایند	حداکثر نیاز	تخصیص داده شده
A	70	45
B	60	40
C	60	15

در صورتیکه فرایند D با حداکثر حافظه مورد نیاز 60 و مقدار نیاز اولیه 35 واحد وارد سیستم شود، آنگاه با بکارگیری الگوریتم بانکداران، کدام گزینه درست است؟

الف) قبل از ورود فرایندD سیستم در بن بست قرار دارد.

ب) بعد از ورود فرایند D احتمال وقوع بن بست وجود دارد.

ج) بعد از ورود فریند D احتمال وقوع بن بست وجود ندارد.

د) با این اطلاعات نمی توان اظهار نظر کرد

۱۴- یکی از شرایط پیشگیری از بن بست، نقض شرط " گرفتن و منتظر ماندن" است. دو پروتکل برای تحقق آن پیشنهاد داده و معایب آنها را بیان نمایید.

۱۵- سیستمی را با ۵ فرایند p_0 تا p_4 و سه نوع منبع A=7، B=2 و C=6 در نظر بگیرید. اگر حالت تخصیص منابع در لحظه T_0 بصورت زیر باشد، و فرایند P2 درخواست (0,0,1) داشته باشد، بررسی کنید که با تخصیص جدید سیستم در حالت امن است یا خیر؟

	Allocation	*Request*	*Available*
	A B C	*A B C*	*A B C*
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	