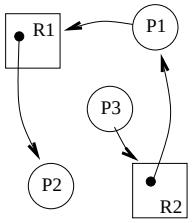
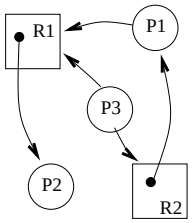
Esercizio 1

Si considerino i seguenti grafi di assegnazione in cui le frecce tratteggiate indicano potenziali richieste future. Indicare, per ogni caso, se il sistema è in stallo (deadlock) e se lo stato è *s*icuro per l'algoritmo del banchiere motivando accuratamente la risposta.

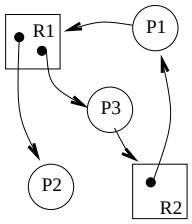
1.



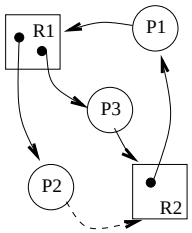
2.



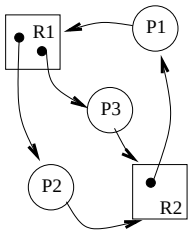
3.



4.



5.



Esercizio 2

La funzione `Bonifico(i,j,c)` esegue un bonifico bancario di `c` euro dal conto `conto`[`i`] al conto `conto`[`j`], nel caso ci sia disponibilità, altrimenti ritorna `false`.

```
boolean Bonifico(int i, int j, int c);
    if conto[i] >= c then
        conto[i] = conto[i]-c;
        conto[j] = conto[j]+c;
        return(true);
    else
        return(false);
```

Ogni terminale della banca puo' eseguire bonifici sugli stessi conti in modo concorrente

1. Illustrare almeno due problemi di interferenza causati dall'esecuzione concorrente di due bonifici;

2. Come si possono risolvere i problemi sopra indicati utilizzando i semafori? (aggiungere le relative wait/P e signal/V a fianco al codice indicando l'inizializzazione dei semafori)

3. La soluzione proposta al punto precedente potrebbe essere utilizzata anche nel caso in cui i bonifici eseguiti in modo concorrente siano piu' di due? Discutere.

Esercizio 3

Un metodo sicuro per evitare interferenze tra thread è quello di NON permettere che essi condividano memoria. Perchè questa soluzione risulta essere, nella maggior parte dei casi, troppo restrittiva? Discutere e illustrare tramite esempi.

Esercizio 4

In un parcheggio viene utilizzato un sistema automatico per conteggiare il numero di auto presenti. Tale numero non deve superare la capienza massima `MAX` del parcheggio. Per questo scopo vengono impiegati due sbarre automatiche, una all'ingresso e una all'uscita, che fanno passare una sola auto alla volta e mantengono il conteggio tramite i seguenti due processi:

```
process Ingresso {
  while(1) {
     <richesta ingresso>
     presenti.incrementa()
     <apri la sbarra ingresso>
  }
}
```

```
Process Uscita {
   while(1) {
      <pagamento e richiesta uscita>
      presenti.decrementa()
      <apri la sbarra uscita>
    }
 }
```

```
monitor presenti{
    int n_presenti = 0;
    condition attendi;

    void incrementa() {

    }
}
```

```
    void decrementa() {

    }
```

Completare il codice del monitor `presenti` illustrandone brevemente il funzionamento. Gestire correttamente la situazione in cui il parcheggio sia pieno: in tale caso `incrementa()` blocca il processo finchè non si libera un posto.

Esercizio 5

Spesso i Sistemi Operativi moderni necessitano di hardware "speciale" per migliorare la gestione delle risorse. Dare un esempio illustrando le fasi in cui il Sistema trae vantaggio dalla presenza di tale hardware.

Esercizio 6

In un sistema, sono pronti per essere eseguiti i 2 processi P1, P2. L'algoritmo di scheduling è SJF (Shortest Job First). P1 e P2 utilizzeranno la CPU per 15 e 3 unita' di tempo, rispettivamente. All'istante 5 termina l'attivita' di I/O il processo P3 che necessita della CPU per 2 unita' di tempo.

1. Illustrare lo scheduling (scrivere CPU per il processo in esecuzione P per i processi pronti e I/O per quelli in attesa di I/O)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				

2. Che tipo di problema si presenta nell'esecuzione riportata sopra? Descrivere una variante di SJF che risolva tale problema mostrandone il relativo scheduling:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				

Esercizio 7

Si consideri una memoria fisica (inizialmente vuota) di 3 frame. Scegliere una sequenza di accesso alle pagine virtuali 1,2,3,4 tale per cui FIFO ed LRU si comportino esattamente allo stesso modo e funzionino peggio della strategia ottima.

1. sostituzione First In First Out (FIFO),

Spiegare brevemente:

Frame 1													
Frame 2													
Frame 3													
Page Fault													

2. LRU (Least Recently Used),

Spiegare brevemente:

Frame 1													
Frame 2													
Frame 3													
Page Fault													

3. Sostituzione Ottima

Spiegare brevemente

Frame 1													
Frame 2													
Frame 3													
Page Fault													

4. Perchè, in questo caso, la strategia ottima funziona meglio e FIFO=LRU?