Esempio di domande della prima provetta

Si consideri la seguente procedura in C: **(1.2)**

```
float calcola(float a, float b, float c, float d, float e, float f)
{
   float x, y, z, t;
   x=a*sqrt(b) ;  y=c*c*d; z=y+e ; t=y*f ; z =x*z ; t=x*t*t ;
   return(z*t) ;
}
```

Si supponga di voler eseguire la procedura in concorrenza. Si scriva dapprima il grafo delle precedenze. Si implementi poi il grafo usando un linguaggio di programmazione **concorrente**. Si implementi poi il grafo usando

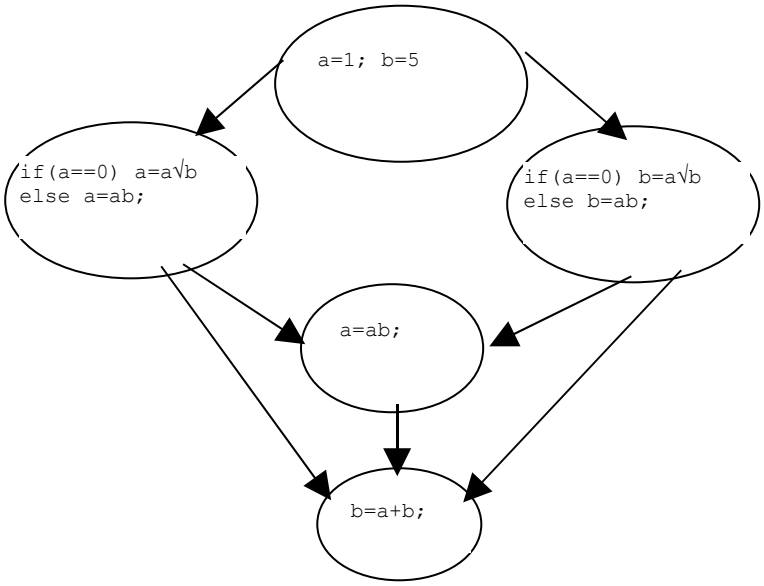
- le primitive (p=fork(processo), join(p))
- le primitive (cobegin, coend).

Si supponga che una architettura di un calcolatore offra al programmatore una istruzione **atomica** scambia(a,b), che realizza la seguente funzione:

```
scambia(short &a, short &b){
   short t;
   t = *a; *a = *b; *b = t;
}
```

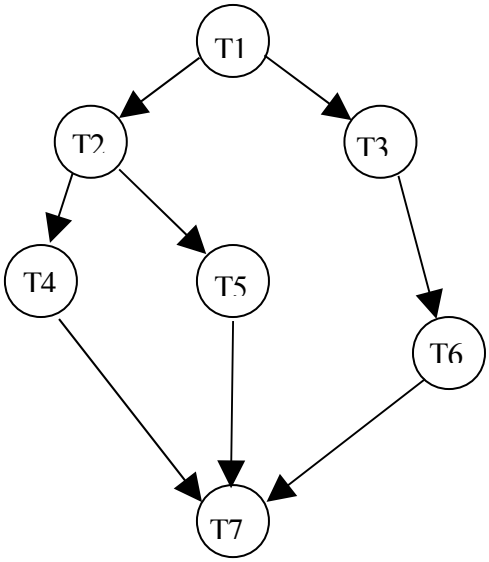
Come si puo' utilizzare questa istruzione per proteggere una sezione critica ? **(1.4)**

Scrivere lo pseudocodice di un programma concorrente che implementa il seguente grafo parzialmente ordinato in un S.O. dotato delle chiamate di sistema `p=fork()` e `join(p)`.



Il grafo del punto precedente è determinato? Se non lo è, mostrare una sequenza di esecuzione che dimostra la non determinezza.

Sia dato un processo concorrente multitasking descritto dal seguente grafo



dove i singoli task sono descritti da:
T1: a=b=c=0; x=y=2;
T2: a=b+1; b=c+1; z=3;
T3: b=x+y; x=1; z=1;
T4: x=a+b; c=2;
T5: if(x==a) y=b; else z=x ;
T6: if(x==1) x=a+b; else x=0;
T7: print(a,b,c,x,y,z,c);

Il processo concorrente descritto dal grafo di cui al punto 6 e' determinato o no? . Dire se esistono almeno due task determinati e, se si, dire quali sono.

. Si supponga che vi siano 3 processi concorrenti e 10 istanze di un unico tipo di risorse. La situazione e' descritta dalla seguente tabella:

	Usate	Richieste
Proc.1	3	3
Proc.2	2	5
Proc.3	2	7

Questo stato è in stallo? Si supponga che il terzo processo chieda una ulteriore risorsa. Lo stato che si ottiene è in stallo?

Descrivere i servizi fondamentali offerti dai Sistemi Operativi **[0.8 punto]**

Descrivere brevemente il nucleo dei Sistemi Operativi (max ½ pagina) **[2.5 punto]**

Cos'è e com'è strutturato il Process Control Block di un Sistema Operativo? **[0.8 punto]**

Quali sono i possibili vantaggi e svantaggi dello 'spooling'? **[0.8 punto]**

Che differenza c'è tra 'programma' e 'processo'? **[0.3 punto]**

Che relazione c'è tra 'time sharing', 'multiprogrammazione' e 'multiutenza'?**[0.7 punto]**

Descrivere l'organizzazione del nucleo di Unix **[0.8 punto]**

Descrivere le funzioni generali del nucleo di un Sistema Operativo (non microkernel).**(0.5)**

Descrivere alcune code d'attesa presenti in un Sistema Operativo. **(0.5)**

Si consideri il codice eseguibile di un Sistema Operativo. **(0.5)**
a. Cosa succede se un utente lo esegue come processo d'utente?
b. A cosa potrebbe servire?

Che differenza c'e' tra le chiamate di sistema e le chiamate a procedura? Quando si usano le chiamate di sistema e quando le chiamate a procedura? **[1 punto]**

Descrivere brevemente due costrutti linguistici per generare la concorrenza. Hanno la stessa potenza (cioè tutti i processi concorrenti realizzati con uno di essi possono programmati anche con l'altro)? **(2pt.)**

Cosa sono e a cosa servono le primitive semaforiche? Dare una brevissima descrizione del loro funzionamento.

Definire brevemente le differenze tra un Sistema Operativo 'interrompibile' ed uno 'non interrompibile'.

Descrivere brevemente alcune organizzazioni interne di Sistemi Operativi con i relativi vantaggi/svantaggi (es.:monolitico, a livelli…)

Un codice eseguibile può essere considerato una risorsa del Sistema Operativo? Se sì, caratterizzarla in termini di condivisibilità, sottraibilità e riutilizzabilità.

Si consideri i seguente pseudocodici di due thread:

```
PC1(){
  while(true){
    el=produci();
    down(mutex);
    down(pieno);
    inserisci(el);
    up(vuoto);
    up(mutex);
  }
}

PC2(){
  while(true){
    down(mutex);
    down(vuoto);
    val=rimuovi();
    up(pieno);
    up(mutex);
    consuma(val);
  }
}
```

Si tratta di due thread concorrenti che si passano i dati attraverso un buffer condiviso con il

meccanismo del produttore/consumatore. Il semaforo ‘mutex’ è inizializzato ad 1, i semafori ‘pieno’ e ‘vuoto’ sono inizializzati a 0 e 10 (la dimensione del buffer) rispettivamente. Discutere il funzionamento dei due thread concorrenti e, nel caso ci sia un errore di programmazione o di inizializzazione, proporre una soluzione. **(4pt)**

Lo stesso problema si vuole risolvere con i seguenti due thread:

```
PC1(){
  while(true){
    el=produci();
    down(pieno);
    down(mutex);
    inserisci(el);
    up(vuoto);
    up(mutex);
  }
}
PC2(){
  while(true){
    down(vuoto);
    down(mutex);
    val=rimuovi();
    up(pieno);
    up(mutex);
    consuma(val);
  }
}
```

I semafori siano inizializzati nello stesso modo. Discutere il funzionamento dei due thread concorrenti e, nel caso ci sia un errore di programmazione o di inizializzazione, proporre una soluzione.

Si scriva lo pseudocodice di tre thread concorrenti (il dettaglio degli pseudocodici sia del livello di quelli usati al punto 4.) che risolvono il seguente problema:
Si vogliono sommare gli elementi di un array ‘buf’ di 20 elementi, cioè fare l’operazione buf[0]+buf[1]+…+buf[19]. L’operazione viene fatta però in modo concorrente, in modo tale cioè che il primo thread somma gli elementi pari dell’array, mentre il secondo somma gli elementi dispari. Il terzo thread somma i risultati dei primi due. Si sincronizzi l’operazione del terzo thread con un semaforo, ‘sincro’. Come deve essere inizializzato il semaforo?
(7pt)

Si consideri una variante del problema dei lettori scrittori con i seguenti due lettori:

```
lettore1(){
  while(true){
    down(mutex);
    nrlettori++;
    if(nrlettori==1)
      printf("primo lettore");
    up(mutex);
    leggi_archivio();
    down(mutex);
    nrlettori--;
    if(nrlettori==0)
      printf("ultimolettore");
    up(mutex);
  }
}
lettore2(){
  while(true){
    down(mutex);
    nrlettori++;
    if(nrlettori==1)
      printf("primolettore");
```

```
    up(mutex);
    leggi_archivio();
    down(mutex);
    nrlettori--;
    if(nrlettori==0)
    printf("ultimo lettore");
    up(mutex);
  }
}
```

Questa variante si vuole scrivere su una piattaforma che non ha le primitive semaforiche. Come si possono proteggere le sezioni critiche in questo caso? Come si possono modificare i codici riportati sopra?

Descrivere i servizi fondamentali offerti dai Sistemi Operativi **[0.8 punto]**

Descrivere brevemente il nucleo dei Sistemi Operativi (max ½ pagina) **[2.5 punto]**

Cos'è e com'è strutturato il Process Control Block di un Sistema Operativo? **[0.8 punto]**

Quali sono i possibili vantaggi e svantaggi dello 'spooling'? **[0.8 punto]**

Che differenza c'è tra 'programma' e 'processo'? **[0.3 punto]**

Che relazione c'è tra 'time sharing', 'multiprogrammazione' e 'multiutenza'?**[0.7 punto]**

Si faccia un breve confronto tra ognuno di questi termini: (1.5pt)

1. Sistema Operativo dedicato
2. Sistema Operativo interattivo
3. Sistema Operativo Time Sharing

Si valutino le architetture monolitiche, a strati e microkernel con riferimento a:(1.5pt)

1. Efficienza
2. Robustezza
3. Espandibilità

Si rifletta sulla durata del quanto di Time Sharing, cioè la durata degli intervalli di context switch. (1.5pt)

- Che effetto avrebbe impostare il quanto ad un valore molto grande?
- Che effetto avrebbe impostare il quanto ad un valore molto piccolo?

Il coefficiente di utilizzazione della CPU indica (segnare le risposte vere e quelle false)(1pt)

1. Il numero medio di processi nella CPU V F
2. Il numero di processi nel sistema al tempo t V F
3. Il numero medio di risorse occupate V F
4. La probabilità che la CPU sia occupata V F

Segnare quali delle seguenti affermazioni sono vere e quali quelle false (1pt)

3. Le procedure di gestione degli interrupt vengono eseguite sempre in modalità utente V F
4. Una istruzione privilegiata può essere eseguita sono in modalità Utente V F
5. L'istruzione che disabilita gli interrupt non è privilegiata V F

Il context switch fra thread dello stesso processo è più efficiente se i thread sono implementati a livello utente invece che a livello kernel V F

Si consideri un semplice Sistema Operativo senza interazioni di I/O e senza semafori. Si vuole mantenere l'utilizzazione della CPU maggiore dell'80%. Se la frequenza degli arrivi è di 10 processi/secondo, indicare: (2pt)
1. Qual'è la massima frequenza di terminazione? (processi/secondo)
2. Qual'è il minimo tempo d'attesa dei processi? (secondi)
[si ricordano le formule fondamentali della coda M/M/1:
● coefficiente di utilizzazione CPU=$\rho=\lambda/\mu$ dove λ è la frequenza degli arrivi di processi (nr.arrivi/secondo) e μ la frequenza della terminazione delle esecuzioni dei processi (nr.terminazioni/secondo).
● numero medio di processi nel sistema= $\rho/(1-\rho)$
● tempo d'attesa dei processi = $1/(\mu-\lambda)$]

Si supponga di essere in un ambiente di programmazione a Thread. Scrivere lo pseudocodice del produttore/consumatore usando l'algoritmo della alternanza stretta (per la mutua esclusione) e due semafori (per la sincronizzazione).(3pt)

8) Si supponga di eseguire i seguenti due Thread.(5pt)

```
Public class Inc() extends Threads{
  System.out.println("i="+i);
  i++;
}
Public class Dec() extends Threads{
  i--;
  System.out.println("i="+i);
}
```

Si voglia generare l'uscita 3 3 3 3 3 3 .....
Modificare il programma usando semafori per avere l'uscita desiderata. Come dovrebbero essere inizializzati i semafori e la variabile `i`?

Si abbiano i seguenti due thread concorrenti: (5pt)

```
PC1(){
  while(true){
   el=rimuovi();
   consuma(el);
   dato=produci();
   inserisci(dato);
  }
}
PC2(){
  while(true){
   val=rimuovi();
   consuma(val);
   elto=produci();
   inserisci(elto);
  }
}
```

Si tratta di due thread che si scambiano un dato: una variabile condivisa `var`, inizializzata ad un certo valore, viene letta dalla procedura `rimuovi()` e scritta dalla procedura `inserisci()`.
Ad esempio, supponiamo che `var` sia inizializzata a 5 e che parta per primo il Thread PC1. Allora PC1 legge 5 e scrive un altro valore, ad esempio 7, poi si sospende per lasciare al Thread PC2 di leggere il valore 7 e di scrivere un altro valore nella variabile.

Sostanzialmente i due Thread sono chiamati nella sequenza PC1-PC2-PC1-PC2 e cosi' via. Sincronizzazione i due Thread con due semafori e indicare come devono essere inizializzati.

Come si potrebbe scrivere questa applicazione usando due `coroutine` invece che due Threads? (4pt)

1. 2pt)Definire brevemente le differenze tra un Sistema Operativo `pre-emptive` ed uno
   `non pre-emptive`?

2. 2pt) Un codice eseguibile può essere considerato una risorsa del Sistema Operativo? Se
   sì, caratterizzarla in termini di `condivisibilità, sottraibilità e riutilizzabilità`.

3. 2pt) Descrivere e commentare brevemente le differenze tra una chiamata di sistema e una
   chiamata di programma.

4. 3pt)Descrivere alcuni aspetti fondamentali -a scelta- del file system di Unix.

5. 3pt)Descrivere brevemente due costrutti linguistici per generare la concorrenza. Hanno la stessa potenza (cioè tutti i processi concorrenti realizzati con uno di essi possono programmati anche con l'altro)?

6. 6pt) Si consideri il seguente pseudocodice:

```
PC1(){
       while(true){
              el=produci();
              down(mutex);
              down(pieno);
              inserisci(el);
              up(vuoto);
              up(mutex);
       }
}
```

```
PC2(){
while(true){
       down(mutex);
       down(vuoto);
       val=rimuovi();
       up(pieno);
       up(mutex);
       consuma(val);
}
}
```

Si tratta di due thread concorrenti che si passano i dati attraverso un buffer condiviso con il meccanismo del produttore/consumatore. Il semaforo 'mutex' è inizializzato ad 1, i semafori 'pieno' e 'vuoto' sono inizializzati a 0 e 10 (la dimensione del buffer) rispettivamente. Discutere il funzionamento dei due thread concorrenti e, nel caso ci sia un errore di programmazione o di inizializzazione, proporre una soluzione.

7. 6pt) Lo stesso problema descritto al punto 8) si vuole risolvere con i seguenti due thread:

```
PC1(){
       while(true){
              el=produci();
              down(pieno);
              down(mutex);
              inserisci(el);
              up(vuoto);
              up(mutex);
       }
}
```

```
PC2(){
while(true){
       down(vuoto);
       down(mutex);
       val=rimuovi();
       up(pieno);
       up(mutex);
       consuma(val);
}
}
```

I semafori siano inizializzati nello stesso modo. Discutere il funzionamento dei due thread concorrenti e, nel caso ci sia un errore di programmazione o di inizializzazione, proporre una soluzione.

Si scriva lo pseudocodice di tre thread concorrenti (il dettaglio degli pseudocodici sia del livello di quelli usati al punto 6.) che risolvono il seguente problema:
"si vogliono sommare gli elementi di un array 'buf' di 20 elementi, cioè fare l'operazione buf[0]+buf[1]+…+buf[19]. L'operazione viene fatta però in modo concorrente, in modo tale cioè che il primo thread somma gli elementi pari dell'array, mentre il secondo somma gli elementi dispari. Il terzo thread somma i risultati dei primi due. Si sincronizzi l'operazione del terzo thread con un semaforo, 'sincro'. Come deve essere inizializzato il semaforo?"

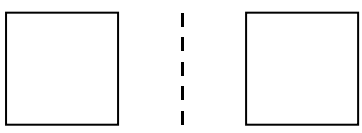
8. 1)Definire brevemente le differenze tra un Sistema Operativo `pre-emptive` ed uno
   `non pre-emptive`?

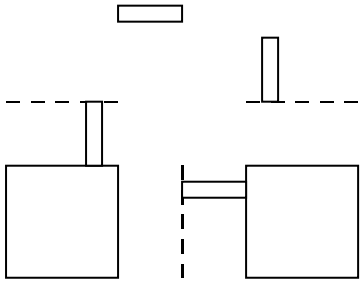
9. 3)Descrivere brevemente alcune organizzazioni interne di Sistemi Operativi con i relativi vantaggi/svantaggi (es.:monolitico, a livelli?)

10. 1) Un codice eseguibile può essere considerato una risorsa del Sistema Operativo? Se
    sì, caratterizzarla in termini di `condivisibilità, sottraibilità e riutilizzabilità`.

11. **1) Descrivere e commentare brevemente le differenze tra una chiamata di sistema e una**
    chiamata di programma.

12. Si consideri un incrocio stradale, dove vale la regola della precedenza a destra.

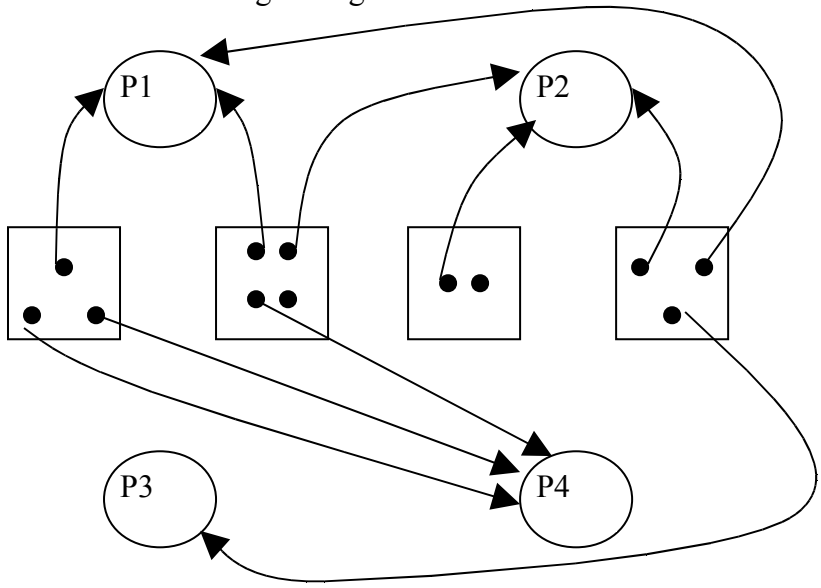




Usando i grafi di allocazione risorse, il candidato:

a) (2pt) decida cosa può essere rappresentato come processo concorrente e cosa come risorsa
b) (2pt) rappresenti una situazione di stallo. Usando sempre i grafi di allocazione risorse, il candidato descriva anche una situazione di non stallo

13. Si consideri il seguente grafo di allocazione risorse:



Il candidato risponda alle seguenti due domande:

1. (4pt) Supponendo che i processi richiedano le seguente risorse:
   2. P1 richiede (2,2,2,2)
   3. P2 richiede (1,1,1,1)
   4. P3 richiede (0,1,0,1)
   5. P4 richiede (0,1,0,0)

   si valuti, usando l'algoritmo di rilevazione dello stallo, se il sistema è in stallo o meno.

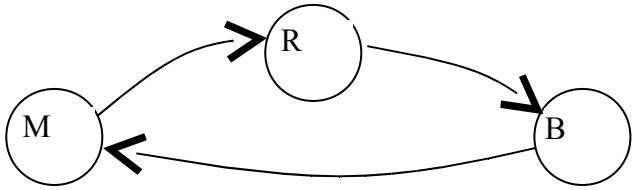
1. Descrivere la funzione del Process Control Block di un Sistema Operativo e le informazioni in esso contenute. **(1)**

2. Descrivere le principali differenza tra thread e processo. In che modalità (utente/sistema) può eseguire un processo? In che modalità può eseguire un thread?

   (2)

3. Che differenza c'è tra Chiamata di sistema e chiamata a procedura? Perché un Sistema Operativo non usa solo Chiamate di sistema o solo Chiamate a Procedura ma usa entrambe? **(2)**

4. Il diagramma degli stati di un processo che esegue in un Sistema Operativo può essere semplificato come segue:



   14. Quali possono essere gli stati M, R e B? Definire gli stati e rispondere alle seg. domande:
   15. Perché non c'è una freccia dallo stato M allo stato B?
   16. Perché non c'è una freccia dallo stato B allo stato R?
   17. **Perché non c'è una freccia dallo stato R allo stato M?**

c) Descrivere brevemente due costrutti linguistici per generare la concorrenza. Hanno la stessa potenza (cioè tutti i processi concorrenti realizzati con uno di essi possono programmati anche con l'altro)?
(2pt.)

d) Cosa sono e a cosa servono le primitive semaforiche? Dare una brevissima descrizione del loro funzionamento. **(2pt)**

e) (1.5) Si consideri il seguente pseudocodice:

```
PC1(){
       while(true){
              el=produci();
              down(mutex);
              down(pieno);
              inserisci(el);
              up(vuoto);
              up(mutex);
       }
}
```

```
PC2(){
       while(true){
              down(mutex);
              down(vuoto);
              val=rimuovi();
              up(pieno);
              up(mutex);
              consuma(val);
       }
}
```

Si tratta di due thread concorrenti che si passano i dati attraverso un buffer condiviso con il meccanismo del produttore/consumatore. Il semaforo 'mutex' è inizializzato ad 1, i semafori 'pieno' e 'vuoto' sono inizializzati a 0 e 10 (la dimensione del buffer) rispettivamente. Discutere il funzionamento dei due thread concorrenti e, nel caso ci sia un errore di programmazione o di inizializzazione, proporre una soluzione. **(4pt)**

Lo stesso problema si vuole risolvere con i seguenti due thread:

```
PC1(){
       while(true){
              el=produci();
              down(pieno);
              down(mutex);
              inserisci(el);
              up(vuoto);
              up(mutex);
       }
}
```

```
PC2(){
       while(true){
              down(vuoto);
              down(mutex);
              val=rimuovi();
              up(pieno);
              up(mutex);
              consuma(val);
       }
}
```

```
        consuma(val);                    }
}
```

I semafori siano inizializzati nello stesso modo. Discutere il funzionamento dei due thread concorrenti e, nel caso ci sia un errore di programmazione o di inizializzazione, proporre una soluzione. **(4pt)**

f) Si scriva lo pseudocodice di tre thread concorrenti (il dettaglio degli pseudocodici sia del livello di quelli usati al punto 4.) che risolvono il seguente problema:
si vogliono sommare gli elementi di un array 'buf' di 20 elementi, cioè fare l'operazione buf[0]+buf[1]+…+buf[19]. L'operazione viene fatta però in modo concorrente, in modo tale cioè che il primo thread somma gli elementi pari dell'array, mentre il secondo somma gli elementi dispari. Il terzo thread somma i risultati dei primi due. Si sincronizzi l'operazione del terzo thread con un semaforo, 'sincro'. Come deve essere inizializzato il semaforo?
(7pt)

g) Si consideri un Sistema Operativo dotato di tre risorse, ciascuna presente in un certo numero di istanze.
In un certo istante, la situazione del sistema e' la seguente:
le risorse allocate sono descritte dalla matrice P e quelle richieste dalla matrice Q.

P ==

1	2	0
1	2	3
3	2	1
0	0	0

Q ==

0	0	0
0	2	0
0	2	1
2	1	0

Qual'è il numero minimo di istanze iniziali delle risorse tali che il sistema non sia in stallo?
(5pt)

In una biblioteca c'è un calcolatore dotato di un unico terminale mediante il quale gli utenti fanno ricerche sulla disponibilità e collocazione dei libri. Se un nuovo utente arriva per consultare il terminale in media ogni 3 minuti, ed il tempo medio di attesa che gli utenti aspettano prima di usare il terminale è di 8 minuti, qual è il numero medio di utenti che aspetta in coda?

Si consideri un S.O. non pre-emptive, senza iterazioni di I/O. Supponendo che il tempo medio di esecuzione dei processi sia di 0.2 secondi, trovare la frequenza massima di arrivo dei processi tale che il tempo di attesa nel sistema sia minore di 0.5 secondi.

Si consideri un S.O. non pre-emptive, senza iterazioni di I/O. Si supponga che da alcune misure risulti che il tempo medio di esecuzione dei processi sia pari a 100/(frequenza clock CPU in Mhz). Se il tempo medio di interarrivo e' di 3 secondi, qual'e' il clock minimo della CPU tale che il tempo d'attesa nel sistema sia minore di 0.5 secondi?

Si consideri un S.O. di tipo batch, con un dispositivo di I/O. Supponendo che la CPU lavori al ritmo di 3 elaborazioni al secondo e l'unita' di I/O soddisfi 2 richieste al secondo, qual'e' il livello (minimo) di multiprogrammazione tale che l'utilizzazione della CPU sia maggiore del

60\% ?

Un Sistema Operativo possiede una CPU (che esegue in media di 50 processi al secondo) ed un disco (in media esegue 40 richieste al secondo). Sapendo che un processo ha uno spazio di indirizzamento medio di 500KB, che il Sistema Operativo non usa memoria virtuale, e che la CPU è utilizzata al 60\%, determinare la minima quantità di memoria di cui il Sistema Operativo deve disporre. [ricorda: la probabilità di avere n processi in coda in un sistema a coda ciclica è $\rho^n(1-\rho)/(1-\rho^{n+1})$ dove ρ è (frequenza esecuzione disco)/(frequenza elaborazione CPU)]