**Ricordarsi di mettere il proprio nome, cognome, e numero di matricola in tutti i fogli.** Motivare sempre le risposte date. Non e' necessario dare risposte molto lunghe, ma e' importante rispondere in modo motivato ed esauriente alle domande poste (in altre parole, molto meglio una frase in piu' che una in meno).
**Per avere la sufficienza, e' **necessario** svolgere tutti i primi 4 esercizi.**

Non sono ammesse macchinette calcolatrici o altre macchine elettroniche; non e' consentito uso di appunti o libri.

Malacopia: consegnare, se necessario **solo** gli esercizi che devono essere corretti (non riportati in bella copia); barrare quindi gli altri

**Esercizio 1** Descrivere i principali vantaggi e svantaggi della memoria virtuale.

**Esercizio 2**
1. Qual'e' la differenza tra "modalita' utente" e "modalita' monitor" (nell'esecuzione di codice su un processore)?
2. E' vero che una interrupt viene gestita in modalita' utente? Perche'?

**Risposta**(Sketch) vedere appunti di corso

**Esercizio 3** Siano dati tre programmi concorrenti A, B e C, al cui interno vengono eseguite, rispettivamente, le procedure PA, PB e PC. Si vuole essere certi che PA verra' eseguita prima di PB e PC, mentre PB e PC devono poter essere eseguiti in qualsiasi ordine (quindi puo' succedere che venga eseguita prima PB e poi PC, o viceversa). Fornite una possible soluzione al problema, usando semafori; usate il minor numero di semafori possibile.

**Risposta**(Sketch)

```
Semaphore S = 0;
A            B           C

             P(S);       P(S);
PA;          PB;         PC
V(S);
V(S);
```

**Esercizio 4** quali sono i vantaggi e svantaggi principali di un algoritmo di CPU scheduling di tipo Round Robin rispetto ad un algoritmo FIFO?

**Risposta**(Sketch) vedere appunti di corso

**Esercizio 5** In un processore:

1. Cosa e' la Arithmetical Logical Unit (ALU)?
2. Cosa si intende per pipelining e per superscalarita'?

**Risposta**(Sketch) vedere appunti di corso

**Esercizio 6** In una stringa, usiamo la notazione {AB} per dire che in quel punto la stringa puo' avere sia AB che BA.

Considerare questi 2 processi, lanciati in parallelo:

```
P1                P2
loop forever     loop forever
print<A>          print<B>
```

Volgiamo che le strighe stampate siano {AB}{AB}....

1. Proponete una soluzione usando semafori. Indicare i valori di inizializzazione. (Come al solito, una soluzione con non-determinismo, che ammette piu' stringhe stampate, e' preferibile ad una deterministica.)

2. Proponete ora una soluzione equivalente con valori di inizializzazione dei semafori diversi (almeno un semaforo deve avere un valore iniziale diverso).

**Risposta**(Sketch)

```
P1                P2

loop forever      loop forever

print<A>          print<B>
S.V               T.V
T.P               S.P
```

con S,T inizialmente = 0.
Altra possibilita':

```
P1                P2

loop forever      loop forever
T.P                      S.P
print<A>          print<B>
S.V                    T.V
```

con S,T inizialmente = 1.

**Esercizio 7** Un hard disk ha una dimensione di $2^{36}$ byte, suddivisi in blocchi da 4Kbyte. Il sistema operativo che usa l'hard disk adotta una allocazione indicizzata dello spazio su disco, e usa 4 byte per scrivere il numero di un blocco. Sul disco e' memorizzato un file A grande 400 Kbyte. Tutti gli attributi del file A sono gia' in RAM. (Tutte le risposte sotto date vanno adeguatamente motivate)

1. Quante operazioni di I/O su disco sono necessarie per leggere il byte numero $1 + (200 \times 2^{10})$ del file?

2. Quanti accessi sarebbero stati necessari in caso di allocazione concatenata?

**Risposta**(Sketch)

1. In un blocco indice possono essere contenuti $2^{12}/4 = 2^{10}$ puntatori a blocco. Il file A e' composto da $400 \times 2^{10}/2^{12} = 100 \times 2^{12}/2^{12} = 100$ blocchi. L'uso di un solo blocco indice e' quindi sufficiente a memorizzare i numeri di tutti i blocchi in cui e' memorizzato A, per cui sono necessarie 2 operazioni di I/O: lettura del blocco indice, lettura del blocco desiderato del file

2. Se non ci fossero i byte persi per i puntatori, il byte numero $1+(200\times 2^{10})$ sarebbe nel blocco n. 50 (perche' con i primi 50 blocchi, con numerazione che parte da 0, copro $50 \times 2^{12} = 200 \times 2^{10}$ byte). Infatti il byte che mi interessa sarebbe il primo del blocco numero 50. Avendo i puntatori, e supponendo di avere solo i puntatori in avanti, in ogni blocco 4B sono necessari per il puntatore. Il byte che mi interessa e' comunque sempre al blocco numero 50; semplicemente e' preceduto da altri $50 \times 4$ bytes (che stanno dentro un singolo blocco).

   Il blocco n. 50 e' il 51-esimo blocco.

   Quindi: nel caso di allocazione concatenata dovro' percorrere tutti e 51 i blocchi fino ad arrivare al blocco 51-esimo

**Esercizio 8** Dato un sistema con una RAM di 4 frame, la tabella sotto indica il numero di pagina memorizzata nel frame, il tempo di caricamento, il tempo dell'ultimo accesso alla pagina, e il reference bit.

| Page | Load_Time | Last_Reference | Reference_Bit |
|---|---|---|---|
| 0 | 19 | 30 | 1 |
| 1 | 29 | 35 | 0 |
| 2 | 8 | 32 | 1 |
| 3 | 26 | 39 | 0 |

1. Quale pagina sara' sostituita con l'algoritmo di page replacement FIFO? Spiegare brevemente anche il perche'.
2. Stessa domanda per la LRU.
3. Stessa domanda per l'algoritmo della "second chance" (assumendo che l'ordine con cui le pagine vengono esaminate e' dalla piu' vecchia, cioe' presente da piu' tempo, a quella piu' giovane).

**Risposta**(Sketch)

1. 2
2. 0
3. 3 (la pagina piu' "vecchia" con reference bit 0)

**Esercizio 9** Si consideri una dimensione di pagina di $2^6$ byte, e la tabella di pagine seguente (la prima colonna indica il valid bit):

| in/out | Frame |
|---|---|
| out | 00101 |
| in | 00001 |
| in | 11011 |
| in | 11010 |
| out | 10001 |
| out | 11000 |
| in | 00101 |
| ... | ... |

Quale dei seguenti indirizzi virtuali produrrebbe un page fault? Per quelli che non producono un page fault, a quale indirizzo fisico corrispondono?

(1) 0000101101001
(2) 0000010010010
(3) 0000100010101
(4) 0000001110101

**Risposta**(Sketch) 1. fault (su page 5) 2. 11011010010 3. fault 4. 00001110101

**Esercizio 10**
1. Il 'cifrario di Cesare' (Caesar cipher) si basa su uno shift delle lettere dell'alfabeto. Con un alfabeto di 21 caratteri quante sono le chiavi possibili? Come posso creare una chiave piu' sicura basandomi comunque sul cifrario di Cesare?
2. Se usassi invece una permutazione delle lettere dell'alfabeto (sempre a 21 caratteri), quante chiavi ci sarebbero? Sarebbe una chiave sicura?
3. Usare le proprieta' dell'aritmetica modulo per calcolare in modo semplice $26^5 \bmod 11$

**Risposta**(Sketch)

1. 20. banale da attaccare via ricerca esaustiva (brute force), che puo' essere comunque ridotta usando analisi statistiche

   Miglioria: polyalphabetic substitution. Si sceglie una sequenza finita di numeri compresi tra 1 e 20, ognuno dei quali da intepretare come una chiave di un Caesar cipher, e da usare in sequenza, ripetendo la sequenza se necessario rispetto alla lunghezza del messaggio. Non e' comunque considerata una crittografia sicura, anche in questo caso soggetta ad attacchi di analisi statistiche (combinata con ricerca esaustiva). La ricerca esaustiva puo' essere sufficiente da sola, se la sequenza non e' troppo lunga

2. 21!. anche questa non e' comunque considerata una crittografia sicura, anche in questo caso soggetta ad attacchi di analisi statistiche (eventualmente combinata con ricerca esaustiva). In questo caso la ricerca esaustiva non sarebbe possibile, dato il numero troppo alto di chiavi.

3. Sfruttiamo la proprieta' che l'operazione di modulo distribuisce sulle operazioni come moltiplicazione ed esponenziazione (vedi appunti di corso). $26^5 \times 11 = (4^5) \times 11$ (poiche' $26^5 \times 11 = 4$)

   Quindi possiamo calcolare $4^2 \bmod 11 = 5$

   $5^2 \bmod 11 = 3$

   $(4^5) \times 11 = 4 \times 3 \bmod 11 = 1$