Λειτουργικά Συστήματα

2021 - 2022

2η Εργαστηριακή Άσκηση

Βασιλική – Ευαγγελία Δούρου, ΑΜ: 1072633

Παύλος Πεσκελίδης, ΑΜ: 1072483

Δημήτριος Μπαλάφας, ΑΜ: 1072499

Έτος 3ο, Εξάμηνο 5ο

Μέρος 1ο

Ερώτημα Α:

Σημείωση : Παρόλο που κάναμε τη χρήση της συνάρτησης clock(), όπως ανέφερε η εκφώνηση, το αποτέλεσμα σε κάθε εκτέλεση ήταν 0.00s. Οι κώδικες των δύο υποερωτημάτων βρίσκονται στο zip.

Α. Ακολουθεί εξήγηση του κώδικα υλοποίησης. Αρχικά, δημιουργούμε και αρχικοποιούμε το shared heap tree και τη shared μεταβλητή *shroot ,στην οποία θα αποθηκεύεται το ζητούμενο άθροισμα. Έπειτα, καλούμε μια for με την οποία δημιουργούμε Ν θυγατρικές διεργασίες και τα παιδιά υπολογίζουν το άθροισμα μέσω της παρακάτω for.

```
for(i=1; i<=10*N; i++)
{
*shroot=*shroot+p[i];
}
```

Εξαίρεση αποτελεί το τελευταίο παιδί, το οποίο επιπλέον κάνει print το αποτέλεσμα που είναι αποθηκευμένο στη *shroot. Ο χρόνος υπολογίζεται από τη γονική διεργασία.

Β. Αρχικά, έχουμε ορίσει μία συνάρτηση calc(int *p,int *shroot), η οποία παίρνει ως όρισμα τον πίνακα p[10*N + 1], στον οποίο είναι αποθηκευμένο το heap tree, καθώς δεν είναι shared πια και την διαμοιραζόμενη μεταβλητή *shroot, η οποία αποθηκεύει και υπολογίζει το ζητούμενο άθροισμα. Στη main() ορίζουμε με παρόμοιο τρόπο τη shared μεταβλητή *shroot και με μία for δημιουργούμε ,όπως στο προηγούμενο ερώτημα, Ν θυγατρικές διεργασίες, οι οποίες καλούν την calc(). Εξαίρεση αποτελεί το τελευταίο παιδί, το οποίο επιπλέον κάνει print το αποτέλεσμα που είναι αποθηκευμένο στη *shroot. Ο χρόνος υπολογίζεται από τη γονική διεργασία.

Ερώτημα Β:

Ο ζητούμενος κώδικας είναι στο zip. Πρωτίστως, δηλώνουμε τρεις μεταβλητές, δύο λειτουργούν ως counters (η rc και η wc) και μία μεταβλητή priority που δηλώνει τη προτεραιότητα. Παράλληλα, ορίζουμε δύο σημαφόρους, ο πρώτος θα χρησιμοποιείται για πρόσβαση στις rc,wc και ο δεύτερος θα χρησιμοποιείται για πρόσβαση στη βάση. Έπειτα, δημιουργούμε δύο συναρτήσεις , τη read_data() και τη write_data(), οι οποίες καλούνται όταν μπαίνουν στα κρίσιμα τμήματά τους οι Reader και Writer.

Στη συνέχεια, ορίζουμε μία συνάρτηση reader(), η οποία θα καλείται από τους αναγνώστες και θα ελαττώνει το σημαφόρο countsem κατά ένα, θα μεταβάλει τη τιμή του rc και τέλος θα αυξάνει τη τιμή του countsem κατά ένα. Έπειτα, θα μειώνει τη τιμή του datab κατά ένα έτσι ώστε να έχει πρόσβαση στη βάση και αν η μεταβλητή priority είναι ίση με 0, καλείται η read_data(), ελαττώνεται η τιμή του σημαφόρου countsem κατά ένα, ελαττώνεται η rc κατά ένα και αν η μεταβλητή wc είναι διαφορετική του μηδενός θέτουμε την priority ίση με ένα και τέλος αυξάνουμε κατά ένα τον countsem. Αλλιώς, αν το priority είναι ίσο με 1, ελαττώνουμε κατά ένα τον countsem και αν η wc είναι ίση με μηδέν, θέτουμε το priority ίσο με μηδέν, ελαττώνουμε την rc κατά ένα και αυξάνουμε τη τιμή του countsem κατά ένα. Τέλος, αυξάνουμε τη τιμή του σημαφόρου datab κατά ένα.

Ομοίως, ορίζουμε μία συνάρτηση writer(), η οποία λειτουργεί ακριβώς όπως η reader() με τη διαφορά, ότι αυξάνει και ελαττώνει τη τιμή του wc αντί για το rc, εκτελεί τη write_data() αντί για τη read_data(), αν το priority της είναι ίσο με ένα και θέτει το priority ίσο με μηδέν αν η rc είναι διαφορετικό του μηδενός και ίσο με ένα ,αν η rc είναι ίσο με μηδέν.
Τέλος, ορίζουμε τη main(), στην οποία αρχικοποιούνται οι σημαφόροι με ένα και οι τρεις διαμοιραζόμενες μεταβλητές μας (*rc = 1, *wc = 0, *priority = 1) και δημιουργούμε δύο διεργασίες (μία child και μία parent), οι οποίες θα καλέσουν έκαστος τη reader() και τη writer() με σκοπό να δούμε ότι ο κώδικας είναι λειτουργικός.
Παραθέτουμε ένα στιγμιότυπο από την εκτέλεση του κώδικα :

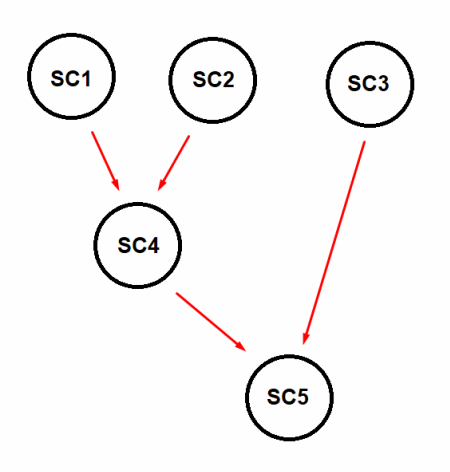
```
Written
Read
```

Ερώτημα Γ:

Ο ζητούμενος κώδικας είναι στο zip.Οι εντολές συστήματος είναι οι ακόλουθες :

SC_1: system(“pwd”), SC_2: system("ls -l"), SC_3: system("ps -l"), SC_4: system("date"), SC_5: system("ls”).

Παρακάτω, διακρίνουμε τις εξαρτήσεις σε ένα γράφο προτεραιοτήτων :



Περίπτωση 1η (Ελάχιστος αριθμός σημαφόρων)

Ορίζουμε 2 σημαφόρους για τις εξής εξαρτήσεις

- S_4 για την εξάρτηση ανάμεσα στις SC_1 και SC_4 και την εξάρτηση ανάμεσα στις SC_2 και SC_4
- S_5 για την εξάρτηση ανάμεσα στις SC_4 και SC_5 και την εξάρτηση ανάμεσα στις SC_3 και SC_5

Και οι δύο σημαφόροι αρχικοποιούνται με τη τιμή -1.

Στη main() εκτελούμε την SC_1, αυξάνουμε το σημαφόρο s_4 , εκτελούμε την SC_2, αυξάνουμε το σημαφόρο s_4, εκτελούμε την SC_3, αυξάνουμε το σημαφόρο s_5, ελαττώνουμε τον σημαφόρο s_4, εκτελούμε την SC_4, αυξάνουμε το σημαφόρο s_5, ελαττώνουμε το σημαφόρο s_5 και εκτελούμε την SC_5.

Περίπτωση 2η

Αρχικά, ορίζουμε 4 σημαφόρους για τις εξής εξαρτήσεις :

- s_1 για την εξάρτηση ανάμεσα στις SC_1 και SC_4
- s_2για την εξάρτηση ανάμεσα στις SC_2 και SC_4
- s_3 για την εξάρτηση ανάμεσα στις SC_3 και SC_5
- s_4για την εξάρτηση ανάμεσα στις SC_4 και SC_5

Έπειτα, στη main() αρχικοποιούμε τους σημαφόρους με μηδέν και εκτελούμε την SC_1, αυξάνουμε το σημαφόρο s_1 , εκτελούμε την SC_2, αυξάνουμε το σημαφόρο s_2, εκτελούμε την SC_3, αυξάνουμε το σημαφόρο s_3, ελαττώνουμε τους σημαφόρους s_1 και s_2, εκτελούμε την SC_4, αυξάνουμε το σημαφόρο s_4, ελαττώνουμε τους σημαφόρους s_3 και s_4 και εκτελούμε την SC_5.

Παραθέτουμε ένα στιγμιότυπο από την εκτέλεση του κώδικα (Προκύπτει το ίδιο στιγμιότυπο και στις 2 περιπτώσεις) :

```
/home/st1072633
total 240
-rw-r--r--.  1 st1072633 st1072633 2176 2021-12-09 17:07 1072633stored.txt
-rw-r--r--.  1 st1072633 st1072633 4233 2021-11-25 16:50 1072633.txt
drwxr-xr-x.  3 st1072633 st1072633   71 2021-11-25 15:56 Desktop
drwxr-xr-x.  2 st1072633 st1072633    6 2019-11-11 09:02 Documents
drwxr-xr-x.  2 st1072633 st1072633    6 2019-11-11 09:02 Downloads
-rw-rw-r--.  1 st1072633 st1072633  770 2020-01-13 09:44 exam1.c
-rw-rw-r--.  1 st1072633 st1072633  282 2020-01-13 10:20 exam2.c
-rw-rw-r--.  1 st1072633 st1072633  360 2020-01-13 10:35 exam3.c
-rwxrwxr-x.  1 st1072633 st1072633 6736 2021-10-30 13:14 example1
-rw-rw-r--.  1 st1072633 st1072633  188 2019-11-18 10:41 example1.c
-rwxrwxr-x.  1 st1072633 st1072633 6976 2021-12-29 22:01 example2
-rw-rw-r--.  1 st1072633 st1072633  424 2019-11-18 10:52 example2.c
-rw-rw-r--.  1 st1072633 st1072633  195 2019-11-18 10:16 example.c
-rw-rw-r--.  1 st1072633 st1072633 1896 2019-11-18 10:23 example.o
-rwxrwxr-x.  1 st1072633 st1072633 6751 2019-11-18 10:20 exe
-rwxrwxr-x.  1 st1072633 st1072633 7103 2021-12-22 20:21 ls2
-rwxrwxr-x.  1 st1072633 st1072633 6748 2021-12-21 15:44 ls2b
-rw-rw-r--.  1 st1072633 st1072633  352 2021-12-21 15:44 ls2b.c
-rwxrwxr-x.  1 st1072633 st1072633 7157 2021-12-23 21:07 ls2c
-rw-rw-r--.  1 st1072633 st1072633  420 2021-12-22 20:24 ls2.c
-rw-rw-r--.  1 st1072633 st1072633  521 2021-12-23 21:06 ls2c.c
-rw-------.  1 st1072633 st1072633    7 2021-12-21 14:39 ls2.cehelp.save
-rw-rw-r--.  1 st1072633 st1072633  432 2021-12-22 21:26 ls2d.c
-rwxrwxr-x.  1 st1072633 st1072633 6892 2021-12-23 16:57 ls2e
-rw-rw-r--.  1 st1072633 st1072633  743 2021-12-23 16:57 ls2e.c
-rwxrwxr-x.  1 st1072633 st1072633 7926 2021-12-23 21:23 ls2f
-rw-rw-r--.  1 st1072633 st1072633  992 2021-12-23 18:53 ls2f.c
drwxr-xr-x.  2 st1072633 st1072633    6 2019-11-11 09:02 Music
-rwxrwxr-x.  1 st1072633 st1072633 7758 2021-12-30 18:23 p21
-rwxrwxr-x.  1 st1072633 st1072633 7743 2021-12-30 19:14 p21b
-rw-rw-r--.  1 st1072633 st1072633  848 2021-12-30 19:02 p21b.c
-rw-rw-r--.  1 st1072633 st1072633  907 2021-12-30 18:23 p21.c
-rwxrwxr-x.  1 st1072633 st1072633 8180 2021-12-30 20:24 p22
-rw-rw-r--.  1 st1072633 st1072633 1175 2021-12-30 20:24 p22.c
-rwxrwxr-x.  1 st1072633 st1072633 7375 2021-12-31 16:25 p23
-rw-rw-r--.  1 st1072633 st1072633  627 2021-12-31 16:25 p23.c
drwxr-xr-x.  2 st1072633 st1072633    6 2019-11-11 09:02 Pictures
drwxr-xr-x.  2 st1072633 st1072633    6 2019-11-11 09:02 Public
drwx--x--x.  2 st1072633 st1072633    6 2019-10-25 18:00 public_html
drwxrwxr-x. 26 st1072633 st1072633 4096 2019-11-26 22:10 set1
drwxrwxr-x. 11 st1072633 st1072633 4096 2019-12-14 17:18 set2
drwxrwxr-x.  2 st1072633 st1072633    6 2019-11-11 10:19 set3
drwxr-xr-x.  2 st1072633 st1072633    6 2019-11-11 09:02 Templates
drwxrwxr-x.  2 st1072633 st1072633   88 2019-11-11 11:03 test2
-rw-rw-r--.  1 st1072633 st1072633   26 2019-11-11 10:35 test2.c
-rwxrwxr-x.  1 st1072633 st1072633 6435 2021-10-30 12:57 test3
-rw-rw-r--.  1 st1072633 st1072633   62 2021-10-30 12:50 test3.c
-rwxrwxr-x.  1 st1072633 st1072633 6870 2021-12-20 17:44 testls
```

```
-rwxrwxr-x.  1 st1072633 st1072633 6631 2021-12-20 18:39 testls2
-rw-rw-r--.  1 st1072633 st1072633  225 2021-12-20 18:39 testls2.c
-rw-rw-r--.  1 st1072633 st1072633  210 2021-12-20 17:44 testls.c
-rw-------.  1 st1072633 st1072633  309 2021-12-20 17:04 testls.c.save
-rw-rw-r--.  1 st1072633 st1072633 1776 2021-12-20 16:22 testls.o
-rw-rw-r--.  1 st1072633 st1072633 1791 2021-12-29 21:14 testp.c
drwxr-xr-x.  2 st1072633 st1072633    6 2019-11-11 09:02 Videos
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S 1755407418 7987 7783  0 80 0 - 30741 do_wai pts/2    00:00:00 bash
0 S 1755407418 8985 7987  0 80 0 -  2588 do_wai pts/2    00:00:00 p23
0 R 1755407418 9106 8985  3 80 0 - 27549 -      pts/2    00:00:00 ps
Πορ 31 Δεκ 2021 04:25:53 μμ EET
1072633stored.txt  Downloads  example1    example.c  ls2b    ls2c.c          ls2e.c  p21     p22    Pictures     set2       test2.c  testls2       testls.o
1072633.txt        exam1.c    example1.c  example.o  ls2b.c  ls2.cehelp.save ls2f    p21b    p22.c  Public       set3       test3    testls2.c     testp.c
Desktop            exam2.c    example2    exe        ls2c    ls2d.c          ls2f.c  p21b.c  p23    public_html  Templates  test3.c  testls.c      Videos
Documents          exam3.c    example2.c  ls2        ls2.c   ls2e            Music   p21.c   p23.c  set1         test2      testls   testls.c.save
```

Οι κώδικες και για τις δύο περιπτώσεις υπάρχουν στο zip.

Ερώτημα Δ:

Στον ψευδοκώδικα που δίνεται στην εκφώνηση, παρατηρείται ένα λάθος στην συνάρτηση **Leave_p().** Πιο συγκεκριμένα, το σφάλμα προκύπτει όταν όλες οι θέσεις είναι γεμάτες και ένας πελάτης εκτελεί την **enter_p()** για να μπει στο παρκινγκ, ενώ ταυτόχρονα ένας πελάτης που είναι ήδη στο παρκινγκ, εκτελεί τη συνάρτηση **leave_p()** για να βγει από το παρκινγκ. Σε αυτή τη περίπτωση, ο πρώτος πελάτης θα περιμένει με την await(free_s > 0); , μέχρι το free_s να είναι μεγαλύτερο του 0, δηλαδή μέχρι να ελευθερωθεί κάποια θέση, ενώ ο δεύτερος πελάτης θα αυξάνει την τιμή της free_s κατά ένα για να απελευθερώσει μία θέση στο παρκινγκ. Έτσι, το πρόβλημα προκύπτει αν πρώτα ο πρώτος πελάτης εκτελέσει το free_p:= Επιλογή_Θέσης(free_a); Και έπειτα ο δεύτερος πελάτης θέσει free_a[free_p] = TRUE; . Αυτό συμβαίνει καθώς, ενώ υπάρχει ελεύθερη θέση, δεν έχει προλάβει να ενημερωθεί ο πίνακας free_a[] για να την επιλέξει ο πελάτης 1. Το παραπάνω μπορεί να αντιμετωπισθεί, αν στη συνάρτηση **leave_p()** εκτελεστεί πρώτα το region free_a και μετά το region free_s. Παρακάτω απεικονίζεται ο ψευδοκώδικας με την επίλυση του προαναφερθέντος λάθους :

```
Leave_p(free_p)

{
region free_a do
        {
        free_a[free_p]:= TRUE;
        }

region free_s do
        {
        free_s:= free_s + 1;
        }
}
```

Ο αλγόριθμος αυτός, υλοποιημένος σε C περιλαμβάνεται στο zip file.

Μέρος 2ο

Ερώτημα Α:

Ο ζητούμενος πίνακας συμπληρωμένος είναι ο ακόλουθος :

Χρονική Στιγμή	Άφιξη	Εικόνα Μνήμης	ΚΜΕ	Ουρά μνήμης	Ουρά ΚΜΕ	Τέλος
0	P1	<Οπή 620K>	-	P1	-	-
1	P2	<P1-180K> <Οπή 440K>	P1	P2	-	-
2	P3	<P1-180K> <P2-100K> <Οπή 340K>	P2	P3	P1	-
3	P4,P5	<P1-180K> <P2-100K> <Οπή 340K>	P2	P3,P4,P5	P1	P2
4	P6	<P1-180K> <P4-100K> <P5-90K> <Οπή 250K>	P5	P3,P6	P1,P4	-
5	-	<P1-180K> <P4-100K> <P5-90K> <P6 80K> <Οπή 170K>	P6	P3	P1,P4,P5	-
6	-	<P1-180K> <P4-100K> <P5-90K> <P6 80K> <Οπή 170K>	P6	P3	P1,P4,P5	P6
7	-	<P1-180K> <P4-100K> <P5-90K> <Οπή 250K>	P5	P3	P1,P4	-
8	-	<P1-180K> <P4-100K> <P5-90K> <Οπή 250K>	P5	P3	P1,P4	-
9	-	<P1-180K> <P4-100K> <P5-90K> <Οπή 250K>	P5	P3	P1,P4	P5
10	-	<P1-180K> <P4-100K> <Οπή 340K>	P1	P3	P4	-
11	-	<P1-180K> <P4-100K> <Οπή 340K>	P1	P3	P4	-
12	-	<P1-180K> <P4-100K> <Οπή 340K>	P1	P3	P4	-
13	-	<P1-180K> <P4-100K> <Οπή 340K>	P1	P3	P4	-
14	-	<P1-180K> <P4-100K> <Οπή 340K>	P1	P3	P4	-
15	-	<P1-180K> <P4-100K> <Οπή 340K>	P1	P3	P4	P1
16	-	<Οπή 180K> <P4-100K> <Οπή 340K>	P4	P3	-	-
17	-	<Οπή 180K> <P4-100K> <Οπή 340K>	P4	P3	-	-
18	-	<Οπή 180K> <P4-100K> <Οπή 340K>	P4	P3	-	-
19	-	<Οπή 180K> <P4-100K> <Οπή 340K>	P4	P3	-	-
20	-	<Οπή 180K> <P4-100K> <Οπή 340K>	P4	P3	-	-
21	-	<Οπή 180K> <P4-100K> <Οπή 340K>	P4	P3	-	P4
22	-	<P3-350K> <Οπή 270K>	P3	-	-	-
23	-	<P3-350K> <Οπή 270K>	P3	-	-	-
24	-	<P3-350K> <Οπή 270K>	P3	-	-	-
25	-	<P3-350K> <Οπή 270K>	P3	-	-	-
26	-	<P3-350K> <Οπή 270K>	P3	-	-	P3

Ερώτημα Β:

Α) Η εσωτερική κλασματοποίηση ορίζεται ως εξής :

Μια διαδικασία που χρειάζεται m θέσεις μνήμης μπορεί να τρέξει σε μία περιοχή n bytes, όπου n>=m. Η διαφορά των δύο αυτών αριθμών (n-m) είναι η εσωτερική κλασματοποίηση, δηλαδή η μνήμη που εμπεριέχεται σε μία περιοχή και δε χρησιμοποιείται.

Οι λογικές διευθύνσεις είναι των 20 bits. Κάθε λογική διεύθυνση χωρίζεται στον αριθμό σελίδας και στη διεύθυνση μέσα στη σελίδα. Το μέγεθος κάθε σελίδας / πλαισίου είναι 1KB ή 2^{10} bytes, οπότε χρειάζεται 10 bits. Άρα για τον αριθμό σελίδας απομένουν 20-10 = 10bits. Η φυσική μνήμη είναι 4MB ή 2^{22} bytes, οπότε χρειάζονται 22 bits για τη προσπέλασή της. Το πλαίσιο έχει το ίδιο μέγεθος με τη σελίδα, δηλαδή απαιτούνται 10 bits.

Φυσική διεύθυνση

12 bits	10 bits

Λογική διεύθυνση

10 bits	10 bits

Η διεργασία αποτελείται από 6100 bytes ή 5,957 KB. Επομένως, είναι κατανοητό πως η διεργασία θα απασχολεί 6 πλαίσια των 1 KB. Ωστόσο, από τη στιγμή που η διεργασία δεν θα απασχολεί 6 KB ακριβώς θα υπάρχουν 6 KB – 5,957 KB = 0,043 KB υπολειπόμενα στο $6^{ο}$ πλαίσιο. Συνολικά η εσωτερική κλασματοποίηση θα είναι 0,043 KB.

Β)

Ι. Λογική διεύθυνση

0000	0000	0000	0011	1000	1000

Φυσική διεύθυνση

0000	0000	0010	1111	1000	1000
0	0	2	F	8	8

Άρα η φυσική διεύθυνση $002F88_{16}$

ΙΙ. Λογική διεύθυνση

0000	0000	0001	0010	0101	1111

Φυσική διεύθυνση

-	-	-	-	-	-
-	-	-	-	-	-

Η φυσική διεύθυνση δε μπορεί να βρεθεί γιατί η λογική σελίδα δε βρίσκεται στη φυσική μνήμη.

ΙΙΙ. Λογική διεύθυνση

0000	0001	0101	1010	1010	0100

Φυσική διεύθυνση

0000	0000	0010	0001	1010	0100
0	0	2	1	A	4

Άρα η φυσική διεύθυνση $0021A4_{16}$

Ερώτημα Γ:

Ο πίνακας της εκφώνησης παρατίθεται συμπληρωμένος :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	2	5	8	1	8	7	5	1	8	2	4	2	1	3	6	4	7	5	3	7

0	2	2	2	2	2	7	7	7	7	2	2	2	2	2	2	4	4	4	4	4
1		5	5	5	5	5	5	5	5	5	4	4	4	4	6	6	6	6	3	3
2			8	8	8	8	8	8	8	8	8	8	8	3	3	3	3	5	5	5
3				1	1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7
Σφάλμα	X	X	X	X		X				X	X			X	X	X	X	X	X	

Σύμφωνα με τον αλγόριθμο LRU αρχικά θα τοποθετήσουμε την ακολουθία στα πλαίσια μέχρι να συμπληρωθούν και τα 4 πλαίσια. Ύστερα, εφόσον ο επόμενος αριθμός υπάρχει ήδη στα πλαίσια, δεν θα υπάρχει σφάλμα σελίδας και θα τοποθετηθούν οι αριθμοί όπως και προηγουμένως. Στη συνέχεια εμφανίζεται ο αριθμός 7 ο οποίος θα πρέπει να αντικαταστήσει έναν αριθμό. Το 7, σύμφωνα με την εκτέλεση του LRU θα μπει στη θέση του αριθμού ο οποίος κλήθηκε τελευταίος στην ακολουθία, στη συγκεκριμένη περίπτωση του 2. Συνεχίζοντας, εκτελούμε τον αλγόριθμο με ανάλογο τρόπο ως το τέλος της ακολουθίας.