Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Università degli Studi di Padova – Dipartimento di Matematica. - Corso di Laurea in Informatica

Il presente esame scritto deve essere svolto in forma individuale in un tempo massimo di 90 minuti dalla sua presentazione.
Non è consentita la consultazione di libri o appunti in forma cartacea o elettronica, né l'uso di palmari e telefoni cellulari.
La correzione e la sessione orale avverranno in data e ora comunicate dal docente durante la prova scritta; i risultati saranno esposti sul sito del docente entro il giorno precedente gli orali.
Per superare l'esame, il candidato deve acquisire almeno 1.5 punti nel Quesito 1 e un totale di almeno 18 punti su tutti i quesiti, inserendo le proprie risposte interamente su questi fogli. Riportare generalità e matricola negli spazi indicati.
Per la convalida e registrazione del voto finale il docente si riserva di proporre al singolo candidato una prova orale.

Quesito 1 (punti 4): ***1 punto per risposta giusta, diminuzione di 0,33 punti per risposta sbagliata, 0 punti per risposta vuota***

[1.A] In quale tra i seguenti sistemi operativi è più conveniente l'utilizzo di *Inverted Page Tables*:

1. nessuno dei seguenti, il vantaggio è pari per tutti
2. sistemi a 16 bit
3. sistemi a 32 bit
4. sistemi a 64 bit

[1.B] Una *system call* bloccante causa sempre un *context switch*:

1. Sempre
2. Mai
3. Sì ma solo se la macchina ha più di un processore
4. Sì ma solo se c'è qualche altro processo attivo

[1.C]: Quale tra i seguenti costituisce un criterio valido di valutazione di una politica di ordinamento di processi:

1. la capacità di trattare anche processi di lunga durata
2. il numero di processi completati per unità di tempo
3. il numero di processi in esecuzione per unità di tempo
4. il numero di processi in attesa di essere eseguiti.

[1.D] Quale tra le seguenti affermazioni, fatte osservando un grafo di allocazione delle risorse, è certamente vera in generale:

1. se vi sono percorsi chiusi, allora vi è situazione di stallo
2. se non vi sono percorsi chiusi allora non vi è situazione di stallo
3. se in un percorso chiuso rilevato si trovano solo risorse a molteplicità unaria, occorre analizzare il caso per decidere
4. nessuna delle precedenti tre possibili risposte.

RISPOSTE AL QUESITO 1: A ____ B ____ C ____ D ____

Quesito 2– (4 punti):
Si consideri la seguente serie di riferimenti a pagine di memoria: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.
Si considerino le seguenti politiche di rimpiazzo:
• LRU
• Optimal
Quanti *page fault* avvengono considerando una RAM con solo 4 *page frame* ed inizialmente vuota?
Si completino inoltre le tabelle mostrando ad ogni istante il contenuto dei 4 page frame di cui è composta la RAM (non è necessario che lo studente mantenga un preciso ordine delle pagine virtuali nelle *page frame*.
Nota: nella tabella la prima riga indica la pagina di memoria virtuale riferita in quell'istante.

Politica di rimpiazzo **LRU**; totale *page fault*? ____

r1	**r2**	**r3**	**r4**	**r2**	**r1**	**r5**	**r6**	**r2**	**r1**	**r2**	**r3**	**r7**	**r6**	**r3**	**r2**	**r1**	**r2**	**r3**	**r6**

Politica di rimpiazzo **Optimal**; totale *page fault*? ____

r1	**r2**	**r3**	**r4**	**r2**	**r1**	**r5**	**r6**	**r2**	**r1**	**r2**	**r3**	**r7**	**r6**	**r3**	**r2**	**r1**	**r2**	**r3**	**r6**

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Quesito 3 – (12 punti):

Il problema del "produttore/consumatore" è un classico problema di sincronizzazione tra più processi che accedono concorrentemente a risorse condivise.

A) Lo studente utilizzi i **monitor** per scrivere due procedure chiamate `Producer` e `Consumer` che possano essere eseguite concorrentemente al fine di risolvere il problema evitando il *deadlock* del sistema.
(Si consideri il caso in cui le risorse prodotte e non ancora consumate possano essere al massimo `N`).

B) Lo studente utilizzi i **semafori** per risolvere lo stesso problema

(Lo studente si ricordi che per scrivere le sue risposte può utilizzare, se necessario, anche la parte posteriore del foglio).

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Quesito 4 – (4 punti):
Una "chiavetta USB" da 8 GB è formattata con un *filesystem* di tipo FAT con blocchi da 4kB.
Calcolare la dimensione di ogni record della FAT (scegliendo fra lunghezze che siano potenze di 2; ovvero di 8, 16, 32 e 64 bit, non altri valori intermedi) e la dimensione totale della FAT.

Quesito 5 – (7 punti):
Gli hard disk sono componenti molto importanti di un computer che permettono di immagazzinare permanentemente un insieme moderatamente grande di informazioni. Il sistema operativo si occupa di gestire anche queste componenti *hardware* permettendo, ad esempio, operazioni su file quali memorizzazione, recupero, cancellazione, ecc.
I computer moderni sono dotati di *hard disk* di capacità sempre maggiore fornendo dunque un vantaggio in termini di spazio di memorizzazione agli utenti ma anche nuove complessità di gestione per il sistema operativo.
Lo studente illustri, in massimo una pagina, le implicazioni (es. problematiche e possibili soluzioni, ma anche semplificazioni che diventerebbero possibili) per le varie componenti e strutture di un sistema operativo che si trovasse a dover gestire un *hard disk* di capacità infinita.

(Lo studente si ricordi che per scrivere le sue risposte può utilizzare anche la parte posteriore del foglio).

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Soluzione

Soluzione al Quesito 1
[1.A]: risposta 4
[1.B]: risposta 4
[1.C]: risposta 2
[1.D]: risposta 2

Soluzione al Quesito 2

Politica di rimpiazzo **LRU**; totale *page fault*? **_10_** (quelli evidenziati)

r1	**r2**	**r3**	**r4**	**r2**	**r1**	**r5**	**r6**	**r2**	**r1**	**r2**	**r3**	**r7**	**r6**	**r3**	**r2**	**r1**	**r2**	**r3**	**r6**
1	**2**	**3**	**4**	2	1	**5**	**6**	2	1	2	**3**	**7**	**6**	3	2	**1**	2	3	6
	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3
		1	2	3	4	2	1	5	6	6	1	2	3	7	6	3	3	1	2
			1	1	3	4	2	1	5	5	6	1	2	2	7	6	6	6	1

Politica di rimpiazzo **Optimal**; totale *page fault*? **_8_** (quelli evidenziati)

r1	**r2**	**r3**	**r4**	**r2**	**r1**	**r5**	**r6**	**r2**	**r1**	**r2**	**r3**	**r7**	**r6**	**r3**	**r2**	**r1**	**r2**	**r3**	**r6**
1	**2**	**3**	**4**	4	4	**5**	**6**	6	6	6	6	**7**	7	7	7	**1**	1	1	1
	1	2	3	3	3	3	3	3	3	3	3	6	6	6	6	6	6	6	6
		1	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3
			1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2

Soluzione al Quesito 3
Il problema e le sue varie soluzioni sono chiaramente spiegate nel libro di testo e nei lucidi. Ad esempio:

A) Soluzione **MONITOR**

```
monitor ProducerConsumer
    condition full, empty;
    integer count;
    procedure insert(item: integer);
    begin
        if count = N then wait(full);
        insert_item(item);
        count := count + 1;
        if count = 1 then signal(empty)
    end;
    function remove: integer;
    begin
        if count = 0 then wait(empty);
        remove = remove_item;
        count := count – 1;
        if count = N – 1 then signal(full)
    end;
    count := 0;
end monitor;
```

```
procedure producer;
begin
    while true do
    begin
        item = produce_item;
        ProducerConsumer.insert(item)
    end
end;
procedure consumer;
begin
    while true do
    begin
        item = ProducerConsumer.remove;
        consume_item(item)
    end
end;
```

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

B) Soluzione **SEMAFORI**

```
#define N 100                           /* number of slots in the buffer */
typedef int semaphore;                  /* semaphores are a special kind of int */
semaphore mutex = 1;                    /* controls access to critical region */
semaphore empty = N;                    /* counts empty buffer slots */
semaphore full = 0;                     /* counts full buffer slots */

void producer(void)
{
     int item;

     while (TRUE) {                     /* TRUE is the constant 1 */
          item = produce_item( );       /* generate something to put in buffer */
          down(&empty);                 /* decrement empty count */
          down(&mutex);                 /* enter critical region */
          insert_item(item);            /* put new item in buffer */
          up(&mutex);                   /* leave critical region */
          up(&full);                    /* increment count of full slots */
     }
}


void consumer(void)
{
     int item;

     while (TRUE) {                     /* infinite loop */
          down(&full);                  /* decrement full count */
          down(&mutex);                 /* enter critical region */
          item = remove_item( );        /* take item from buffer */
          up(&mutex);                   /* leave critical region */
          up(&empty);                   /* increment count of empty slots */
          consume_item(item);           /* do something with the item */
     }
}
```

Soluzione al Quesito 4
totale blocchi = 8 GB / 4kB = 2^33 / 2^12 = 2^21 blocchi; servono 21 bit per indirizzare ogni blocco -> record da 32 bit.
totale dimensione della FAT = 2^21 record per 4 byte (ovvero 32 bit) = 8388608 byte.

Soluzione al Quesito 5
Molte funzioni del Sistema Operativo sarebbero coinvolte (e stravolte) nel dover gestire un *hard-disk* di dimensione infinita. Lo studente è invitato a rivisitare criticamente il programma del corso provando a riflettere sulle modifiche necessarie.
Consideri ad esempio le implicazioni riguardo a:

- dimensione degli indirizzi
- utilizzabilità dei file system noti
- dimensione strutture dati
- tempi di spostamento della testina
- gestione dei blocchi liberi
- questione “frammentazione del disco”
- … (molto altro ancora)