Ricordarsi di mettere il proprio nome, cognome, e numero di matricola in tutti i fogli.

Exercizio 1 1. Per i processi indicati nella tabella sotto, disegnare un diagramma di Gant che illustri la loro esecuzione usando:

- Round Robin (quantum = 2)
- Round Robin (quantum = 3)
- Shortest Job First non-preemptive

Processi	Tempo di arrivo	Tempo di esecuzione
A	0	5
B	1	3
C	2	1
D	3	4

Nel caso di arrivi simultanei di processi allo stato di pronto si dia la precedenza ai processi usciti dallo stato di esecuzione rispetto a quelli appena arrivati.

2. Cosa e' il waiting time medio?
3. Indicare per ciascuna politica al punto (1) sopra il waiting time medio.
4. Qual'e' il pregio principale della Round Robin? E quello della SJF? E qual'e' il difetto principale della SJF?

Risposta(Sketch)

1.
   - RR, quanto = 2:
     (0)A(2)B(4)A(6)C(7)D(9)B(10)A(11)D(13)
   - RR, quanto = 3:
     (0)A(3)B(6)C(7)A(9)D(12)D(13)
   - SJF:
     (0)A(5)C(6)B(9)D(13)
2. vedi note corso
3.
   - RR 2: A = 6, B= 6, C= 4, D=6, media = 22/4
   - RR 3: A = 4, B= 2, C= 4, D=6, media = 16/4
   - SJF: A=0,B=5,C=3,D=6, media 14/4
4. vantaggio RR: response time (cioe' reattivita'). Vantaggio SJF: optimal waiting time; svantaggio: difficolta' di implementazione.

Exercizio 2 1. Sia dato un file grande 500KB, con dimensione dei blocchi di 1KB (occupazione totale, 500 blocchi da 1024 byte). Si assuma che le informazioni relative agli attributi del file sono già contenute in memoria principale, e che un puntatore a blocco occupi 4 byte. Quanti accessi su disco sono necessari per leggere il 240-esimo blocco del file nel caso dei tre tipi di allocazione (includendo anche la lettura del blocco stesso)? Cambia qualcosa se si volesse leggere un blocco piu' avanti, esempio il blocco 280? (Se necessario, specificate quali assunzioni fate nella risposta data, per ogni tipo di allocazione.)

2. Qual è la quantità minima di memoria secondaria (in byte) che viene "sprecata" per registrare i blocchi in cui è memorizzato il file, per ciascun tipo di allocazione?

Risposta(Sketch)

1. Alloc. contigua: 1 accesso.

   Alloc. concatenata: 240 accessi (necessari per seguire l'intera catena).

   Alloc. indicizzata: 2 accessi. Assumiamo che in RAM sia presente il numero del blocco indice, e che si usi uno schema concatenato nel caso di piu' blocchi indice. In questo caso sul primo blocco indice posso avere il puntatore per i primi 256 blocchi, quindi incluso anche il blocco 240.

   Per il blocco 280: cambia solo per la indicizzata, dove sarebbero necessari 3 accessi, per i motivi spiegati sopra

2. Alloc. contigua: 0 byte

   Alloc. concatenata: (500 4) byte (spazio per allocare i puntatori al blocco successivo)

   Alloc. indicizzata: 2048 byte (necessari a memorizzare 2 blocchi indice nello schema concatenato)

Exercizio 3 1. Cos'e' il problema della Sezione Critica?

2. Quali proprieta' caratterizzano una corretta implementazione di una sezione critica?

Risposta(Sketch) Vedere appunti di corso.

Exercizio 4 [] Due processi A e B eseguono il seguente codice:

```
y = 0; (valore iniziale)
r = 1; (valore iniziale)
x = 1; (valore iniziale)
z = 0; (valore iniziale)
```

```
A:                          B:
  r:=r+1                      y:=x
  x:=x+2                      z:=z+1
  print(x)                    print(z)
  print(r)
```

Vogliamo che la esecuzione dei 2 processi termini con la variabile y che vale 3.

Agite, se necessario, sul codice, inserendo opportune operazioni su semaforo, in numero il piu' basso possibile (cioe' meno operazioni inserite meglio e'), in modo da garantire questo. Indicare bene i semafori utilizzati e il loro valore di inizializzazione.

Stessa domanda di prima su questa variante del codice:

```
y = 0; (valore iniziale)
r = 1; (valore iniziale)
x = 1; (valore iniziale)
z = 0; (valore iniziale)
```

```
A:                          B:

                            x:=x+1
  x:=x+1                    y:=x
  print(x)                  z:=z+1
  print(r)                  print(z)
```

In questo caso pero' vogliamo anche assicurarci che le operazioni su semaforo limitino il meno possibile il parallelismo tra A e B. Quindi spiegate anche brevemente perche' le posizioni in cui avete messo le operazioni su semaforo sono importanti e perche' tali operazioni non possono essere messe altrove (senza queste spiegazioni non considero l'esercizio svolto).

Risposta(Sketch)

```
             y = 0; (valore iniziale)
             r = 1; (valore iniziale)
             x = 1; (valore iniziale)
             z = 0; (valore iniziale)
 =>   semaphore S = 0; (valore iniziale)

A:                          B:
                           => P(S)
  r:=r+1                       y:=x
  x:=x+2
=>V(S)                          z:=z+1
  print(x)                     print(z)
  print(r)

             y = 0; (valore iniziale)
             r = 1; (valore iniziale)
             x = 1; (valore iniziale)
             z = 0; (valore iniziale)
 =>   semaphore S = 0; (valore iniziale)

A:                          B:
                              => P(S)
                              x:=x+1
                               y:=x
  x:=x+1
=>V(S)                          z:=z+1
  print(x)                     print(z)
  print(r)
```

Importante che la P sia messa comunque in alto per evitare il parallelismo tra i 2 incrementi di x in alto

Exercizio 5 [] Due processi A e B eseguono il seguente codice:

```
semaphore mutex = 1; (valore iniziale)
semaphore times_a = 2; (valore iniziale)
```

```
            semaphore times_b = 0; (valore iniziale)

A:                                              B:
repeat forever:                                 repeat forever:
               P(times_a)                           P(times_b)
               P(mutex)                             P(mutex)
                 <A1>                                 <B1>
               V(mutex)                             V(mutex)
               V(times_b)                           V(times_a)
```

L'esecuzione concorrente di A e B produce una sequenza (di lunghezza indefinita) di chiamate alle procedure <A1> e <B1>. Quale delle sequenze qui sotto riportate può essere la porzione iniziale di sequenze prodotte dall'esecuzione concorrente di A e B?

```
(1)            A1, A1, B1, A1, A1, B1, A1, A1, B1, ...
(2)            A1, B1, A1, A1, B1, A1, B1, A1, A1, ...
(3)            A1, B1, A1, B1, A1, B1, A1, B1, A1, ...
(4)            A1, A1, B1, B1, A1, B1, B1, A1, A1, ...
```

Motivare la risposta.

Risposta(Sketch) Solo la 3a. Il fatto che si parte con il semaforo `times_a` che vale 2, mentre `times_b` vale 0 implica che la differenza tra il numero di A1 eseguite e quelle di B1 non potra' mai essere negativa o maggiore di 2.

Exercizio 6 1. Cosa e' un algoritmo di page replacement?

2. Cosa si cerca di ottimizzare in un algoritmo del genere?

3. Come funziona l'optimal page replacement?

4. E l'algoritmo della "second chance"?

Risposta(Sketch) Vedi note di corso

Exercizio 7 1. Uno schema di crittografia a chiave simmetrica in cui i messaggi siano codificati usando una permutazione delle lettere dell'alfabeto che svantaggi comporterebbe?

2. Come si ovvia a questi svantaggi negli schemi moderni di crittografia a chiave simmetrica?

3. Usare le proprieta' dell'aritmetica modulo per calcolare in modo semplice $11^8 \bmod 7$

Risposta(Sketch)

Sfruttiamo la proprieta' che l'operazione di modulo distribuisce sulle operazioni come moltiplicazione ed esponenziazione (vedi appunti di corso). $11^8 = (11^2)^4$ Quindi possiamo calcolare

$11 \bmod 7 = 4$

$11^2 \bmod 7 = 4^2 \bmod 7 = 16 \bmod 7 = 2$

Ora possiamo calcolare $11^4 \bmod 7$ come $2^2 \bmod 4$, cioe' 4.

Infine $11^8 \bmod 7$ diventa di nuovo 2

Esercizio sotto: Come alternativa all'esercizio sopra (con meno punti in palio)

Exercizio 8 1. Che cos'e' la tabella delle pagine di un processo?

2. Si consideri un sistema in cui la tabella delle pagine di un processo puo' avere al massimo 64 entry. Un indirizzo fisico generato dal sistema e' scritto su 14 bit, e la RAM e' suddivisa in 16 frame.

   Quanto e' grande lo spazio di indirizzamento logico del sistema?

3. La tabella delle pagine di un processo del sistema sopra potrebbe dover essere a sua volta paginata? (giustificate numericamente la vostra risposta)

Risposta(Sketch) E' array in cui ogni entry corrisponde ad una delle pagine in cui e' stata suddivisa l'immagine del processo, e contiene il numero del frame in RAM nel quale e' stata memorizzata la pagina corrispondente.

64KByte (64×2^{10} byte)

No. Infatti, la tabella delle pagine di un processo ha al massimo 64 entry. Ogni entry deve contenere il numero di un frame, che e' scritto su 4 bit. Se anche si usa un byte per ogni entry, in tutto la tabella occupa 64 byte. Poiche' un frame e' grande 1024 byte, la tabella in questione puo' essere contenuta in un unico frame, e non deve essere paginata a sua volta.