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 Cos'e' il process control block? A cosa serve? Che informazioni contiene (indicate almeno 2 campi)?

Risposta(Sketch) Vedi appunti di corso.

Esercizio 2 Considerate i seguenti processi:

```
  P             Q

x:= x+1       x:= x-1
x:= x*2
```

Essi vengono lanciati in parallelo con valore di x iniziale 0. Alla fine della loro esecuzione, x puo' avere i valori $0, 1$. Puo' avere altri valori?

Risposta(Sketch) Si, ci possono essere altri valori perche' c'e' una variabile x condivisa e le operazioni di incremento o moltiplicazione indicate non sono atomiche (le operazioni effettivamente eseguite sono quelle del linguaggio assembler). Per esempio x puo' assumere il valore -1. Si veda su questo le note corso su critical region.

Esercizio 3

1. Un file occupa, sull'hard disk, 3 blocchi. Descrivete come questo file viene memorizzato sull'hard disk secondo le tre tecniche fondamentali di allocazione dello spazio su disco. Si consiglia vivamente di accompagnare la descrizione ad opportuni disegni esemplificativi. Ad esempio, assumete che l'ipotetico hard disk su cui e' memorizzato il file sia formato da 16 blocchi in tutto, e che non contenga altri file. Scegliete anche dei valori adeguati – ovviamente compresi tra 0 e 15 – per i 3 blocchi in cui e' memorizzato il file.

2. Quale delle tre tecniche descritte e' meno adatta ad un accesso diretto ai dati del file, e perche'?

Risposta(Sketch)

1. vedi appunti di corso

2. L'allocazione concatenata, perche' sono necessari n accessi al disco per leggere i dati del file contenuti nell'ennesimo blocco del file

Esercizio 4

1. Quali sono i vantaggi della paginazione della memoria?

2. E quali gli svantaggi?

Risposta(Sketch) Vantaggi: no frammentazione esterna; non necessario compattare; migliore gestione dimensioni variabili di processi

Svantaggi: Maggior lavoro per SO (gestione dei frame liberi, aumento del tempo di context switch), memoria presa da tabelle delle pagine, maggior tempo di traduzione degli indirizzi da logici a fisici, supporto hardware per tale traduzione (memoria associativa TLB)

Esercizio 5 Un sistema ha una memoria RAM formata da 1024 frames di 512 byte ciascuno. Lo spazio di indirizzamento logico e' grande il quadruplo di quello fisico.

1. Qual e' l'indirizzo fisico piu' grande ammesso dal sistema?
2. Qual e' l'indirizzo logico piu' grande ammesso dal sistema?
3. In quante pagine e' suddiviso lo spazio di indirizzamento logico del sistema?
4. Se la tabella delle pagine e' mantenuta esclusivamente in RAM, di quanto si degradano approssimativamente le prestazioni del sistema rispetto ad un sistema equivalente ma senza paginazione della memoria?
5. In pratica, la tabella delle pagine, in quale posto (o quali posti) e' mantenuta? Perche' questo limita l'eccessiva perdita di prestazioni del sistema?

Risposta(Sketch)

1. La RAM e' formata da $2^{10} * 2^9 = 2^{19}$ celle, il cui indirizzo va da 0 a $2^{19} - 1$
2. Lo spazio logico e' pari a $4 * 2^{19} = 2^{21}$ per cui indirizzo piu' grande e' $2^{21} - 1$
3. Lo spazio di indirizzamento logico e' suddiviso in $2^{21}/2^9 = 2^{12}$ pagine
4. Il tempo medio di esecuzione dei programmi raddoppia, poiche' il tempo di accesso ad ogni cella di memoria (dovendo passare attraverso la tabella delle pagine) raddoppia
5. Si usa una memoria associativa di supporto (TLB). Vedere appunti di corso.

Esercizio 6 Sotto, e' riportato il programma java che implementa il "bounded buffer" usando semafori, come visto a lezione. I produttori chiamano il metodo `enter` ed i consumatori il metodo `remove`.

```
public class BoundedBuffer
{
   public BoundedBuffer()
   {  in = 0;
      out = 0;

      buffer = new Object[BUFFER_SIZE];
      mutex = new Semaphore(1);
      empty = new Semaphore(BUFFER_SIZE);
      full = new Semaphore(0);
   }
   public void enter(Object item) {
      empty.P();
```

```
      mutex.P();

      buffer[in] = item;
      in = (in + 1) % BUFFER_SIZE;

      mutex.V();
      full.V();
   }
   public Object remove() {
      full.P();
      mutex.P();

      Object item = buffer[out];
      out = (out + 1) % BUFFER_SIZE;

      mutex.V();
      empty.V();
      return item;
   }
   private static final int   BUFFER_SIZE = 2;
   private Semaphore mutex;
   private Semaphore empty;
   private Semaphore full;
   private int in, out;
   private Object[] buffer;
}
```

Rispondere alle domande seguenti:

1. Che problemi ci sarebbero se il semaforo `mutex` fosse tolto?
2. Che problemi ci sarebbero se i semafori `empty, full` fossero tolti?
3. Nel codice per il metodo `remove`, il comando `full.P()` precede `mutex.P()`. E' possibile modificare l'ordine di questi 2 comandi?
4. Nel codice per il metodo `remove`, il comando `mutex.V()` precede il comando `empty.V()`. E' possibile modificare l'ordine di questi 2 comandi?

Risposta(Sketch) Alla 1a domanda: mutex implementa la mutua exclusion su accesso buffer. Senza questo semaforo quindi piu' processi potrebbero lavorare in contemporanea sul buffer (esempio piu' produttori e allora ci potrebbe essere conflitto su operazioni su indice `in`)

Alla 2a domanda: full e empty servono per verificare le condizioni di accesso buffer; senza di essi un produttore potrebbe inserire nel buffer anche se questo e' pieno, e similmente per un consumatore a buffer vuoto.

Alla 3a domanda: L'ordine delle 2 operazioni non puo' essere invertito, poiche' si potrebbe avere un deadlock (bloccaggio) da parte di produttori e consumatori.

alla 4a: L'ordine delle 2 operazioni potrebbe essere invertito, e la soluzione sarebbe ancora corretta.

Esercizio 7 1. Supponiamo che un sistema prende 1 unita' di tempo per fare un a context switch. Abbiamo un sistema con 3 task; il primo richiede 70 unita' di tempo, il secondo 25 unita' e il terzo 45. Dato uno scheduler di tipo round robin, con quanto pari a 20 unita' di tempo, mostrate con un diagramma di Gantt quale task gira e quando. Calcolate poi il waiting time di ogni task, e impostate la formula per il waiting time medio.

2. Quale o quali sono a vostro avviso i vantaggi principali della politica di round robin?

Risposta(Sketch)

```
[A 20]21[B 41]42[C 62]63[A 83]84[B 89]90[C 110] 111 [A 131] 132 [C 137]138[A 148]
```

Vantaggio RR: responsiveness

Esercizio 8 (CPU scheduling) 1. In pratica, nel CPU scheduling dei sistemi operativi moderni si usano 'multilevel queue' (code multilivelli). Cosa sono e perche' si usano?

2. Cosa e' la process affinity di un processo rispetto ad certo processore?

Risposta(Sketch) Vedi appunti di corso.

Esercizio 9 1. Cifratura a chiave simmetrica; perche' una permutazione sull'alfabeto non e' considerata una buona chiave?

2. Nella "cifratura a blocchi", con blocchi di 64 bit, e' fattibile usare come chiave una qualunque permutazione di questi bit, rappresentata da una tabella che specifica come viene tradotta ogni sequenza di bit?

Risposta(Sketch) 1a domanda: attacchi di tipo statistico
2a domanda: con 64 bit una tabella avrebbe 2^{64} entry, troppo grande per essere manipolata (esempio: comunicata o aggiornata)