ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

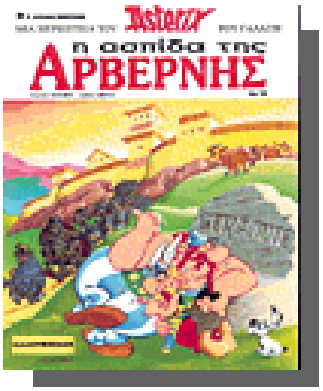
Εαρινό 2007

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 2

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής αποστολής: 30/8/2007, 12:30

Εμπορικές συναλλαγές στην Αρβέρνη (0.1 + 0.3 = 0.4 βαθμοί)



Όπως είναι ευρέως γνωστό από τον Αστερίξ και συγκεκριμένα από το κόμικ "Η ασπίδα της Αρβέρνης", στη Γαλατία υπάρχει μια πόλη με μία μόνο ευθεία οδό στην οποία όλοι οι κάτοικοι έχουν από ένα οινοπωλείο. Πιθανώς να αναρωτιέστε πως στο καλό δουλεύει η οικονομία της συγκεκριμένης πόλης. Η απάντηση είναι: σχετικά απλά. Κάθε κάτοικος που ξεμένει από το δικό του κρασί αγοράζει κρασί από κάποιο άλλο οινοπωλείο της συγκεκριμένης οδού. Κάθε πρωί, ο κάθε κάτοικος αποφασίζει πόσο κρασί θέλει να αγοράσει ή να πουλήσει. Το περίεργο της υπόθεσης είναι ότι η συνολική ποσότητα τόσο της προσφοράς όσο και της ζήτησης είναι πάντα ίσες ανά μέρα, οπότε ο κάθε κάτοικος μπορεί πάντα να αγοράσει ή να πουλήσει ότι επιθυμεί.

Υπάρχει όπως ένα πρόβλημα. Η μεταφορά κρασιού από οινοπωλείο σε οινοπωλείο έχει κόπο ανάλογο της απόστασης μεταξύ των δύο μαγαζιών, μια μονάδα "κόπου" για μεταφορά από ένα μαγαζί στο γειτονικό του, δύο για μεταφορά στο αμέσως επόμενο, κοκ. Όμως, από τη στιγμή που όλα τα κρασιά είναι εξίσου καλά, οι κάτοικοι-αγοραστές δεν ενδιαφέρονται από ποιόν πωλητή θα αγοράσουν. Επιπλέον, είναι αρκετά έξυπνοι ώστε να οργανώσουν την αγοραπωλησία με τέτοιο τρόπο που ο συνολικός κόπος μεταφοράς του κρασιού να είναι ο ελάχιστος δυνατός.

Το ζητούμενο της άσκησης είναι να γραφούν προγράμματα που να αυτοματοποιούν τη διαδικασία της συναλλαγής ανά μέρα και να υπολογίζουν το συνολικό "κόπο" μεταφοράς. Για απλοποίηση, θεωρείστε ότι όλα τα οινοπωλεία βρίσκονται στη μία μόνο πλευρά της οδού της πόλης.

Θέλουμε δύο προγράμματα: ένα σε C με χρήση πίνακα και ένα σε Java με χρήση αντικειμένων (που φυλάσσονται σε μια συλλογή) όπου το κάθε αντικείμενο αντιστοιχεί σε ένα οινοπωλείο, έχει ως πεδία τη θέση του στην οδό και την ποσότητα του κρασιού την οποία ο ιδιοκτήτης του θέλει να αγοράσει ή να πουλήσει, και μεθόδους για αγοραπωλησία. Και τα δύο προγράμματα διαβάζουν την είσοδό τους από ένα αρχείο (`asterix.input`) και επιστρέφουν την έξοδό τους στην οθόνη. Το αρχείο εισόδου έχει πολλές γραμμές κάθε ζεύγος των οποίων αντιστοιχεί στο πλήθος και στις επιθυμίες αγοραπωλησίας μιας συγκεκριμένης μέρας. Για παράδειγμα, το παρακάτω αρχείο εισόδου (πάνω από τη γραμμή) λέει ότι την πρώτη μέρα υπάρχουν 5 οινοπωλεία ανοιχτά, όπου το ο ιδιοκτήτης του πρώτου θέλει να πουλήσει 5 μπουκάλια κρασί, του δεύτερου θέλει να αγοράσει 4, του τρίτου και του πέμπτου να πουλήσουν 1, και του τέταρτου να αγοράσει 3. Τη δεύτερη μέρα τα οινοπωλεία είναι 7. Η τελευταία μέρα δηλώνεται με ένα 0 που σημαίνει ότι τελείωσαν οι αγοραπωλησίες.

5 5 -4 1 -3 1 7 -1000 -1000 -1000 0 1000 1000 1000 0	**asterix.input**
9 12000	**έξοδος προγράμματος**

Κάτω από την οριζόντια γραμμή δείχνουμε την έξοδο του προγράμματος για τη συγκεκριμένη είσοδο. Παρακαλούμε ονομάστε τα κύριά σας αρχεία **asterix.c** και **Asterix.java.**

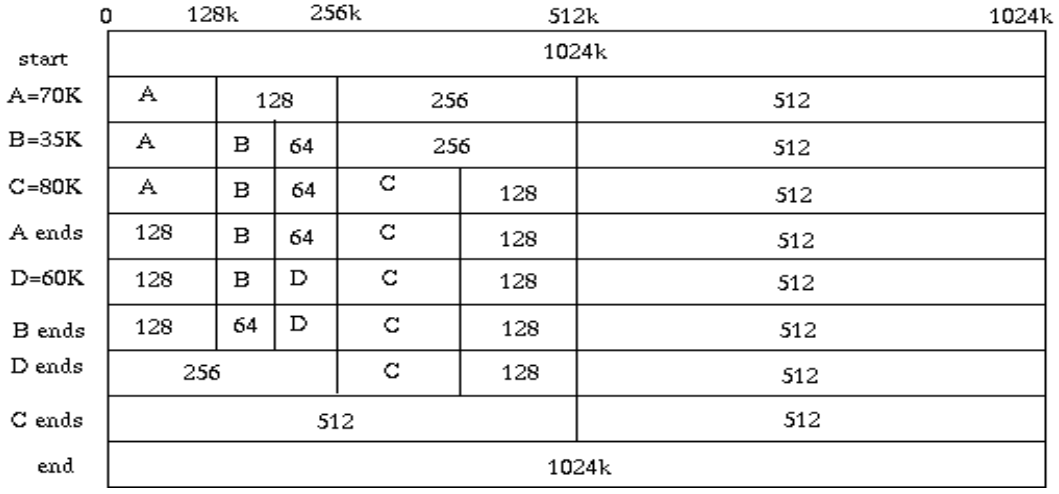
Διαχείριση μνήμης με "κολλητούς" (0.25 βαθμοί)

Η διαχείριση μνήμης, και ειδικά η παραχώρηση μνήμης σε διεργασίες, είναι θεμελιώδης λειτουργία των λειτουργικών συστημάτων. Εάν η διαίρεση της μνήμης σε μπλοκ γίνει με κάποιο στατικό και μη ευέλικτο τρόπο, τότε είναι πιθανό η μνήμη να μη χρησιμοποιηθεί αποδοτικά και ο αριθμός των διεργασιών να περιοριστεί. Αντίθετα, σε ένα σύστημα στο οποίο τα μεγέθη των μπλοκ καθορίζονται δυναμικά, η διαχείριση είναι μεν πιο πολύπλοκη (διότι πρέπει να διατηρηθούν διαφορετικές λίστες ελεύθερων μπλοκ και το σύστημα θα πρέπει να είναι σε θέση να κάνει συμπίεση) αλλά προβλήματα τέτοιου είδους αποφεύγονται. Μια ενδιαφέρουσα ενδιάμεση λύση είναι το σύστημα των "κολλητών" (buddy system).

Σε ένα σύστημα buddy, όλη η προς παραχώρηση μνήμη ξεκινάει ως ένα αδιαίρετο μπλοκ μεγέθους κάποιας δύναμης του 2. Εάν η πρώτη αίτηση ζητήσει ένα μπλοκ μεγέθους μεγαλύτερου του μισού αρχικού, τότε η αίτηση εξυπηρετείται με την παραχώρηση ολόκληρου του μπλοκ. Αλλιώς, το μπλοκ χωρίζεται στη μέση σε δύο "κολλητά" μπλοκ ίδιου μεγέθους. Εάν το μέγεθος της απαίτησης είναι μεγαλύτερο από το μισό του μεγέθους των κολλητών που μόλις δημιουργήθηκαν, τότε κάποιο κολλητό μπλοκ παραχωρείται ως έχει. Αλλιώς, ένα από τα μπλοκ χωρίζεται ξανά στη μέση σε δύο νέα κολλητά μπλοκ και η μέθοδος επαναλαμβάνεται μέχρι να ικανοποιηθεί η αίτηση.

Στη συγκεκριμένη μέθοδο, όταν μια διεργασία τερματιστεί τα μπλοκ που της έχουν παραχωρηθεί ελευθερώνονται. Όποτε είναι δυνατό τα μπλοκ συνενώνονται ξανά με τον κολλητό τους (αυτόν από τον οποίο δημιουργήθηκαν) ώστε να δημιουργήσουν ελεύθερα μπλοκ μεγαλύτερου μεγέθους.

Η παρακάτω εικόνα δείχνει τον τρόπο λειτουργίας του συστήματος buddy, με ένα αρχικό μπλοκ μεγέθους 1024K όταν οι διεργασίες απαιτούν τα μεγέθη μνήμης που δείχνονται στην εικόνα.



Το ζητούμενο της άσκησης είναι να υλοποιήσετε σε Java ένα σύστημα διαχείρισης μνήμης buddy και να προσομοιώσετε τη λειτουργία του. Ως είσοδος θα σας δοθεί το άνω και κάτω όριο του μεγέθους των μπλοκ και μια ακολουθία από αιτήσεις διαχείρισης μνήμης. Μια αίτηση είναι είτε μια αίτηση για παραχώρηση μνήμης κάποιου μεγέθους σε μια διεργασία ή μια ένδειξη τερματισμού της διεργασίας (μια αίτηση για μηδενική παραχώρηση). Όλες οι αιτήσεις εξυπηρετούνται με τη σειρά που έρχονται στο σύστημα. Στο τέλος το πρόγραμμα θα πρέπει να τυπώνει την κατάσταση της μνήμης: ποιες διεργασίες έχουν δεσμεύσει ποια μπλοκ και ποια μπλοκ είναι ελεύθερα.

Μερικά στοιχεία ακόμα. Για την εξυπηρέτηση μιας αίτησης δέσμευσης μνήμης προσπαθούμε πάντα να χρησιμοποιήσουμε το μπλοκ που απελευθερώθηκε πιο πρόσφατα και μπορεί φυσικά να εξυπηρετήσει την αίτηση. Τα ελεύθερα μπλοκ εισάγονται στην κεφαλή της free list. Όταν ένα μπλοκ διαιρεθεί στα δύο, το αριστερό από αυτά (αυτό με τις μικρότερες διευθύνσεις) επιλέγεται πριν το δεξί μπλοκ. Μπορείτε επίσης να υποθέσετε ότι οι αιτήσεις είναι τέτοιες ώστε μπορούν πάντα να εξυπηρετηθούν. Με άλλα λόγια, καμία διεργασία δε θα ζητήσει κάποια στιγμή μνήμη περισσότερη από όση είναι διαθέσιμη, και καμία διεργασία δε θα τερματίσει πριν από την τελευταία αίτησή της για δέσμευση μνήμης.

Η είσοδος του προγράμματος δίνεται σε ένα αρχείο το οποίο αρχίζει με έναν θετικό ακέραιο που δηλώνει τον αριθμό των προσομοιώσεων. Κάθε ακολουθία αιτήσεων ξεκινάει με μια κενή γραμμή και ακολουθείται από μια γραμμή με δύο ακέραιους (U και L), που προσδιορίζουν τα άνω $(2^{\mathrm{U}}\mathrm{K})$ και κάτω $(2^{\mathrm{L}}\mathrm{K})$ όρια του μεγέθους των μπλοκ. Μπορείτε να υποθέσετε ότι $U > L > 0$. Στις επόμενες γραμμές υπάρχουν αιτήσεις, μία ανά γραμμή. Κάθε αίτηση έχει μια ένδειξη του ονόματος της διεργασίας (ένα string) και έναν ακέραιο, είτε θετικό (αίτηση δέσμευσης μνήμης για τη διεργασία) είτε μηδέν (τέλος διεργασίας και αποδέσμευση της μνήμης της αντίστοιχης διεργασίας).

Η έξοδος του προγράμματος αποτυπώνει την κατάσταση της μνήμης στο τέλος της εξυπηρέτησης όλων των αιτήσεων και έχει τη μορφή που φαίνεται παρακάτω. Τα μπλοκ της μνήμης τυπώνονται με αύξουσα σειρά διευθύνσεων μνήμης και δείχνουν είτε ότι είναι ελεύθερα (με χρήση μιας παύλας) ή ότι ανήκουν σε κάποια διεργασία (με το να τυπώσουν το όνομά της). Και στις δυο περιπτώσεις τυπώνεται μια άνω-κάτω τελεία και το μέγεθος του μπλοκ. Επίσης, τυπώνεται μια κενή γραμμή μεταξύ διαφορετικών προσομοιώσεων.

Αρχείο εισόδου **buddy.in**	Έξοδος προγράμματος
2	- : 128
	- : 64
10 4	D : 60
A 70	C : 80
B 35	- : 128
C 80	- : 512
A 0	
D 60	- : 128
B 0	F : 120
	G : 20
9 3	- : 32
E 80	- : 64
F 120	- : 128
G 20	
E 0	

Το φιδάκι ο Διαμαντής (0.25 βαθμοί)

Ο Διαμαντής είναι ένα μικρό φιδάκι το οποίο κινείται σε ένα τετράγωνο πάτωμα σαν και αυτό της διπλανής εικόνας. Το πάτωμα είναι χωρισμένο σε τετραγωνάκια στα οποία κάθε πρωί εμφανίζονται αριθμοί. Του Διαμαντή του αρέσουν πολύ οι βόλτες και η γνωριμία με καινούριους αριθμούς και θέλει να συναντήσει όσους το δυνατό πιο πολλούς κάθε μέρα. Όμως βαριέται όταν δε μπορεί να συναντήσει νέους αριθμούς και σταματάει αμέσως τη βόλτα του. Για να μεγιστοποιήσει τη βόλτα του, έχει τη δυνατότητα να ξεκινήσει από οποιοδήποτε τετράγωνο θέλει και να κινηθεί πάνω, κάτω, αριστερά ή δεξιά αλλά όχι διαγώνια. Γράψτε ένα πρόγραμμα στην αγαπημένη σας γλώσσα το οποίο να δέχεται ως είσοδο το μέγεθος και τους αριθμούς του πατώματος και να επιστρέφει το μήκος της μέγιστης βόλτας που ο Διαμαντής μπορεί να κάνει στο συγκεκριμένο πάτωμα. Ένα παράδειγμα εισόδου και εξόδου του προγράμματος δίνεται παρακάτω. Η μεγαλύτερη βόλτα στο πάτωμα της παραπάνω εικόνας είναι 20.

Αρχείο εισόδου **`diamantis.in`**	Έξοδος προγράμματος

`3`
`1 2 1`
`2 3 4`
`3 2 1`
	`4`

Θέματα συζήτησης (0.1 βαθμοί)

Συζητήστε σύντομα τα παρακάτω θέματα σε σχέση με τις λύσεις σας:

- Για τον Αστερίξ είναι πιο εύκολη η λύση με πίνακες στη C ή με αντικείμενα στη Java και γιατί;
- Για τους κολλητούς τι δομές δεδομένων της Java χρησιμοποιήσατε;
- Για το Διαμαντή, ποια γλώσσα χρησιμοποιήσατε και γιατί; Τι χαρακτηριστικά θα θέλατε να έχει μια γλώσσα που θα ήταν η πιο κατάλληλη για την επίλυση του συγκεκριμένου προβλήματος;

Οδηγίες για την αποστολή των λύσεων

- Μπορείτε να δουλέψετε σε ομάδες το πολύ 2 ατόμων. Αν δε θέλετε να συνεχίσετε με την ομάδα που είχατε στην πρώτη άσκηση (π.χ. να δημιουργήσατε μια νέα, να αλλάξετε ομάδα, ή να δουλέψετε μόνοι σας) φυσικά και μπορείτε να το κάνετε αλλά **θα πρέπει να το δηλώσετε μέσω mail το αργότερο μέχρι τη Δευτέρα 27 Αυγούστου.** Δε χρειάζεται κάποια τέτοια δήλωση εάν διατηρήσετε την ίδια ομάδα με αυτήν της προηγούμενης άσκησης.
- Δεν επιτρέπεται να μοιράζεστε ασκήσεις με άλλους συμφοιτητές σας ή να βάλετε τις ασκήσεις σας σε μέρος που άλλοι μπορούν να τις βρουν εύκολα
- Τα προγράμματα σε C πρέπει να είναι σε ένα αρχείο και να μπορούν να μεταγλωττιστούν με gcc 4.x.x (ή 3.x.x) με μια εντολή της μορφής: **`gcc -O3 -o file file.c`**
- Τα προγράμματα σε Java πρέπει να δουλεύουν με τον Sun Java Compiler
- Η αποστολή των ασκήσεων θα πρέπει να γίνει ηλεκτρονικά μέσω του `moodle`. Το καλύτερο είναι να στείλετε ένα `.gz` ή `.zip` αρχείο (παρακαλούμε όχι `.rar`) το οποίο να περιέχει τα 4 αρχεία με τα προγράμματα, ένα αρχείο με όνομα `AUTHORS` που να έχει τα ονόματα και τους αριθμούς μητρώων σας, ένα αρχείο `DISCUSSION` που να απαντάει σύντομα στα παραπάνω θέματα συζήτησης, και προαιρετικά ένα αρχείο `README` εάν υπάρχει κάτι το οποίο είναι άξιο ανάγνωσης που αφορά στο πως τα προγράμματα πρέπει να μεταγλωττιστούν και να τρέξουν.