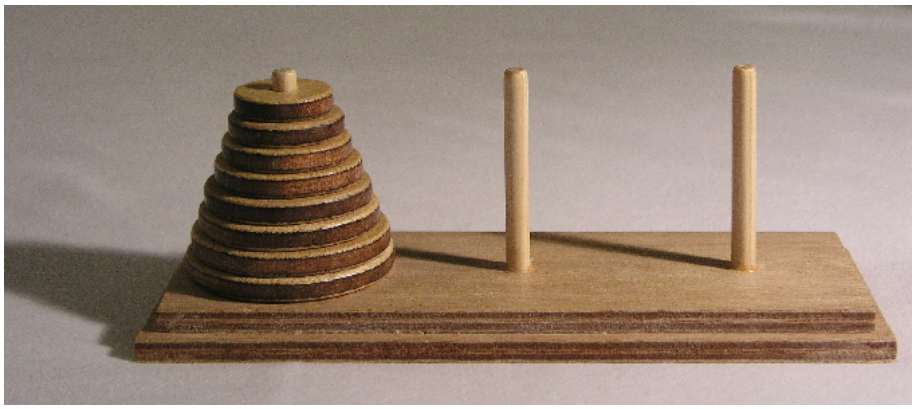
Προγραμματισμός Ι
Εργαστήριο 9

Διδάσκων: Χρήστος Δίου
Βασισμένο στο υλικό του κ. Δημήτρη Μιχαήλ

1 Οι πύργοι του Ανόι

Κάθε νέος επιστήμονας στην πληροφορική πρέπει να παλέψει με ορισμένα κλασικά προβλήματα, και οι "πύργοι του Ανόι" είναι ένα από τα πλέον δημοφιλή από αυτά. Ο μύθος λέει ότι, σε ένα ναό στην Άπω Ανατολή, οι ιερείς προσπαθούν να μετακινήσουν έναν σωρό δίσκων από ένα πείρο σε έναν άλλο. Ο αρχικός σωρός είχε 64 δίσκους τοποθετημένους από κάτω προς τα πάνω σε φθίνουσα σειρά ως προς τις διαστάσεις τους. Οι ιερείς προσπαθούν να μετακινήσουν τον σωρό από αυτόν τον πείρο σε έναν άλλο, με τον περιορισμό ότι ένας δίσκος μόνο μετακινείται κάθε φορά και ότι σε καμία περίπτωση δεν μπορεί ένας μεγαλύτερος δίσκος να τοποθετηθεί πάνω σε έναν μικρότερο. Ένας τρίτος πείρος είναι διαθέσιμος για να φέρει προσωρινά δίσκους. Υποτίθεται πως θα έρθει το τέλος του κόσμου όταν οι ιερείς τελειώσουν την δουλειά τους.

Το πρόβλημα αυτό ενώ φαίνεται αρκετά δύσκολο στην διαχείριση του, λύνεται πάρα πολύ εύκολα με έναν αναδρομικό αλγόριθμο.



Σχήμα 1: Οι πύργοι του Ανόι για 8 δίσκους.

Παρακάτω φαίνεται ένας τρόπος για να μοντελοποιήσουμε τις κινήσεις των δίσκων με μία συνάρτηση.

```c
/*
 * Move N disks from rod start to rod target using rod temp.
 * Rods are just numbers.
 */
void hanoi(int N, int start, int target, int temp) {
    /* if N is 1 move disk from start to target and return */
    /* move N-1 disks from start to temp using target */
    /* move 1 disk from start to target */
    /* move N-1 disks from temp to target using start */
}
```

Γράψτε αυτή την συνάρτηση αναδρομικά. Στην περίπτωση που το $N = 1$ φροντίστε να τυπώνει στην έξοδο

```
start -> target
```

όπου τα start και target είναι οι τιμές των αντίστοιχων μεταβλητών, και σημαίνει πως μετακινούμε ένα δίσκο από τον πείρο start στον πείρο target.

Για παράδειγμα εάν καλέσουμε την συνάρτηση ως `hanoi(4, 0, 2, 1)` πρέπει να εκτυπώσει:

```
0 -> 1
0 -> 2
1 -> 2
0 -> 1
2 -> 0
2 -> 1
0 -> 1
0 -> 2
1 -> 2
1 -> 0
2 -> 0
1 -> 2
0 -> 1
0 -> 2
1 -> 2
```

Μπορείτε να ελέγξετε την αναδρομική συνάρτηση σας με τον παρακάτω κώδικα:

```c
#include <stdio.h>
#include <stdlib.h>

void hanoi(int N, int start, int target, int temp) {
    /* this is missing */
}

int main() {
    hanoi(4, 1, 3, 2);

    return 0;
}
```

2 Ταξινόμηση Επιλογής (Selection Sort)

Ας υποθέσουμε πως έχουμε ως είσοδο ένα πίνακα $a[0..n-1]$ με ακεραίους και θέλουμε να τους βάλουμε σε σειρά από τον μικρότερο προς τον μεγαλύτερο. Η συνάρτηση που πρέπει να γράψετε σε αυτήν την άσκηση λέγεται `selectionsort()` και πρέπει να υλοποιεί τον αλγόριθμο ταξινόμησης με επιλογή που θα περιγράψουμε αμέσως μετά.

```c
#include <stdio.h>
#include <stdlib.h>

void selectionsort(int a[], int n) {
    /* missing code !! */
}

int main() {
    int a[] = { 2, 3, 4, -1, 0, 100, 99, 5, 7, 10, 1 };
    int i;

    selectionsort(a, 11);

    for(i = 0; i < 11; i++) {
```

```
15          printf("%4d", a[i]);
16      }
17      printf("\n");
18
19      return 0;
20  }
```

Ο αλγόριθμος εκτελεί n βρόχους.

- Στον 1ο βρόχο βρίσκει το ελάχιστο στοιχείο του πίνακα $a[0..n-1]$ και το αντιμεταθέτει με το 1ο στοιχείο του πίνακα $a[0]$.
- Στον 2ο βρόχο βρίσκει το ελάχιστο στοιχείο του πίνακα $a[1..n-1]$ και το αντιμεταθέτει με το 2ο στοιχείο του πίνακα $a[1]$.
- Στον i βρόχο βρίσκει το ελάχιστο στοιχείο του πίνακα $a[i-1..n-1]$ και το αντιμεταθέτει με το i στοιχείο του πίνακα $a[i-1]$.
- κ.τ.λ

Ο παραπάνω αλγόριθμος είναι ένας από τους πιο απλούς αλγόριθμους ταξινόμησης. Είναι όμως αρκετά αργός και μπορεί να χρειαστεί αρκετό χρόνο για να ταξινομήσει πολύ μεγάλες εισόδους. Για να υλοποιήσετε τον παραπάνω αλγόριθμο θα χρειαστεί να υλοποιήσετε και μια συνάρτηση που να αντιμεταθέτει δύο στοιχεία ενός πίνακα.

3 Παλινδρόμηση Αλφαριθμητικών

Παλιδρόμηση σε ένα αλφαριθμητικό υπάρχει αν το αλφαριθμητικό είναι το ίδιο είτε το δούμε από αριστερά προς τα δεξιά είτε από τα δεξιά προς τα αριστερά. Για παράδειγμα η έκφραση

"Νίψον ανομήματα μη μόναν όψιν"

περιέχει παλιδρόμηση.

Στην άσκηση αυτή πρέπει να γράψετε ένα πρόγραμμα που να διαβάζει από την τυπική είσοδο ένα κείμενο μέχρι 1000 χαρακτήρες και να ελέγχει εαν υπάρχει παλιδρόμηση. Για τον έλεγχο δεν θα πρέπει να λαμβάνετε υπ'όψη τα κενά, τα σημεία στίξης και το αν υπάρχουν πεζά ή κεφαλαία γράμματα.

Για παράδειγμα στην έκφραση

Madam, I'm Adam!

υπάρχει παλινδρόμηση γιατί αν αγνοήσουμε τα κενά και τα σημεία στίξης και θεωρήσουμε όλα τα γράμματα πεζά έχουμε την είσοδο

madamimadam

η οποία αντιστραμμένη είναι ακριβώς η ίδια. Το ίδιο ισχύει και για την φράση:

A man, a plan, a canal: Panama!

Διαβάστε την είσοδο με την χρήση της συνάρτησης `getchar()` μέχρι να δείτε `EOF`. Για να είναι πιο απλή η υλοποίηση μπορείτε να χρησιμοποιήσετε συναρτήσεις της βιβλιοθήκης της C που βρίσκονται στο αρχείο `ctype.h`. Δύο από αυτές είναι οι εξής:

1. `int isalnum(int c);`
   η συνάρτηση αυτή επιστρέφει αλήθεια εαν ο χαρακτήρας `c` είναι γράμμα ή νούμερο και ψέμα αλλιώς.
2. `int tolower(int c);`
   η συνάρτηση αυτή μετατρέπει τον γράμμα `c` σε πεζό γράμμα (εαν είναι δυνατόν).

Η έξοδος του προγράμματος σας πρέπει να είναι YES ή NO.