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 ognuno dei 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 Qual'e' l'idea della paginazione? Perche' e' preferibile una gestione della memoria con "paginazione" rispetto alla gestione "contigua"? (In altre parole, perche' e' stata introdotta la paginazione?)

Risposta(Sketch) Vedere appunti di corso.

Esercizio 2 Cosa e' un algoritmo di page replacement?

Risposta(Sketch) Vedi note di corso

Esercizio 3
1. Che differenza c'e' tra 'thread' e 'processo'? E che vantaggi ha il 'thread'?
2. E' possibile nei sistemi operativi moderni, avere sia processi che thread, o sono due concetti incompatibili?

Risposta(Sketch) Vedere appunti

Esercizio 4 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, con la condizione che tutti i semafori usati abbiano un valore di inizializzazione 0. Come al solito, una soluzione con non-determinismo, che ammette piu' stringhe stampate, e' preferibile ad una deterministica.
2. Stessa cosa del punto precedente, ma ora imponendo che tutti i semafori abbiano valore di inizializzaione 1

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.

Per avere inizializzazione a 1 sufficiente spostare le P su semafori all'inizio

Esercizio 5 All'interno di un sistema operativo, un certo processo P e' correntemente in stato di "Running" e si sa che non dovra' piu' rilasciare la CPU volontariamente prima di aver terminato la propria esecuzione (in altre parole, non deve piu' eseguire operazioni di I/O, di sincronizzazione o di comunicazione con altri processi).

1. Quale/quali, tra gli algoritmi di scheduling FCFS, SJF preemptive, SJF non-preemptive, round robin garantisce/garantiscono che il processo P riuscira' a portare a termine la propria computazione senza mai perdere il controllo del processore (cioe' rimanendo sempre in stato running)?

2. Quale/quali, tra gli algoritmi di scheduling FCFS, SJF preemptive, SJF non-preemptive, round robin garantisce/garantiscono che il processo P riuscira' a portare a termine la propria computazione?

3. In quale caso un processo potrebbe uscire volontariamente dallo stato di Ready? (cioe': un processo che e' in stato Ready, in quale altro stato puo' andare?)

Risposta(Sketch)

1. FCFS, SJF non-preemptive.

2. FCFS, SJF non-preemptive e round robin. Infatti, nel caso di SJF preemptive potrebbe sempre arrivare in coda di ready un processo che deve usare la CPU per un tempo minore di quanto rimane da eseguire a P.

3. mai (cioe' un processo Running puo' andare solo in stato Running).

Esercizio 6 Riportate un semplice esempio in pseudo-codice (simile a quello usato per la prima parte di questa domanda) di due processi concorrenti che usano uno o piu' semafori per sincronizzarsi e che, a seconda dell'ordine relativo in cui vengono eseguite le istruzioni dei due processi, puo' sia funzionare correttamente che portare in una situazione di deadlock. Indicate anche come devono essere inizializzati i semafori che usate.

Risposta(Sketch)

```
P1
wait(mutex1)
wait(mutex2)
sez. critica
signal(mutex2)
signal(mutex1)
```

```
P2
wait(mutex2)
wait(mutex1)
sez. critica
signal(mutex1)
signal(mutex2)

semaphore mutex1 = 1; semaphore mutex2 = 1;
```

Esercizio 7 Supponiamo che le pagine in uno spazio di indirizzi virtuali siano usate secondo l'ordine seguente (quella sotto e' cioe' la "reference stream")

```
1 2 1 3 2 1 4 3 1 4
```

Supponiamo anche ci siano 3 frame disponibili, inizialmente liberi. Assumiamo anche che le decisioni sulla paginazione siano fatte su domanda, cioe' quando un page fault ha luogo. Mostrare i contenuti dei frame dopo ogni accesso a memoria, secondo la politica LRU. Quanti page fault occorrono?

Risposta(Sketch)

Frame	Reference									
	1	2	1	3	2	1	4	3	1	4
0	1	1	1	1	1	1	1	1	1	1
1		2	2	2	2	2	2	3	3	3
2				3	3	3	4	4	4	4
Fault?	Y	Y	N	Y	N	N	Y	Y	N	N

Numero totale faults = 5.

Esercizio 8 In un sistema la memoria fisica e' divisa in 2^{20} frame, un indirizzo logico e' scritto su 31 bit, e una pagina e' grande 512 byte.

Quanto e' grande un frame? Quanti frame occupa la page table piu' grande del sistema?

Risposta(Sketch) Un frame e' grande come una pagina, quindi $2^9 = 512$ byte. Quindi la page table piu' grande puo' avere $2^{(31-9)} = 2^{22}$ entry. Nel sistema vi sono 2^{20} frame, per cui sono necessari tre byte per scrivere il numero di un frame, e quindi la page table piu' grande occupa $(2^{22} \times 3)/2^9$ frame $= 2^{13} \times 3$ frame $=$ 24K frame

Esercizio 9 Riportare la distinzione tra crittografia a chiave simmetrica e a chiave pubblica. Quali sono i principali vantaggi della prima? E quelli della seconda?

Risposta(Sketch) Vedere appunti di corso.