Quesito 1 (punti 8). Cinque processi *batch*, identificati dalle lettere $A - E$ rispettivamente, arrivano all'elaboratore agli istanti $0, 2, 5, 6, 8$ rispettivamente. Tali processi hanno un tempo di esecuzione stimato di $6, 6, 4, 2, 2$ unità di tempo rispettivamente. Per ognuna delle seguenti politiche di ordinamento:

1. FPS (priorità esplicita e costante, con prerilascio)
2. RR (divisione di tempo, senza priorità e con quanto tempo di ampiezza 2)
3. SJF (senza considerazione di valori di priorità espliciti[1] e con prerilascio)

determinare, trascurando i ritardi dovuti allo scambio di contesto: (i) il tempo medio di risposta; (ii) il tempo medio di attesa; (iii) il tempo medio di *turn around*.
Ove la politica di ordinamento in esame consideri i valori di priorità, tali valori, mantenuti staticamente per l'intera durata dell'esecuzione, sono rispettivamente: $2, 3, 4, 5, 5$ (con 5 valore maggiore).
Nel caso di arrivi simultanei di processi allo stato di pronto, fatta salva l'eventuale considerazione del rispettivo valore di priorità, si dia la precedenza ai processi usciti dallo stato di esecuzione rispetto a quelli appena arrivati.

Quesito 2 (punti 8).
Si consideri un sistema composto da $m = 4$ risorse non prerilasciabili dello stesso tipo, condivise in mutua esclusione da $n = 3$ processi, ciascuno dei quali necessita simultaneamente di $k \leq 2$ risorse. Si dimostri che in un tale sistema non possono verificarsi situazioni di stallo.

Quesito 3 (punti 8). L'anomalia di Belady mostra che, con alcuni algoritmi di rimpiazzo delle pagine, l'aumento dei *page frame* disponibili può non portare a una diminuzione del numero di *page fault*. L'algoritmo ottimale è quello che non presenta l'anomalia di Belady e presenta invece il minor numero di *page fault*. È stato provato che l'algoritmo ottimale è quello che sostituisce la pagina che non si userà per il periodo di tempo più lungo. Malauguratamente, non ne esiste a oggi alcuna realizzazione concreta: si formuli qualche ipotesi intorno alle ragioni di tale situazione.
Ogni altro algoritmo di rimpiazzo è dunque necessariamente una approssimazione di quello ottimale. Così dovrebbe anche essere per l'algoritmo LRU (*least recently used*): si discuta se e come l'algoritmo LRU approssima quello ottimale.

Quesito 4 (punti 8). Si consideri un *file system* residente su una partizione di disco con dimensione dei blocchi logici e fisici di 512 B, dimensione dei *file* non superiori a 512 blocchi, e con tutte le informazioni su ciascun *file* già presenti in memoria principale. Per ciascuno dei tre metodi di allocazione visti a lezione (contigua, concatenata, indicizzata):

1. si illustri come gli indirizzi logici vengono fatti corrispondere agli indirizzi fisici
2. assumendo che l'ultimo accesso sia stato fatto al blocco logico 10, si determini quanti blocchi fisici debbano essere letti dal disco per accedere al blocco logico 4.

[1] Esclusi ovviamente i valori di priorità impliciti determinati dalla durata (residua) dei processi.

Soluzione 1 (punti 8).

- FPS (a priorità esplicita e costante, con prerilascio)

```
processo A  AAaaaaaaaaaaaaaaAAAA   LEGENDA DEI SIMBOLI
processo B  --BBBbbbbbbbbBBB       - non ancora arrivato
processo C  -----CccccCCC          x (minuscolo) attesa
processo D  ------DD               X (maiuscolo) esecuzione
processo E  --------EE             . coda vuota

CPU         AABBBCDDEECCCBBBAAAA
coda        ..aaabccccbbbaaa...
            .....abbbbbaaa......
            .......aaaa.........
```

processo	risposta	tempo di attesa	*turn-around*
A	0	0+14=14	14+6=20
B	0	0+ 8= 8	8+6=14
C	0	0+ 4= 4	4+4= 8
D	0	0+0=0	0+2= 2
E	0	0+0=0	0+2= 2
medie	0	5,2	9,2

- RR (a divisione di tempo, senza priorità e con quanto di ampiezza 2)

```
processo A  AAAAaaAA               LEGENDA DEI SIMBOLI
processo B  --bbBBbbbbBBbbbbbbBB   - non ancora arrivato
processo C  -----cccCCccccccCC     x (minuscolo) attesa
processo D  ------ddddddDD         X (maiuscolo) esecuzione
processo E  --------eeeeeeEE       . coda vuota

CPU         AAAABBAACCBBDDEECCBB
coda        ..bbaaccbbddeeccbb..
            .....cbbddeeccbb....
            ......ddeeccbb......
```

processo	risposta	tempo di attesa	*turn-around*
A	0	2	2+6= 8
B	2	2+4+6=12	12+6=18
C	3	3+6= 9	9+4=13
D	6	6	6+2= 8
E	6	6	6+2= 8
medie	3,4	7,0	11,0

- SJF (senza considerazione di valori di priorità espliciti e con prerilascio)

```
processo A  AAAAAA                 LEGENDA DEI SIMBOLI
processo B  --bbbbbbbbbbbbBBBBBB   - non ancora arrivato
processo C  -----cccccCCCC         x (minuscolo) attesa
processo D  ------DD               X (maiuscolo) esecuzione
processo E  --------EE             . coda vuota

CPU         AAAAAADDEECCCCBBBBBB
coda        ..bbbccccbbbb......
            .....bbbbb.........
```

processo	tempo di risposta	tempo di attesa	tempo di *turn-around*
A	0	0	0+6= 6
B	12	12	12+6=18
C	5	5	5+4= 9
D	0	0	0+2= 2
E	0	0	0+2= 2
medie	3,40	3,40	7,40

Soluzione 2 (punti 8). Come sappiamo (cf. diapositiva 67 della lezione S01 dell'a.a. 2006/7), 4 condizioni devono essere simultaneamente soddisfatte per il verificarsi di una situazione di stallo:

1. accesso esclusivo a risorse condivise
2. accumulo di risorse
3. inibizione di prerilascio
4. attesa circolare.

Dal quesito sappiamo che le condizioni $1-3$ sono soddisfatte per definizione. Resta quindi solo da studiare se la condizione 4 si verifichi nel problema posto dal quesito.
È facile osservare che la condizione di attesa circolare che provochi uno stallo completo del sistema si verifica solamente nel caso in cui tutti i processi abbiano ottenuto accesso esclusivo a $t = k - 1 = 1$ risorse ma nessuno di essi abbia ottenuto tutte le risorse richieste: ciò non può verificarsi nel caso in questione perché il numero m di risorse disponibili è superiore al numero dei processi n e dunque vi sarà sempre 1 risorsa supplementare a disposizione di un processo richiedente, che potrà pertanto completare l'esecuzione e rimettere le sue risorse a disposizione degli altri processi, che potranno pertanto procedere, completare e fare lo stesso.

Soluzione 3 (punti 8). L'algoritmo ottimale è chiaramente predittivo, in quanto si basa sul comportamento del sistema previsto per il futuro prossimo. In questo senso è simile alla politica *Shortest Job First* per l'ordinamento dei processi, che attribuisce priorità implicita superiore ai processi a minor durata d'esecuzione futura. Per ogni programma realistico però sia la successione effettiva dei riferimenti a pagine che la durata reale di una esecuzione dipendono da fattori non prevedibili con certezza a priori (e.g.: la scelta del cammino d'esecuzione): per questo motivo né l'algoritmo ottimale di rimpiazzo né la politica SJF sono realizzabili in concreto.
L'algoritmo LRU è retrospettivo, in quanto si basa sul comportamento del sistema osservato nel passato prossimo. LRU rimpiazza infatti la pagina che non è stata usata per il periodo più lungo nel passato, il che è senza dubbio una approssimazione dell'algoritmo ottimale visto che non è certo (ma è solo statisticamente improbabile) che la pagina meno usata nella finestra temporale di osservazione non debba essere usata nel prossimo futuro.

Soluzione 4 (punti 8). Si vedano le diapositive $211-220$ della lezione S05-2 dell'a.a. 2006/7.

Corrispondenza tra indirizzi logici e indirizzi fisici :

Allocazione contigua : il *file* è denotato dall'indice del primo blocco fisico e dalla sua ampiezza in blocchi; vista la corrispondenza di ampiezza tra blocchi logici e fisici, ogni posizione interna al *file* (blocco logico e *offset* in esso) ha una corrispondenza diretta sul disco (blocco fisico e *offset*).

Allocazione concatenata : il *file* è denotato dagli indici del primo e dell'ultimo blocco fisico; una parte dei dati di ogni blocco contiene il puntatore al blocco successivo. La posizione interna al *file* espressa in (blocco logico i, *offset* o) viene dunque tradotta mediante l'attraversamento di i posizioni nella lista concatenata a partire dalla testa.

Allocazione indicizzata : il *file* è denotato da un blocco speciale (detto appunto "indice"), che contiene gli indici dei blocchi fisici ove risiedono i dati. La posizione interna al *file* espressa in (blocco logico i, *offset* o) viene dunque tradotta localizzando il blocco fisico denotato dalla posizione i entro il blocco indice e la posizione o al suo interno. (Come noto, il blocco indice può essere realizzato come una tabella concatenata, tipo FAT, oppure come un blocco contiguo dedicato, tipo *i-node*.)

Blocchi fisici acceduti per procedere dal blocco 10 al blocco 4 :

Allocazione contigua : 1 (direttamente il blocco 4).

Allocazione concatenata : 4 (fino al blocco 4 a partire dalla testa della lista).

Allocazione indicizzata : 1 (direttamente il blocco 4, ma solo in virtù dell'ipotesi favorevole del quesito per la quale la dimensione massima del *file* sia interamente rappresentabile con un singolo blocco indice).