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 Considerate i 3 processi sotto

```
     P1          P2        P3

  print(A)    print(B)   print(C)

  print(D)    print(E)
```

Agite, se necessario, sul codice, inserendo opportune operazioni su semaforo, in modo che le sole stringhe stampabili siano ABECD oppure BAECD.

Risposta(Sketch)

```
     P1          P2        P3
                           P(S)
  print(A)    print(B)   print(C)
V(T);P(U)       P(T)       V(U)
  print(D)    print(E)
                V(S)
```

Esercizio 2 Quali sono i vantaggi della paginazione, rispetto ad una allocazione della memoria centrale di tipo contiguo? Ci sono anche degli svantaggi?

Risposta(Sketch) Vedi appunti di corso

Esercizio 3 Che differenza c'e' tra una variabile di tipo 'intero' e un semaforo (non binario)?

Risposta(Sketch)
Risposta(Sketch) Un semaforo puo' essere utilizzato unicamente con le operazioni P e V. Non posso, per esempio, utilizzarlo in operazioni aritmetiche o espessioni booleane di confronto.

Esercizio 4 Disegnare il grafo che rappresenta gli stati possibili di un processo, e le transizioni tra stati. Per ogni transizione, indicare almeno una ragione che puo' causare quella transizione.

Risposta(Sketch) Vedi appunti di corso

Esercizio 5 Tre processi, identificati dalle lettere A, B, C, arrivano agli istanti 0, 2, 4, rispettivamente. Tali processi hanno un tempo di esecuzione di 5, 6, 4 unita' di tempo rispettivamente e priorita' 3, 5, 4 (dove 5 e' la massima priorita' e 0 la minima). Si considerino le seguenti politiche di ordinamento:

1. Round Robin con quanto di tempo di ampiezza 3
2. Priority Scheduling (preemptive)

Per ognuna di esse: illustrarne la esecuzione con un diagramma Gannt; determinare, trascurando i ritardi dovuti allo scambio di contesto, il tempo medio di attesa.

Risposta(Sketch)

1. RR:

   ```
   [A]3[B]6[A]8[C]11[B]14[C]15
   ```

   W.T: A=3, B=6, C=7 quindi WT medio = 16/3

2. Priority scheduling

   ```
   [A]2[B]8[C]12[A]15
   ```

   W.T: A=10, B=0, C=4, quindi WT medio = 14/3

Esercizio 6 1. Cosa contiene la page table (cioe' come e' fatta)?

2. C'e' una qualche relazione tra la dimensione della page table e la dimensione degli indirizzi logici?

Risposta(Sketch) Vedere note di corso. Per il punto 2: un indirizzo logico e' fatto da una parte "numero pagina" e una parte offset. Se il number di bit della prima parte e' n, allora il numero di entry della PT puo' essere al max 2^n

Esercizio 7 Quelli sotto sono alcuni tra gli algoritmi per page-replacement visti a lezione. Indicare per ciascuno di essi il problema principale che comporta:
First-In First-Out; Optimal Algorithm; LRU (Least Recently Used).

Risposta(Sketch)

1. Si puo' togliere una pagina ancora fortemente usata
2. impossibile da implementare perche' fa riferimento al futuro
3. necessaria assistenza harware; resta comunque costoso (c'e' qualcosa da fare per ogni accesso alla memoria, come visto nelle implementazioni possibili basate su stack o su counter/timer discusse a lezione); in genere si implementano delle approssimazioni dell'algoritmo, come la "second chance"

Esercizio 8 Considerate i protocolli sotto, per gestire il problema della sezione critica.

1.
```
// Shared variables
  boolean c1 = false, c2 = false;

// Process 1                          // Process 2
<init1>;                               <init2>;
while(true) {                          while(true) {
     c1 = true;                             c2 = true;
```

```
     while (c2 ) { };                    while (c1) { };
     <crit1>;                               <crit2>;
     c1 = false;                           c2 = false;
    <non-crit1>;                          <non-crit1>;
}                                                    }
```

2.
```
                // Shared variables
        boolean c1 = false, c2 = false; integer turn = 1;
// Process 1                          // Process 2
<init1>;                               <init2>;
while(true) {                         while(true) {
     turn = 2;                              turn = 1;
     c1 = true;                             c2 = true;
     while (c2 && turn == 2) { };    while (c1 && turn == 1) { };
     <crit1>;                               <crit2>;
     c1 = false;                           c2 = false;
    <non-crit1>;                          <non-crit1>;
}                                                    }
```

I protocolli sopra soddisfano le properta' di mutual exclusion? E' di assenza di deadlock? Spiegate intuitivamente perche'.

Risposta(Sketch) Sul primo:

Mutua esclusion: ok. Esiste possibilita' deadlock (vedere appunti lezione: e' sostanzialmente l'algoritmo 2 visto a lezione quando si discuteva della soluzione al problema della sezione critica; il dealock occorre se, eseguendo il primo processo, c'e' un context switch dopo c1=true; a questo punto esegue anche il secondo e puo' mettere c2=true. Essendo sia c1 ch c2 true, nessun processo e' in grado di superare il while.

Sul secondo:

Mutual exclusion: non e' garantita. Esempio: supponiamo esegua il primo processo, e ci sia un context switch dopo turn=2. Il secondo processo mette turn=1 e puo' entrare in sezione critica (perche' c1 e' ancora false). Se ora c'e' un context switch dentro la sezione critica, il primo processo torna ad eseguire, ed entra a sua volta in sezione critica (perche' turn=1)

Deadlock: e' deadlock-free: la presenza di turn impedisce la possibilita' che entrambi i processi restino bloccati sul while interno.

Il secondo protocollo e' una variante dell' algoritmo 3 visto a lezione; la differenza e' che l'ordine degli assegnamenti su turn e c1/c2 e' invertito, e a causa di questo la mutual exclusion non e' garantita.

Esercizio 9

1. Sia dato un file grande 1500KB, con dimensione dei blocchi di 2KB (2^{11} B). Si assuma che le informazioni relative agli attributi del file sono gia' contenute in memoria principale, e che un puntatore a blocco occupi 4 byte. Quanti accessi su disco sono necessari per leggere il 600-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 precendete, esempio il blocco 300? (Se necessario, specificate quali assunzioni fate nella risposta data, per ogni tipo di allocazione.)

2. Qual e' la quantita' minima di memoria secondaria (in byte) che viene 'sprecata' per registrare i blocchi in cui e' memorizzato il file, nel caso di allocazione contigua e indicizzata (incluso spazio eventuale per puntatori)?

Risposta(Sketch)

1. Alloc. contigua: 1 accesso.

   Alloc. concatenata: 600 accessi (necessari per seguire l'intera catena).

   Alloc. indicizzata: 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 un blocco possono essere contenuti $2^{11}/2^2 = 2^9$ (circa 500) puntatori, quindi per accedere al blocco 600 devo leggere anche il secondo blocco indice. Quindi 3 accessi.

   Nel caso blocco 300: 2 accessi sono sufficienti.

2. Alloc. contigua: 0 byte

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

Esercizio 10 1. Descrivere i passi che portano alla determinazione della chiave pubblica e privata nell'algoritmo di RSA.

2. Nello stesso algoritmo, il codaggio di un messaggio deve essere necessariamente fatto con una delle 2 chiavi, o puo' essere fatto con entrambe? Spiegare bene il perche', a partire dal modo con cui la crittografia e' effettuata con RSA, se necessario richiamando proprieta' aritmetiche.

Risposta(Sketch)

1. vedi appunti corso

2. si puo' codificare con entrambe le chiavi. Motivo: tutto alla fine e' riconducibile alla commutativita' della moltiplicazione (in una risposta completa i dettagli sarebbero da mostrare); vedere appunti corso.