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

Εαρινό 2010

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 1

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 28/4/2010, 12:30

Λείοι αριθμοί (0.25 + 0.25 = 0.5 βαθμοί)

Στη θεωρία αριθμών, *λείοι αριθμοί (smooth numbers)* είναι οι φυσικοί των οποίων οι παράγοντες είναι μικροί πρώτοι αριθμοί. Ένας φυσικός λέγεται *B-λείος (B-smooth)* εάν κανένας από τους πρώτους παράγοντές του δεν είναι μεγαλύτερος από *B*. Για παράδειγμα, ο αριθμός 1620 παραγοντοποείται ως $2^2 \times 3^4 \times 5$ και άρα είναι 5-λείος (και 42-λείος), αλλά δεν είναι 4-λείος. Η αντίστοιχη σελίδα της Wikipedia (http://en.wikipedia.org/wiki/Smooth_number) και όλοι οι σύνδεσμοι από εκεί έχουν περισσότερες πληροφορίες για τους λείους αριθμούς.

Αυτό που ζητάει η άσκηση είναι να γραφούν δύο προγράμματα (ένα σε C και ένα σε ML) τα οποία να παίρνουν ως είσοδο το **B** και δύο ακεραίους **i** και **j** (όπου **i** ≤ **j**) και να επιστρέφουν το πλήθος των **B**-λείων αριθμών μεταξύ των **i** και **j** συμπεριλαμβανομένων. Στην ιστοσελίδα του μαθήματος υπάρχει ένας σκελετός προγράμματος τον οποίο μπορείτε να χρησιμοποιήσετε για το πρόγραμμά σας σε C. Παρακάτω δείχνουμε κάποιες πιθανές κλήσεις των προγραμμάτων σε C και σε ML.

C

```
> smooth 2 1 42
6
> smooth 3 42 420
17
> smooth 7 33 192
42
```

ML

```
- smooth 2 1 42;
val it = 6 : int
- smooth 3 42 420;
val it = 17 : int
- smooth 7 33 192;
val it = 42 : int
```

Μπορείτε να υποθέσετε ότι ο **B** είναι το πολύ 1024 και οι υπόλοιποι 2 αριθμοί εισόδου είναι φυσικοί οι οποίοι μπορεί να αναπαρασταθούν σε 64 bits.

Βοήθεια, "γάιδαροι"! (0.5 βαθμοί)

Η κατάσταση με το παρκάρισμα έξω από τα νέα κτήρια τείνει να γίνει απελπιστική. Κάποιοι "γάιδαροι" παρκάρουν το αυτοκίνητό του όπου τους βολεύει αδιαφορώντας για το αν τα υπόλοιπα αυτοκίνητα μπορούν να ξεπαρκάρουν εύκολα και με ασφάλεια. Σε λίγο το να μπορέσεις να φύγεις με το αυτοκίνητό σου θα καταντήσει να γίνεται περίπου όπως στο παιχνίδι Rush Hour το οποίο περιγράφεται στην ιστοσελίδα: http://en.wikipedia.org/wiki/Rush_Hour_(board_game) και θα μας απασχολήσει σε αυτήν την άσκηση.[1]

1 Στο διαδίκτυο υπάρχουν πάρα πολλές ιστοσελίδες για το συγκεκριμένο παιχνίδι. Για παράδειγμα, δείτε τα video στο http://www.puzzles.com/products/RushHour/RushHourVideo.htm όπως και ιστοσελίδες με δύσκολους 6x6 σχηματισμούς, όμως προσέξτε ότι σε μερικές από αυτές υπάρχει διαφοροποίηση σχετικά με το τι θεωρείται ως μία κίνηση.

Ο σκοπός του παιχνιδιού είναι απλός. Το αυτοκίνητο που μας ενδιαφέρει (κόκκινο στο αυθεντικό παιχνίδι, το αυτοκίνητο με αριθμό 0 στο δικό μας) πρέπει να μπορέσει να βγει από την έξοδο. Για να μπορέσει να βγει, πρέπει τα υπόλοιπα αυτοκίνητα να μετακινηθούν. Ο χρόνος είναι διακριτός και σε κάθε χρονική στιγμή μόνο ένα αυτοκίνητο μετακινείται κάποιες θέσεις κατά τον οριζόντιο ή τον κάθετο άξονα στον οποίο βρίσκεται. Θεωρείστε την κάθε μετακίνηση ως κάτι που αποτελεί μία κίνηση του παιχνιδιού. Το ζητούμενο είναι να βρούμε τον αριθμό των ελάχιστων δυνατών κινήσεων.

Η άσκηση ζητάει να γράψτε ένα πρόγραμμα σε ML που να παίρνει ως είσοδο την πληροφορία για τον αρχικό σχηματισμό (X και Y διαστάσεις του "γκαράζ"), τις συντεταγμένες της εξόδου, και μια λίστα από που βρίσκονται τα διάφορα αυτοκίνητα) και να επιστρέφει τον ελάχιστο αριθμό κινήσεων με τις οποίες το αυτοκίνητο με το νούμερο 0 μπορεί να βρεθεί στην έξοδο.

Η χρήση του προγράμματος σε ML δείχνει ως εξής:

```
- rush_hour 6 6 (5,2) [(0,(1,2),(2,2)), (1,(0,0),(1,0)),
                       (2,(0,1),(0,3)), (3,(0,4),(0,5)),
                       (4,(3,1),(3,3)), (5,(5,0),(5,2)),
                       (6,(4,4),(5,4)), (7,(2,5),(4,5))];
val it = 8 : int
```

Εξηγούμε την είσοδο: οι δύο πρώτοι αριθμοί δίνουν τις διαστάσεις του "γκαράζ" (6x6), το επόμενο ζεύγος είναι οι συντεταγμένες του τετραγώνου στο οποίο βρίσκεται η έξοδος (η αρίθμηση αρχίζει από το 0), και τέλος η λίστα περιέχει πληροφορία για τα αυτοκίνητα, τα οποία είναι 8 συνολικά στο παράδειγμά μας (βλέπε και τον Πίνακα 1 στο παρακάτω σχήμα). Τα στοιχεία της λίστας λένε ότι το αυτοκίνητο 0 ξεκινάει από τη συντεταγμένη (1,2) και εκτείνεται μέχρι και την (2,2), το αυτοκίνητο 1 ξεκινάει από τη συντεταγμένη (0,0) και εκτείνεται μέχρι την (1,0), κοκ.

Ο μικρότερος δυνατός αριθμός κινήσεων για το παράδειγμά μας είναι 8. Στον Πίνακα 2 δείχνουμε έναν πιθανό σχηματισμό μετά την 7η από αυτές τις κινήσεις.

	0	1	2	3	4	5
0	1	1				5
1	2			4		5
2	2	0	0	4		5
3	2			4		
4	3				6	6
5	3		7	7	7	

Table 1: Ο αρχικός σχηματισμός.

exit

	0	1	2	3	4	5
0	2			1	1	
1	2					
2	2	0	0			
3	3			4		5
4	3	6	6	4		5
5	7	7	7	4		5

Table 2: Σχηματισμός μετά την 7η κίνηση. Στην επόμενη κίνηση το αυτοκίνητο 0 μπορεί να βγει.

Το πρόγραμμά σας πρέπει να επιστέφει -1 όταν δεν υπάρχει κάποιος δυνατός τρόπος να μετακινηθεί το αυτοκίνητο 0 στο σημείο εξόδου.

Μπορείτε να υποθέσετε ότι όλοι οι αριθμοί εισόδου είναι μη αρνητικοί ακέραιοι, το πολύ μέχρι 1024, και ότι όλα τα αυτοκίνητα έχουν μήκος τουλάχιστον δυο και άρα η κατεύθυνσή τους είναι μονοσήμαντα ορισμένη.

Περαιτέρω οδηγίες για την άσκηση

- Μπορείτε να δουλέψετε σε ομάδες το πολύ 2 ατόμων
- Δεν επιτρέπεται να μοιράζεστε ασκήσεις με άλλους συμφοιτητές σας ή να βάλετε τις ασκήσεις σας σε μέρος που άλλοι μπορούν να τις βρουν εύκολα (π.χ. σε κάποια σελίδα στο διαδίκτυο, σε ιστοτόπους συζητήσεων, …)
- Τα προγράμματα σε C πρέπει να είναι σε ένα αρχείο και να μπορούν να μεταγλωττιστούν με gcc με μια εντολή της μορφής: **gcc -Wall -O3 -lm -o file file.c**
- Τα προγράμματα σε ML πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε SML/NJ v110.67 (ή πιο πρόσφατη έκδοση) ή σε Mlton 20070826. Το σύστημα ηλεκτρονικής υποβολής σας επιτρέπει να επιλέξετε μεταξύ αυτών των δύο υλοποιήσεων της ML.
- Η αποστολή των προγραμμάτων θα γίνει ηλεκτρονικά μέσω του moodle. Θα υπάρξει σχετική ανακοίνωση για την ακριβή διαδικασία υποβολής. Τα προγράμματά σας πρέπει να διαβάζουν την είσοδο όπως αναφέρεται και δεν πρέπει να έχουν κάποιου άλλους είδους έξοδο διότι δε θα γίνουν δεκτά από το σύστημα στο οποίο θα υποβληθούν.