ΛΥΜΕΝΑ ΠΡΟΒΛΗΜΑΤΑ

Πρόβλημα 1:

Το Πανεπιστήμιο Μακεδονίας, εφαρμόζοντας την αρχή της ισότητας μεταξύ των δύο φύλων μετέτρεψε τις τουαλέτες των φοιτητών σε κοινές τουαλέτες. Προς αποφυγή όμως παρεξηγήσεων σχεδιάζεται να ορισθεί ότι όταν είναι μέσα μία γυναίκα, μπορούν να μπουν μόνο γυναίκες, και αντίστοιχα όταν είναι μέσα άνδρας μπορούν να μπουν μόνο άνδρες. Μια φωτεινή επιγραφή στην είσοδο θα προσδιορίζει σε ποιά από τις τρεις καταστάσεις βρίσκεται η τουαλέτα:

- Είναι άδεια
- Υπάρχουν μέσα γυναίκες
- Υπάρχουν μέσα άνδρες

Γράψτε τις επόμενες διαδικασίες σε όποια γλώσσα προγραμματισμού προτιμάτε:

α) γυναίκα_θέλει_να_μπεί

β) άνδρας_θέλει_να_μπεί

γ) γυναίκα_αποχωρεί

δ) άνδρας_αποχωρεί

Μπορείτε να χρησιμοποιήσετε μετρητές και όσους σηματοφόρους χρειάζεστε.

Λύση:

Το πρόβλημα αυτό μοιάζει πολύ με το πρόβλημα των αναγνωστών/συγγραφέων. Σε αντίθεση με εκείνο που επέτρεπε είτε πολλούς αναγνώστες είτε έναν συγγραφέα να μπαίνουν στην βάση δεδομένων, τώρα έχουμε δύο ομάδες χρηστών που είναι αμοιβαίως αποκλειόμενες. Μπορούμε επομένως να χρησιμοποιήσουμε ρουτίνες αντίστοιχες με του αναγνώστη για να επιτύχουμε τα ζητούμενα από το πρόβλημα:

```
semaphore mutex=1, mutex2, restroom=1
int mc=0, wc=0

γυναίκα_θέλει_να_μπει()
   {down(mutex1)
   wc=wc+1
   if (wc=1) down(restroom)
   up(mutex1)
   }

γυναίκα_θέλει_να_βγει()
   {down(mutex1)
   wc=wc-1
   if (wc=0) up(db)
```

```
   up(mutex1)
   }

άνδρας_θέλει_να_μπει()
   {down(mutex2)
   mc=mc+1
   if (mc=1) down(restroom)
   up(mutex2)
   }

άνδρας_θέλει_να_βγει()
   {down(mutex2)
   mc=mc-1
   if (mc=0) up(db)
   up(mutex2)
   }
```

Πρόβλημα 2:

Δίδεται η παρακάτω κατάσταση ενός συστήματος:

	Δοσμένοι πόροι	Μέγιστο αιτήσεων	Διαθέσιμοι πόροι
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Απαντήστε στις παρακάτω ερωτήσεις χρησιμοποιώντας τον αλγόριθμο αποφυγής αδιεξόδου:

α. Ποιά είναι τα περιεχόμενα του πίνακα "Απομένουσες αιτήσεις"

β. Είναι το σύστημα σε αδιέξοδο;

γ. Αν έλθει από τη διαδικασία P_1 η αίτηση (0,4,2,0), μπορεί να εξυπηρετηθεί αμέσως;

Λύση:

α) Για να βρούμε τον πίνακα των απομενουσών αιτήσεων δεν έχουμε παρά να αφαιρέσουμε τους δοσμένους πόρους από τις μέγιστες αιτήσεις. Παίρνουμε λοιπόν:

	Απομένουσες αιτήσεις
	A B C D
P_0	0 0 0 0
P_1	0 7 5 0
P_2	1 0 0 2

P_3	0 0 2 0	
P_4	0 6 4 2	

β) Για να ελέγξουμε αν το σύστημα είναι σε αδιέξοδο, πρέπει να ελέγξουμε την ασφάλεια του συστήματος. Οι τρεις πίνακες που χρησιμοποιούνται είναι οι «Δοσμένοι Πόροι» οι «Απομένουσες Αιτήσεις» και οι «Διαθέσιμοι πόροι»

	Δοσμένοι πόροι	Απομένουσες αιτήσεις	Διαθέσιμοι πόροι
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 0 0	1 5 2 0
P_1	1 0 0 0	0 7 5 0	
P_2	1 3 5 4	1 0 0 2	
P_3	0 6 3 2	0 0 2 0	
P_4	0 0 1 4	0 6 4 2	

Μετά το μαρκάρισμα της P_0 και της P_3 που είναι εμφανή έχουμε:

	Δοσμένοι πόροι	Απομένουσες αιτήσεις	Διαθέσιμοι πόροι
	A B C D	A B C D	A B C D
• P_0	0 0 0 0	0 0 0 0	1 11 6 4
P_1	1 0 0 0	0 7 5 0	
P_2	1 3 5 4	1 0 0 2	
• P_3	0 0 0 0	0 0 2 0	
P_4	0 0 1 4	0 6 4 2	

Είναι προφανές τώρα ότι όλες οι γραμμές του πίνακα «Απομενουσών αιτήσεων» είναι μικρότερες ή ίσες των «διαθεσίμων πόρων», επομένως το σύστημα **δεν** είναι σε αδιέξοδο.

γ) Η P_1 έχει το δικαίωμα να ζητήσει αυτούς τους πόρους **και** οι πόροι είναι διαθέσιμοι. Προσποιούμαστε λοιπόν ότι δίνουμε τούς πόρους:

	Δοσμένοι πόροι	Απομένουσες αιτήσεις	Διαθέσιμοι πόροι
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 0 0	1 1 0 0
P_1	1 4 2 0	0 3 3 0	
P_2	1 3 5 4	1 0 0 2	
P_3	0 6 3 2	0 0 2 0	
P_4	0 0 1 4	0 6 4 2	

Δεν έχουμε παρά να ελέγξουμε το σύστημα για ασφάλεια. Πρώτα μαρκάρεται η P_0.

	Δοσμένοι πόροι	Απομένουσες αιτήσεις	Διαθέσιμοι πόροι
	A B C D	A B C D	A B C D
• P_0	0 0 0 0	0 0 0 0	1 1 1 2
P_1	1 0 0 0	0 7 5 0	
P_2	1 3 5 4	1 0 0 2	
P_3	0 6 3 2	0 0 2 0	

P_4	0 0 1 4	0 6 4 2

Τώρα μπορεί να μαρκαριστεί η P_2.

	Δοσμένοι πόροι	Απομένουσες αιτήσεις	Διαθέσιμοι πόροι
	A B C D	A B C D	A B C D
• P_0	0 0 0 0	0 0 0 0	2 4 6 6
P_1	1 0 0 0	0 7 5 0	
• P_2	0 0 0 0	1 0 0 2	
P_3	0 6 3 2	0 0 2 0	
P_4	0 0 1 4	0 6 4 2	

Τώρα μπορεί να μαρκαριστεί η P_3.

	Δοσμένοι πόροι	Απομένουσες αιτήσεις	Διαθέσιμοι πόροι
	A B C D	A B C D	A B C D
• P_0	0 0 0 0	0 0 0 0	2 10 9 10
P_1	1 0 0 0	0 7 5 0	
• P_2	0 0 0 0	1 0 0 2	
• P_3	0 0 0 0	0 0 2 0	
P_4	0 0 1 4	0 6 4 2	

Είναι προφανές τώρα ότι όλες οι γραμμές του πίνακα «Απομενουσών αιτήσεων» είναι μικρότερες ή ίσες των «διαθεσίμων πόρων», επομένως το σύστημα **δεν** κινδυνεύει να βρεθεί σε αδιέξοδο. Άρα, δίνουμε τον πόρο.

Πρόβλημα 3:

Δίνονται οι παρακάτω ορισμοί:

R= Χρόνος που απαιτείται από τον βραχυπρόθεσμο χρονοδρομολογητή κάθε φορά που τρέχει (χρόνος μεταγωγής διεργασιών)

N= Μέγιστος αριθμός έτοιμων (ready) διεργασιών για την δρομολόγηση

T= Μέγιστος χρόνος που επιτρέπεται μέχρι να έρθει η σειρά μιας έτοιμης (ready) διαδικασίας να τρέξει.

Γράψτε την εξίσωση που δίνει το κβάντο χρόνου ***Q*** ως συνάρτηση των ***R***, ***N*** και ***T***.

Λύση:

Σύμφωνα με τα δεδομένα του προβλήματος, σε χρόνο ***T*** θα πρέπει να έχουνε τρέξει όλες οι διεργασίες έστω κατ΄ ελάχιστον, δηλαδή να έχει πάρει κβάντο κάθε διεργασία. Ο χρόνος που καταναλώνεται για κάθε διεργασία, περιλαμβάνει εκτός από το κβάντο, και τον χρόνο μεταγωγής των διεργασιών. Επομένως, ο συνολικός χρόνος θα δίνεται από την εξίσωση:

$$T = (Q + R)N$$

Λύνοντας ως προς ***Q*** παίρνουμε:

$$Q = \frac{T}{N} - R$$

Πρόβλημα 4:

Ένα σύστημα χρησιμοποιεί έναν αλγόριθμο με προτεραιότητες (priorities) γιά την χρονοδρομολόγηση, σε συνδυασμό με τον Round Robin. Δηλαδή μόνο οι διεργασίες που έχουν την υψηλότερη προτεραιότητα και είναι «έτοιμες», μοιράζονται κάθε φορά τον χρόνο της CPU. Το σύστημα στον χρόνο 0 είναι κενό (idle). Έρχονται πέντε εργασίες για εκτέλεση με τα παρακάτω χαρακτηριστικά:

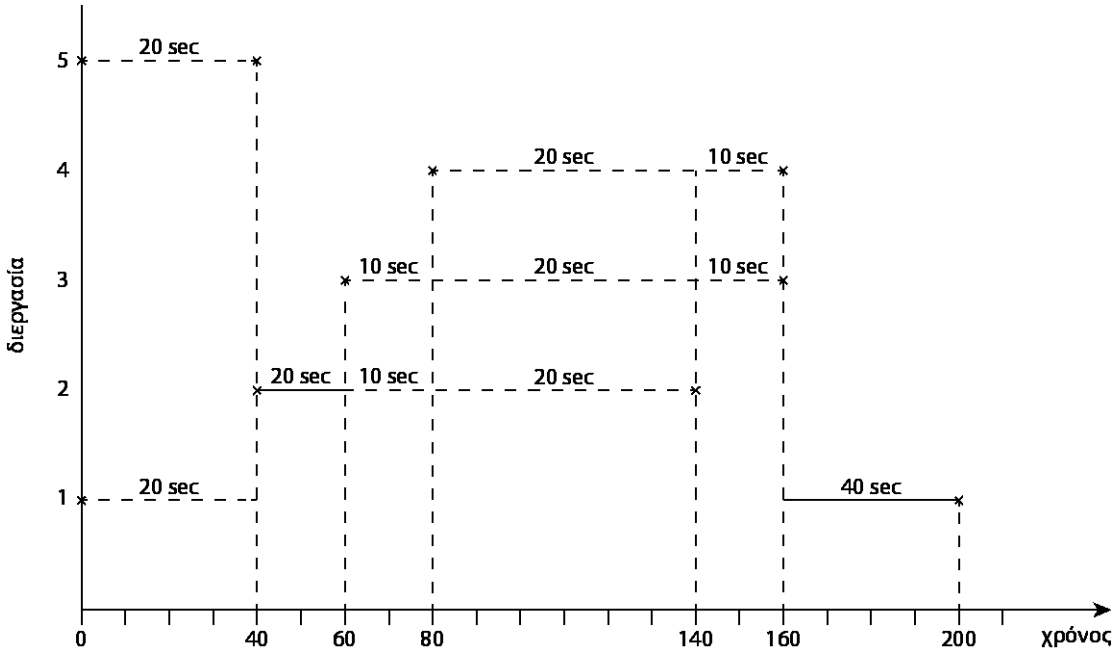
Διεργασία	Priority	Χρόνος Άφιξης	Χρόνος Εκτέλεσης
1	4	0 sec	60 sec
2	2	40 sec	50 sec
3	2	60 sec	40 sec
4	2	80 sec	30 sec
5	4	0 sec	20 sec

Δώστε το χρονοδιάγραμμα εκτέλεσης των διεργασιών καθώς και τους Average Turnaround Time (Μέσο Χρόνο Επιστροφής) και Weighted ATM (Σταθμισμένο Μέσο Χρόνο Επιστροφής).

Σημείωση: Για τις προτεραιότητες, μικρότερο νούμερο σημαίνει μεγαλύτερη προτε-ραιότητα.

Λύση:

Οι διεργασίες αρχίζουν να εκτελούνται με σειρά προσέλευσης, εφ΄ όσον έχουν την ίδια προτεραιότητα. Όταν έρχονται διεργασίες με υψηλότερη προτεραιότητα, τότε παίρνουν αυτές αποκλειστική χρήση της CPU. Το χρονοδιάγραμμα εκτέλεσης είναι:



Οι χρόνοι επιστροφής είναι:

TT_1=200 sec
TT_2=100 sec
TT_3=100 sec
TT_4=80 sec
TT_5=40 sec

Ο μέσος χρόνος επιστροφής είναι, επομένως:

$$ATT = \frac{200 + 100 + 100 + 80 + 40}{5} = \frac{520}{5} = 105 \text{ sec}$$

Οι σταθμισμένοι χρόνοι επιστροφής είναι:

WTT_1=200/60=3.33
WTT_2=100/50=2
WTT_3=100/40=2.5
WTT_4=80/30=2.66
WTT_5=40/20=2

Και ο Σταθμισμένος Μέσος Χρόνος Επιστροφής είναι:

$$AWTT = \frac{3.33 + 2 + 2.5 + 2.66 + 2}{5} = \frac{12.5}{5} = 2.5$$

Πρόβλημα 5:

Μετρήσεις σε συγκεκριμένο σύστημα έχουν δείξει, ότι μια μέση διεργασία εκτελείται για χρόνο *T* πριν ανασταλεί για είσοδο/έξοδο. Η εναλλαγή διεργασίας απαιτεί χρόνο *E*, ο οποίος θεωρείται χαμένος χρόνος επειδή η εναλλαγή λογίζεται ως επιβάρυνση. Για

χρονοπρογραμματισμό εξυπηρέτησης εκ περιτροπής με κβάντο χρόνου *K*, δώστε τον τύπο αποδοτικότητας της CPU, για κάθε μία από τις ακόλουθες περιπτώσεις:

α) $K = \infty$

β) $K > T$

γ) $E < K < T$

δ) $K = E$

ε)$K \cong 0$

Λύση:

α) Αφού το κβάντο είναι τεράστιο κάθε διεργασία θα διακοπεί όταν ζητήσει I/O, δηλαδή μετά από χρόνο *T*. Επομένως:

$$U = \frac{T}{T + E}$$

β) Αφού το κβάντο είναι μεγαλύτερο του χρόνου που θα τρέξει χωρίς διακοπή μια διεργασία, κάθε διεργασία θα διακοπεί όταν ζητήσει I/O, δηλαδή μετά από χρόνο *T*. Επομένως και πάλι:

$$U = \frac{T}{T + E}$$

γ) Αφού το κβάντο είναι μικρότερο από *T*, κάθε διεργασία θα συμπληρώσει ένα κβάντο πριν διακοπεί. Επομένως:

$$U = \frac{K}{K + E}$$

δ) Το κβάντο υποτίθεται ότι είναι μικρότερο του T, αλλιώς θα είχαμε την περίπτωση (β). Έχουμε λοιπόν την περίπτωση (γ) με ίσα τα *K* και *E*. Επομένως:

$$U = 50\%$$

α) Αφού το κβάντο είναι σχεδόν ίσο με το 0, έχουμε την περίπτωση (γ) με πάρα πολύ μικρό το K. Επομένως:

$$U \cong 0$$

Πρόβλημα 6:

Πέντε διεργασίες περιμένουν να εκτελεστούν. Οι αναμενόμενοι χρόνοι εκτέλεσής τους είναι 25, 20, 12, 18 και X. Με ποιά σειρά πρέπει να εκτελεστούν ώστε να ελαχιστοποιηθεί ο μέσος χρόνος επιστροφής (average turnaround time); (Η απάντησή σας πρέπει να είναι συνάρτηση του X).

Λύση:

Για τον μέσο χρόνο επιστροφής θα έχουμε ένα κλάσμα του οποίου ο παρονομαστής θα είναι σταθερά 5 ανεξάρτητα από τη σειρά εκτέλεσης. Πρέπει λοιπόν να ελαχιστοποιηθεί ο αριθμητής, ο οποίος είναι το άθροισμα των χρόνων επιστροφής. Λαμβάνοντας ως χρόνο έναρξης το 0, αν η σειρά εκτέλεσης είναι p1, p2, p3, p4, και p5, όπου το ποιά διεργασία είναι η κάθε μία μένει να διευκρινισθεί , τότε ο συνολικός χρόνος επιστροφής θα είναι:

$$T = 5T_1 + 4T_2 + 3T_3 + 2T_4 + T_5$$

Το άθροισμα αυτό ελαχιστοποιείται αν οι χρόνοι εκτέλεσης είναι με αύξουσα σειρά έτσι ώστε οι μικρότεροι χρόνοι να πολλαπλασιάζονται με μεγαλύτερους συντελεστές. Επομένως η εκτέλεση θα πρέπει να είναι με αύξουσα σειρά χρόνων ως εξής

$$Σειρά = \begin{cases} X, 12,18,20,25 & \text{αν } X \leq 12 \\ 12, X, 18,20,25 & \text{αν } 12 < X \leq 18 \\ 12,18, X, 20,25 & \text{αν } 18 < X \leq 20 \\ 12,18,20, X, 25 & \text{αν } 20 < X \leq 25 \\ 12,18,20,25, X & \text{αν } 25 < X \end{cases}$$

Πρόβλημα 7:

Υπολογίστε την εξίσωση που δίνει τον φαινόμενο χρόνο προσπέλασης της μνήμης σε συστήματα σελιδοποίησης, στα οποία η μετάφραση της ιδεατής σε φυσική διεύθυνση γίνεται με τη βοήθεια ενός TLB. Να εκφράσετε το φαινόμενο χρόνο προσπέλασης ως συνάρτηση του χρόνου προσπέλασης του TLB, του hit ratio του TLB (ποσοστό των μεταφράσεων που επιτυγχάνονται μέσω του TLB και δεν χρειάζεται να συμβουλευτούμε τον πίνακα σελίδων που βρίσκεται στην κυρίως μνήμη), και του χρόνου προσπέλασης της μνήμης. Συγκρίνετε το φαινόμενο αυτό χρόνο με το χρόνο προσπέλασης της μνήμης χωρίς διαχείριση μνήμης. Σχολιάστε τις τιμές των παραμέτρων σε σχέση με το αν η χρήση του TLB συμφέρει ή όχι.

Λύση:

Έστω:

t_t : Χρόνος προσπέλασης του TLB

t_m : Χρόνος προσπέλασης της μνήμης

T_t : Συνολικός χρόνος προσπέλασης με TLB

T_w : Συνολικός χρόνος προσπέλασης χωρίς TLB

Όταν έχουμε επιτυχία (hit), διαβάζεται μια φορά το TLB και μία φορά η μνήμη (δεδομένο).

Όταν έχουμε αποτυχία διαβάζεται μια φορά το TLB και δύο φορές η μνήμη (πίνακας και δεδομένο).

Όταν δεν έχουμε TLB διαβάζεται πάντα δυο φορές η μνήμη.

Επομένως:

$$T_t = t_t + t_m + (1-h)t_m$$

και

$$T_w = 2t_m$$

Το TLB συμφέρει όταν $T_t<T_w$. Δηλαδή όταν:

$$T_t < T_w \Rightarrow$$

$$t_t + t_m + (1-h)t_m < 2t_m \Rightarrow$$

$$t_t < ht_m \Rightarrow$$

$$\frac{t_t}{t_m} < h$$

Πρόβλημα 8:

Δίνεται η παρακάτω σειρά αναφορών στη μνήμη από ένα πρόγραμμα των 460 bytes (όλοι οι αριθμοί στο δεκαδικό):

10, 11, 104, 170, 73, 309, 185, 245, 246, 434, 458, 364

1. Δώστε τη σειρά (λίστα) αναφορών στις σελίδες υποθέτοντας μέγεθος σελίδας 100 bytes.
2. Δώστε τα σφάλματα σελίδας για τη σειρά αυτή υποθέτοντας ότι δίνονται 200 bytes μνήμης στη διεργασία αυτή και χρησιμοποιείται ο αλγόριθμος FIFO.
3. Επαναλάβετε το (2) για τον αλγόριθμο LRU (λιγότερο πρόσφατα χρησιμοποιημένης)
4. Επαναλάβετε το (2) για τον βέλτιστο αλγόριθμο

Λύση:

1. Αφού κάθε σελίδα έχει μέγεθος 100 bytes οι διευθύνσεις που δίνονται αντιστοιχούν στις σελίδες: 0, 0, 1, 1, 0, 3, 1, 2, 2, 4, 4, 3. Αν αγνοήσουμε τις συνεχείς αναφορές στην ίδια σελίδα, η σειρά γίνεται: 0, 1, 0, 3, 1, 2, 4, 3.

2. Τα 200 bytes που δίνονται στην διεργασία αντιστοιχούν σε δύο πλαίσια. Επομένως, δεν μπορούν να βρίσκονται πάνω από 2 σελίδες στη μνήμη. Θεωρώντας ότι αρχικά η μνήμη είναι άδεια, τα πλαίσια γεμίζουν ως ακολούθως:

Αίτηση	0	1	0	3	1	2	4	3
Πλαίσιο 1	⓪	0	0	③	3	3	④	4
Πλαίσιο 2		①	1	1	1	②	2	③
Σφάλμα	*	*		*		*	*	*

όπου τα σφάλματα σελίδας υποδηλώνονται με αστερίσκο και οι παλαιότερες σελίδες δίνονται μέσα σε κύκλο. Έχουμε λοιπόν 6 σφάλματα σελίδας.

3. Για τον αλγόριθμο LRU έχουμε αντιστοίχως:

Αίτηση	0	1	0	3	1	2	4	3
Πλαίσιο 1	⓪	0	⓪	0	①	1	④	4
Πλαίσιο 2		①	1	③	3	②	2	③
Σφάλμα	*	*		*	*	*	*	*

όπου τα σφάλματα σελίδας υποδηλώνονται με αστερίσκο και οι πιο πρόσφατα χρησιμοποιημένες σελίδες δίνονται μέσα σε κύκλο. Έχουμε λοιπόν 7 σφάλματα σελίδας.

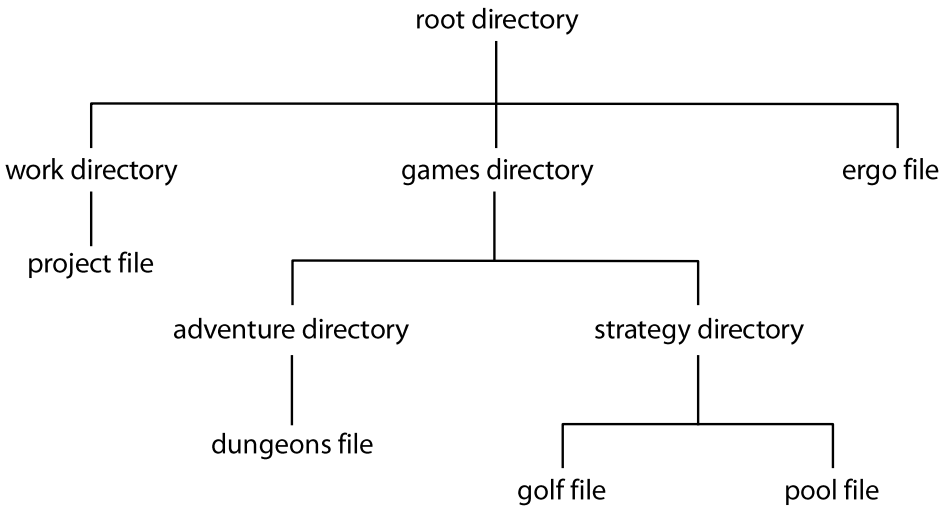
4. Για τον βέλτιστο αλγόριθμο έχουμε αντιστοίχως:

Αίτηση	0	1	0	3	1	2	4	3
Πλαίσιο 1	⓪	⓪	⓪	③	3	3	3	3
Πλαίσιο 2		1	1	1	①	②	4	4
Σφάλμα	*	*		*		*	*	

όπου τα σφάλματα σελίδας υποδηλώνονται με αστερίσκο και οι πιο πρόσφατα χρησιμοποιημένες σελίδες δίνονται μέσα σε κύκλο. Έχουμε λοιπόν 5 σφάλματα σελίδας.

Πρόβλημα 9:

Να δώσετε τους κόμβους-δ και data blocks για το παρακάτω δένδρο (καθορίστε εσείς τα νούμερα των κόμβων-δ και των blocks):



Λύση:

Θέτοντας τυχαίους αριθμούς για τους αριθμούς των κόμβων-δ και τον blocks και ξεκινώντας από τον κατάλογο root παίρνουμε:

root dir

1	.
1	..
45	work
23	games
89	ergo

κόμβος-δ 45

type:dir owner:...
534

block 534

45	.
1	..
93	project

κόμβος-δ 23

type:dir owner:...
396

block 396

23	.
1	..
12	adventure
78	strategy

κόμβος-δ 89

type:file owner:...
239

block 239

(δεδομένα του αρχείου ergo)

κόμβος-δ 12

type:dir owner:...
711

block 711

12	.
23	..
59	dungeons

κόμβος-δ 78

type:dir owner:...
344

block 344

78	.
23	..
129	golf
49	pool

κόμβος-δ 59

type:file owner:...
396

block 396

(δεδομένα του αρχείου dungeons)

κόμβος-δ 129

type:file owner:...
932

block 932

(δεδομένα του αρχείου golf)

κόμβος-δ 49

type:file owner:...
1645

block 1645

(δεδομένα του αρχείου pool

Πρόβλημα 10:

Υποθέστε ότι μια διεργασία ζητά τη μεταφορά 500 τομέων (sectors) από ένα δίσκο. Να εξετάσετε δύο διαφορετικές προσεγγίσεις: έναν ελεγκτή DMA που διαβάζει έναν τομέα και διακόπτει (με interrupt) τη CPU αφού τον διαβάσει, και έναν επεξεργαστή εισόδου/εξόδου που χειρίζεται τις λειτουργίες και για τους 500 τομείς. Υποθέστε ότι χρειάζεται 0.5 ms για την CPU να ανταποκριθεί σε κάθε interrupt και να δώσει εντολή για το επόμενο διάβασμα. Πόσο χρόνο κερδίζει η CPU με τη δεύτερη προσέγγιση; Πόσες εντολές θα μπορούσε να είχε εκτελέσει σ' αυτό τον χρόνο. Υποθέστε ότι η CPU εκτελεί μία εντολή κάθε 500 nanoseconds.

Λύση:

$1^{η}$ προσέγγιση: Η CPU πρέπει να διακοπεί 500 φορές αφιερώνοντας 0.5 ms κάθε φορά για την εξυπηρέτηση της διακοπής. Απασχολείται δηλαδή συνολικά 0.5X500= 250 ms.

$2^{η}$ προσέγγιση: Η CPU διακόπτεται μόνο μία φορά από τον ελεγκτή Ε/Ε όταν έχει τελειώσει όλη η διαδικασία. Απασχολείται λοιπόν συνολικά μόνο για 0.5 ms.

Με την δεύτερη προσέγγιση η CPU κερδίζει 250ms – 0.5 ms = 249.5 ms

Στον χρόνο αυτό μπορεί να εκτελέσει:

$$N = \frac{249.5 \text{ ms}}{500 \text{ ns/εντολή}} = 499000 \text{ εντολές}$$