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 Cosa e' un page fault, e come viene gestito dal Sistema Operativo?

Risposta(Sketch)

Esercizio 2 Cosa si intende con multiprogrammazione? Perche' e' importante, anche su macchine con un solo processore?

Risposta(Sketch)

Esercizio 3 Considerare questi 2 processi, lanciati in parallelo:

```
P1                 P2

loop forever      loop forever

print<A>           print<B>
```

Volgiamo che la striga stampata sia ABAB....

1. Proponete una soluzione usando semafori. Indicare i valori di inizializzazione.
2. Proponete ora una soluzione equivalente in cui almeno un semaforo abbia un valore iniziale diverso.

Risposta(Sketch)

```
P1                 P2

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

con S=1,T=0.

Soluzione con inizializzazioni diverse:

```
P1                 P2

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

con S=T=0.

Esercizio 4 1. Spiegare il funzionamento dell'algoritmo di LRU replacement.

2. Questo algoritmo e' "pratico" (cioe' e' effettivamente implementato su SO) nell'ambito della paginazione? Motivare la risposta.

Risposta(Sketch) Per funzionamento: vedere appunti corso.

Per praticita': l'algoritmo richiederebbe di memorizzare il tempo dell'ultimo accesso ad ogni pagina, sostanzialmente ogni volta che si accede ad una pagina ci sarebbe una informazione da aggiornare.

Questo aggiungerebbe notevole overhead e rende l'algoritmo non "pratico".

Esercizio 5 1. Per effetturare una operazione di Input/Output su periferica, e' necessario l'intervento del SO?

2. Come viene a sapere (il SO, o il processo che ha richiesto l'operazione) del completamento dell'operazione stessa?

Risposta(Sketch) Interviene il SO.

Le periferiche di Input/output sono lente e quindi gestite in modo asynchronous. Una interrupt viene prodotta quando l'operazione e' terminata. Nel frattempo la CPU puo' dedicarsi ad altri processi (multiprogrammazione)

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 in 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 (per dettagli vedere appunti corso).

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 (anche qui: per dettagli vedere appunti corso).

alla 4a: L'ordine delle 2 operazioni potrebbe essere invertito, e la soluzione sarebbe ancora corretta (le operazioni di V – incremento – non sono problematiche perche' non bloccanti).

Esercizio 7 In un sistema che adotta la memoria paginata le pagine hanno una dimensione tale da produrre una frammentazione interna media per processo di 2 Kbyte (cioe' 2^{11} byte). La dimensione della tabella delle pagine piu' grande del sistema e' di X byte, e si sa che se tale dimensione fosse anche solo X+1 byte, il sistema dovrebbe adottare una paginazione a piu' livelli. Ogni entry di una tabella delle pagine occupa 2 byte, e tutti i bit vengono usati per scrivere il numero di un frame.

1. Quanto e' grande una pagina?
2. Quanti frame ci sono nel sistema?

3. Quante entry ci sono al piu' nella tabella di pagine?

4. Quanto sono grossi lo spazio di indirizzamento logico e fisico del sistema?

Risposta(Sketch)

1. Il sistema adotta pagine da 4 Kbyte, ossia 2^{12} byte

2. Ogni entry occupa 2B e sono tutti usati per scrivere un numero di frame. Quindi ci sono 2^{16} frame

3. La tabella delle pagine piu' grande del sistema occupa un intero frame, e quindi ha in tutto $2^{12}/2 = 2^{11}$ entry

4. Lo spazio logico e' quindi grande $2^{11} * 2^{12} = 2^{23}$ byte e quello fisico e' grande $2^{16} * 2^{12} = 2^{28}$ byte

Esercizio 8 1. Nella crittografia, cosa significa adottare una codifica 'a blocchi' ?

2. Come si evita che blocchi uguali diano codifiche identiche?

3. Supponiamo che i messaggi siano sequenze di bit, e che un blocco sia fatto da 3 bit. Seguendo il metodo (o uno dei metodi) che avete descritto al punto (2) precedente, mostrate come viene tradotto il messaggio AB dove A e' il numero binario di 3 cifre che rappresenta il numero 7, e B e' lo stesso per il numero 4.

   Come tabella di mapping, prendere la seguente:

Input	Output	Input	Output
000	110	100	001
001	100	101	111
010	011	110	010
011	000	111	101

   Fate vedere anche come viene fatta la decodifica.

   Eventuali altri parametri che intervengono nella traduzione possono essere liberamente scelti (ovviamente indicandoli).

Risposta(Sketch) Per i primi 2 punti, vedere note del corso.

AB sarebbe 74 in binario, cioe' 111100.

Faccio la traduzione assumendo un random Initialisation Vector (IV), diciamo 010.

Ora per la traduzione faccio cosi'. Il primo blocco di ciphertext e' ottenuto applicando la tabella allo XOR tra A e IV. Indicando con # lo XOR, con $K(-)$ l'applicazione della tabella, e con c(1) il primo blocco di ciphertext, abbiamo

$c(1) = K(A\#IV) = K(111\#010) = K(101) = 111$

Per $c(2)$, si usa come random vector $c(1)$, quindi abbiamo

$c(2) = K(B\#c(1)) = K(100\#111) = K(011) = 000$

Facciamo la decodifica, indicando con K^{-1} l'inverso della tabella:

$A = K^{-1}(c(1))\#IV = K^{-1}(111))\#010 = 101\#010 = 111$

$B = K^{-1}(c(2))\#c(1) = K^{-1}(000))\#111 = 011\#111 = 100$