ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Τμήμα Εφαρμοσμένης Πληροφορικής

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Εξάμηνο Β'

Φύλλο Ασκήσεων 4: ΣΥΝΔΕΤΙΚΗ ΛΙΣΤΑ, ΣΤΟΙΒΑ – ΟΥΡΑ (υλοποίηση με δείκτες)

Μάγια Σατρατζέμη, Γεωργία Κολωνιάρη, Αλέξανδρος Καρακασίδης

Παρατηρήσεις για τις ασκήσεις με Απλή Συνδετική Λίστα (ΑΣΛ):

1. *Τα δεδομένα εισόδου διαβάζονται πάντα με ξεχωριστές εντολές scanf( ) το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
2. *Αντίστοιχα για τα δεδομένα εξόδου και όπου δεν υπάρχουν περαιτέρω διευκρινήσεις για τη μορφή τους, αυτά θα εμφανίζονται με ξεχωριστές εντολές printf( ) το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
3. *Τα στοιχεία των κόμβων της Α.Σ.Λ. θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους. Σε περίπτωση που οι κόμβοι της Α.Σ.Λ. περιέχουν περισσότερα από ένα στοιχεία, τότε τα στοιχεία κάθε κόμβου θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους, ενώ κάθε κόμβος θα εμφανίζεται σε διαφορετική γραμμή.*
    - *Αν κατά τη διάσχιση της λίστας διαπιστώσετε ότι η Α.Σ.Λ. είναι κενή, τότε να εμφανίζετε το μήνυμα 'EMPTY LIST'.*
4. *Σ' όλα τα προγράμματα η ΑΣΛ ή οι ΑΣΛ θεωρούνται δεδομένες, βέβαια θα πρέπει να την/τις κατασκευάσετε διαβάζοντας τα δεδομένα ως εξής : πλήθος στοιχείων λίστας, στοιχεία λίστας ή αν πρόκειται για 2 λίστες τότε θα τα διαβάζετε ως εξής: πλήθος στοιχείων λίστας1, στοιχεία λίστας1, πλήθος στοιχείων λίστας2, στοιχεία λίστας*
5. ***ΠΡΟΣΟΧΗ: Οι ασκήσεις θα πρέπει να λύνονται με χρήση του κώδικα που υλοποιεί τον ΑΤΔ ΑΣΛ. Ο κώδικας που σας δίνεται περιλαμβάνεται στο code.zip στην αντίστοιχη διάλεξη. Οι συναρτήσεις που υλοποιούν τις βασικές λειτουργίες του ΑΤΔ ΑΣΛ δεν τροποποιούνται. Τροποποιήσεις μπορούν να γίνουν ανάλογα με τη άσκηση και εφόσον χρειάζεται μόνο στο πλήθος των στοιχείων της ΑΣΛ και στον τύπο του στοιχείου της ΑΣΛ.***

---

1. Να υλοποιηθεί η συνάρτηση Search με το παρακάτω πρωτότυπο:

```
boolean Search(ListPointer FreePtr, ListPointer List, NodeType Node[NumberOfNodes], ListElementType Item,
ListPointer *PredPtr);
```

η οποία θα δέχεται μια Συνδεδεμένη Λίστα (ΣΛ) υλοποιημένη με πίνακα (παράμετροι FreePtr, List, Node) και ένα στοιχείο Item και θα επιστρέφει boolean τιμή η οποία δείχνει αν βρέθηκε ή όχι το αναζητούμενο στοιχείο Item στη ΣΛ και τη θέση (PredPtr) του στοιχείου που είναι προηγούμενο του Item, ώστε η ΣΛ που θα προκύψει μετά την εισαγωγή ή τη διαγραφή του στοιχείου Item να είναι ταξινομημένη (χρήση της Search πριν την εισαγωγή ή τη διαγραφή στοιχείου στη ΣΛ). Η εισαγωγή στοιχείων στην Σ.Λ. γίνεται επαναληπτικά και ελέγχεται από το μήνυμα 'Continue Y/N:' (δες στιγμιότυπο εκτέλεσης). Στη ΣΛ καταχωρούνται ακέραιοι. Η συνάρτηση Search θα κληθεί 2 φορές μια για στοιχείο που υπάρχει στη ΣΛ και μια για στοιχείο που δεν υπάρχει στη ΣΛ. Αν το στοιχείο βρεθεί θα εμφανίζει το μήνυμα «The number is in the list and its predecessor is in position» και την τιμή της PredPtr, διαφορετικά θα εμφανίζει το μήνυμα «The number is not in the list». Το πρόγραμμα θα πρέπει να εκτελεί κατά σειρά τις παρακάτω λειτουργίες και οι κλήσεις των συναρτήσεων θα γίνονται από τη main().

A. Αρχικοποίηση storage pool
B. Δημιουργία ΑΣΛ (μέγιστο μέγεθος 10)
C. Εμφάνιση της storage pool
D. Εμφάνιση των στοιχείων της ΣΛ
E. Εισαγωγή στοιχείων στην ΣΛ. Μετά από κάθε εισαγωγή στοιχείου στη ΣΛ αυτή θα πρέπει να παραμένει ταξινομημένη σε αύξουσα διάταξη (χρήση της Search πριν την εισαγωγή στοιχείου στη ΣΛ)
F. Εμφάνιση της storage pool
G. Εμφάνιση των στοιχείων της ΣΛ
H. Έλεγχος εάν η ΣΛ είναι άδεια. Αν η ΣΛ είναι άδεια εμφανίζει μήνυμα «Empty List» διαφορετικά «Not an Empty List»
I. Έλεγχος εάν η ΣΛ είναι γεμάτη. Αν η ΣΛ είναι γεμάτη εμφανίζει μήνυμα «Full List» διαφορετικά «Not a Full List»
J. Αναζήτηση ενός στοιχείου στη ΣΛ (για στοιχείο που υπάρχει στη ΣΛ και για στοιχείο που δεν υπάρχει).

```
---------------Question C--------------------
--------------Storage pool-------------------
1o STOIXEIO LISTAS=-1, 1H FREE POSITION=0
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0: -1->1) (1: -1->2) (2: -1->3) (3: -1->4) (4: -1->5) (5: -1->6) (6: -1->7) (7: -1->8) (8: -
```

```
1->9) (9: -1->-1)
---------------Question D--------------------
--------------Linked list--------------------
Empty List ...
---------------Question E--------------------
Give a number: 20

Continue Y/N: y
Give a number: 3

Continue Y/N: y
Give a number: 10

Continue Y/N: y
Give a number: 12

Continue Y/N: y
Give a number: 4

Continue Y/N: n
---------------Question F--------------------
--------------Storage pool------------------
1o STOIXEIO LISTAS=1, 1H FREE POSITION=5
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0: 20->-1) (1: 3->4) (2: 10->3) (3: 12->0) (4: 4->2) (5: -1->6) (6: -1->7) (7: -1->8) (8: -1-
>9) (9: -1->-1)
---------------Question G--------------------
--------------Linked list--------------------
(1: 3 ->4) (4: 4 ->2) (2: 10 ->3) (3: 12 ->0) (0: 20 ->-1)
---------------Question H--------------------
Not an Empty List
---------------Question I--------------------
Not a Full List
---------------Question J--------------------
-----------------Search for a number-----------------
Give a number 4
The number is in the list and its predecessor is in position 1
Give a number 23
The number is not in the list
```

2. Για τις παρακάτω λειτουργίες που αφορούν σε απλή συνδετική λίστα (Α.Σ.Λ.) **με δείκτες** να γραφούν διαφορετικά προγράμματα για καθεμιά από αυτές. Γνωρίζουμε τα εξής:

- Κάθε κόμβος της ΑΣΛ ή των ΑΣΛ περιέχει έναν ακέραιο αριθμό.
- Η είσοδος των δεδομένων θα γίνεται ως εξής: Πρώτα θα διαβάζεται για κάθε ΑΣΛ το πλήθος των κόμβων της Α.Σ.Λ. και στη συνέχεια τα στοιχεία της.
- **Σε όλες τις παρακάτω ασκήσεις η εισαγωγή στοιχείου, για την κατασκευή της υπάρχουσας λίστας γίνεται στην αρχή της λίστας**.
- Σε κάθε πρόγραμμα θα εμφανίζονται οι κόμβοι της αρχικής ή των αρχικών λιστών και στη συνέχεια αυτών που δημιουργούνται κατά την εκτέλεση του προγράμματος.
- Η εμφάνιση των δεδομένων θα γίνεται ως εξής: Τα στοιχεία των κόμβων της Α.Σ.Λ. θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους.
- Αν κατά τη διάσχιση της λίστας διαπιστώσετε ότι η Α.Σ.Λ. είναι κενή, τότε να εμφανίζετε το μήνυμα '*EMPTY LIST*'.

Σ' όλα τα προγράμματα θα πρέπει να κατασκευάσετε την ΑΣΛ ή τις ΑΣΛς διαβάζοντας τα δεδομένα ως εξής: πλήθος στοιχείων λίστας, στοιχεία λίστας ή αν πρόκειται για 2 λίστες τότε θα τα διαβάζετε ως εξής: πλήθος στοιχείων λίστας1, στοιχεία λίστας1, πλήθος στοιχείων λίστας2, στοιχεία λίστας2

Η καθεμία από τις παρακάτω λειτουργίες να υλοποιηθεί με τη χρήση συνάρτησης.

(a) Προσθήκη ενός στοιχείου στο τέλος μιας υπάρχουσας Α.Σ.Λ.

(b) Δημιουργία αντιγράφου Α.Σ.Λ

(c) Συνένωση 2 Α.Σ.Λ. σε μία (πχ 1^{η} ΑΣΛ: 4, 3, 2, 1 και 2^{η} ΑΣΛ: 8, 7, 6, τελική ΑΣΛ: 6, 7, 8, 1, 2, 3, 4). Δίνεται το πρωτότυπο της συνάρτησης: `void concat_list(ListPointer List, ListPointer BList, ListPointer *FinalList)`

Στη συνέχεια δίνεται ένα στιγμιότυπο εκτέλεσης: (τα στοιχεία των λιστών δεν ακολουθούν κάποια διάταξη)

```
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS 1: 4
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 1
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 3
```

```
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 4
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS 2: 3
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 6
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 7
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 8
LISTA 1:
4 3 2 1
LISTA 2:
8 7 6
SYNENWMENH LISTA:
6 7 8 1 2 3 4
```

(d) Διαγραφή όλων των κόμβων της Α.Σ.Λ.

(e) Αντιστροφή των κόμβων της Α.Σ.Λ. (το τελευταίο στοιχείο να γίνει πρώτο κλπ).

(f) Διαγραφή του τελευταίου στοιχείου της Α.Σ.Λ.

(g) Διαγραφή του ν-οστού στοιχείου της Α.Σ.Λ. (να γίνεται έλεγχος ότι το ν είναι θετικός αριθμός και δεν είναι μεγαλύτερος από το πλήθος των στοιχείων της λίστας). Πρώτα διαβάζεται η λίστα (πλήθος στοιχείων λίστας, στοιχεία λίστας) και στη συνέχεια ο αριθμός ν.

Παράδειγμα εκτέλεσης:

```
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS: 7
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 4
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 1
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 8
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 4
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 6
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 3
DWSE TI THESI GIA DIAGRAFI TOY STOIXEIOU: 3
Arxiki lista
3 6 2 4 8 1 4
Teliki lista
3 6 4 8 1 4
```

(h) Διαγραφή κάθε 2ου στοιχείου της Α.Σ.Λ.

(i) Δημιουργία μιας Α.Σ.Λ. που περιέχει την τομή των στοιχείων 2 άλλων Α.Σ.Λ. (οι Α.Σ.Λ. δεν είναι ταξινομημένες). Κάθε στοιχείο εμφανίζεται στην Α.Σ.Λ. της τομής μία φορά ανεξάρτητα από το πλήθος εμφανίσεων του στις αρχικές Α.Σ.Λ..

Παράδειγμα εκτέλεσης:

```
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS 1: 4
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 7
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 2
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS 2: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 1
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 4
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 2: 2
1h lista
2 7 2 5
2h lista
2 4 5 5 1
Teliki lista
5 2
```

(j) Να γραφεί συνάρτηση για την εισαγωγή *m* στοιχείων μετά το n-οστό στοιχείο της Α.Σ.Λ.

Το πρωτότυπο της συνάρτησης είναι `void insert_list_m_elements(ListPointer *List, int n);`

Το διάβασμα των δεδομένων θα γίνεται ως εξής: η Σ.Λ. (πλήθος στοιχείων λίστας, στοιχεία λίστας), στη συνέχεια ο αριθμός n, το πλήθος m και τέλος τα m στοιχεία. Το πλήθος m και τα m στοιχεία θα διαβάζονται μέσα στη `insert_list_m_elements`(). Ο έλεγχος της τιμής της n θα γίνεται μέσα στη συνάρτηση `insert_list_m_elements`() και αν n > πλήθος των κόμβων της Σ.Λ., τότε θα εμφανίζεται το μήνυμα *'ERROR'*.

Στη συνέχεια δίνεται στιγμιότυπο εκτέλεσης:

```
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS: 6
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 8
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 3
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 7
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 2
*********Arxiki lista***********
2 7 2 5 3 8
DWSE TI THESI, META APO TIN OPOIA THA EISAXTHOUN TA
STOIXEIA: 9
```

```
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS: 6
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 8
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 3
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 7
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 2
*********Arxiki lista***********
2 7 2 5 3 8
DWSE TI THESI, META APO TIN OPOIA THA EISAXTHOUN TA
STOIXEIA: 3
```

```
ERROR
*********Teliki lista************
2 7 2 5 3 8
```

```
DWSE TO PLITHOS TWN STOIXEIWN POU THA EISAXTHOUN: 2
DWSE TON ARI8MO GIA EISAGWGH STH LISTA: 12
DWSE TON ARI8MO GIA EISAGWGH STH LISTA: 11
*********Teliki lista************
   2   7 2 12 11 5 3 8
```

(k) Εισαγωγή στοιχείου σε ταξινομημένη Α.Σ.Λ. Τα στοιχεία της υπάρχουσας Α.Σ.Λ. θα είναι ήδη ταξινομημένα κατά αύξουσα σειρά.

(l) Συγχώνευση (merge) 2 ταξινομημένων Α.Σ.Λ. σε μία. Τα στοιχεία των 2 αρχικών λιστών ταξινομούνται κατά το διάβασμά τους.

(m) Επιστροφή του αθροίσματος των στοιχείων μιας Α.Σ.Λ. Η εισαγωγή στοιχείων στην Α.Σ.Λ. θα ελέγχεται από το μήνυμα '*New insertion Y/N (Y=yes, N=No)*'.

(n) Επιστροφή του πλήθους των στοιχείων μιας Α.Σ.Λ. Η εισαγωγή στοιχείων στην Α.Σ.Λ. ελέγχεται από το μήνυμα '*New insertion Y/N (Y=yes, N=No)*'.

(o) Μετακίνηση του κόμβου με το στοιχείο *p* κατά *n* θέσεις προς τα εμπρός (προς την αρχή της Α.Σ.Λ.). Σε περίπτωση που ο κόμβος με στοιχείο *p* δεν υπάρχει, τότε θα εμφανίζεται το μήνυμα '*DATA NOT IN LIST*' και αν *n* > πλήθος των κόμβων της Α.Σ.Λ., τότε θα εμφανίζεται το μήνυμα '*WRONG INDEX*'. Αφού διαβαστεί η Α.Σ.Λ. (πλήθος στοιχείων λίστας, στοιχεία λίστας), στη συνέχεια διαβάζονται οι αριθμοί *p* και *n*.

(p) Διαχωρισμός λίστας. υλοποιήστε την παρακάτω συνάρτηση:

```
void Larger(ListPointer InList, ListElementType number, ListPointer *OutList);
```

όπου InList μια λίστα εισόδου και number ένας αριθμός. Η συνάρτηση αυτή δημιουργεί μια νέα λίστα (OutList) η οποία περιέχει τα στοιχεία εκείνα της InList τα οποία είναι μεγαλύτερα ή ίσα με τον αριθμό number. (*Προσοχή στην* OutList *τα στοιχεία εισάγονται στο τέλος της λίστας*)
Παράδειγμα εκτέλεσης:
INPUT LIST: [100 34 45 12 23 56 78 89 23]
LARGER LIST (THRESHOLD 56): [100 56 78 89]

(q) Διαγραφή γειτονικών κόμβων. Υλοποιήστε την παρακάτω συνάρτηση:

```
void UniqueList(ListPointer InList, ListPointer *OutList);
```

η οποία δέχεται ως είσοδο μια λίστα (InList) και επιστρέφει μια νέα λίστα (OutList) στην οποία έχουν διαγραφεί οι γειτονικοί κόμβοι που περιέχουν ίδιες τιμές. Η αρχική λίστα παραμένει αμετάβλητη. (*Προσοχή στην OutList τα στοιχεία εισάγονται στο τέλος της λίστας*)
Παράδειγμα εκτέλεσης:

```
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS: 10
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 1
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 1
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 1
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 3
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 1
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 3
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 3
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS: 5
Arxiki lista
5 5 3 3 1 3 2 1 1 1
Teliki lista
5 3 1 3 2 1
```

(r) Αφαίρεση ελάχιστων. Υλοποιήστε την παρακάτω συνάρτηση: `ListElementType RemoveMins(ListPointer *InList);` η οποία δέχεται ως είσοδο μια λίστα (InList) και θα:

- Βρίσκει τον ελάχιστο στοιχείο της λίστας και
- αφαιρεί από τη λίστα όλους τους αριθμούς ίσους με τον ελάχιστο αριθμό και τον επιστρέφει. Η εμφάνιση του ελάχιστου θα γίνεται στη main()

Στη συνέχεια δίνεται ένα στιγμιότυπο εκτέλεσης

```
DWSE TON PLH8OS TWN STOIXEIWN THS LISTAS : 6
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 8
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 3
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 5
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 2
DWSE TON ARI8MO GIA EISAGWGH STH ARXH THS LISTAS 1: 7
Traversing list
7 2 5 2 3 8
Removing min from list
Min was 2
```

```
Traversing list with min removed
7 5 3 8
```

(s) Αφαίρεση μεγαλύτερων. Υλοποιήστε την παρακάτω συνάρτηση:

void RemoveMaxs(ListPointer InList, ListPointer *OutList);

η οποία δέχεται ως είσοδο μια λίστα (InList) που περιέχει αριθμούς, και επιστρέφει μια νέα λίστα (OutList) με όλα τα στοιχεία της αρχικής στην ίδια διάταξη εκτός του μεγίστου αριθμού (και όλων των ίσων με το μέγιστο) τον οποίο/α και αφαιρεί. Η αρχική λίστα παραμένει αμετάβλητη. (*Προσοχή στην OutList τα στοιχεία εισάγονται στο τέλος της λίστας*)

3. Γράψτε ένα πρόγραμμα που να διασχίζει μια αρχική λίστα από ακεραίους (λίστα Α) και απ' αυτή θα δημιουργεί δυο νέες λίστες, η μια που θα περιέχει τους περιττούς ακεραίους (λίστα ΠΕ) και η άλλη τους άρτιους (λίστα ΑΡ). Τα στοιχεία της λίστας Α θα τα διαβάζετε από το αρχείο κειμένου 'I3F4.DAT', ένα στοιχείο κάθε φορά μέχρι να συναντήσετε το τέλος του αρχείου, και **κάθε στοιχείο θα το προσθέτετε στην αρχή της λίστας Α**. Στην συνέχεια θα διασχίζετε τη λίστα Α και ανάλογα αν ο κόμβος περιέχει άρτιο αριθμό αυτός θα προστίθεται **στην αρχή της λίστας ΑΡ**, αλλιώς **στην αρχή της λίστας ΠΕ**. Όταν ολοκληρωθεί η διάσχιση της λίστας Α και άρα η δημιουργία της λίστας ΑΡ και της λίστας ΠΕ στη συνέχεια θα εμφανίζετε τα στοιχεία της αρχικής λίστας και των δύο νέων λιστών. Τα στοιχεία κάθε λίστας να εμφανίζονται σε διαφορετική γραμμή και μεταξύ τους θα διαχωρίζονται με ένα κενό χαρακτήρα..

4. Γράψτε πρόγραμμα που θα δημιουργεί μια λίστα πελατών μιας τράπεζας που κάθε κόμβος της περιέχει το ονοματεπώνυμο και το υπόλοιπο του λογαριασμού του πελάτη. Τα στοιχεία της λίστας αυτής θα τα διαβάζετε από το αρχείο κειμένου 'I4F4.DAT'. Στη συνέχεια το πρόγραμμα θα διαβάζει αυτή τη λίστα των πελατών και θα δημιουργεί "μια μαύρη λίστα" με τους πελάτες που έχουν αρνητικό υπόλοιπο. Στη συνέχεια θα εμφανίζετε τους κόμβους της αρχικής λίστας και της μαύρης λίστας. Τα στοιχεία κάθε λίστας να εμφανίζονται σε διαφορετική γραμμή και μεταξύ τους θα διαχωρίζονται με ένα κενό χαρακτήρα. **Η εισαγωγή των στοιχείων θα γίνεται στην αρχή της αντίστοιχης λίστας**.

5. Γράψτε πρόγραμμα που διαβάζει ένα *string* και αντιστρέφει έναν-έναν τους χαρακτήρες προσθέτοντάς τους στην **αρχή της συνδετικής λίστας**. Στη συνέχεια να γίνεται διάσχιση της λίστας και να εμφανίζονται οι αντεστραμμένοι πλέον χαρακτήρες της συμβολοσειράς χωρίς κενά μεταξύ τους.

6. Μια λίστα ακεραίων αριθμών είναι ταξινομημένη σε αύξουσα σειρά μέχρι κάποιο σημείο της (ημιταξινομημένη). Να βρεθεί η θέση μέχρι την οποία υπάρχει ταξινόμηση και τα υπόλοιπα στοιχεία να τοποθετηθούν στην κατάλληλη θέση ώστε η λίστα που θα προκύψει να είναι ταξινομημένη. Κατά την εκτέλεση του προγράμματος να εμφανίζονται τα στοιχεία των κόμβων της αρχικής και της τελικής ταξινομημένης λίστας (σε αύξουσα διάταξη) σε ξεχωριστή γραμμή για κάθε λίστα. Τα στοιχεία στην αρχική λίστα εισάγονται κάθε φορά στην αρχή της λίστας.

7. Να γίνει πρόγραμμα που θα μετατρέπει έναν αριθμό από το δεκαδικό σύστημα στο δυαδικό με τη βοήθεια ΑΣΛ και θα εμφανίζει τον αριθμό στο δυαδικό σύστημα.

Παρατηρήσεις για τις ασκήσεις με στοίβα και ουρά:

1. *Τα δεδομένα εισόδου διαβάζονται πάντα με ξεχωριστές εντολές scanf( ) το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
2. *Αντίστοιχα για τα δεδομένα εξόδου και όπου δεν υπάρχουν περαιτέρω διευκρινήσεις για τη μορφή τους, αυτά θα εμφανίζονται με ξεχωριστές εντολές printf( ) το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
   i) ***Τα στοιχεία των κόμβων της στοίβας/ουράς θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους.*** *Σε περίπτωση που οι κόμβοι της στοίβας/ουράς περιέχουν περισσότερα από ένα στοιχεία, τότε τα στοιχεία κάθε κόμβου θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους, ενώ κάθε κόμβος θα εμφανίζεται σε διαφορετική γραμμή.*
   ii) Αν κατά τη διάσχιση της λίστας διαπιστώσετε ότι η στοίβα/ουρά είναι κενή, τότε να εμφανίζετε αντίστοιχα το μήνυμα 'EMPTY STACK' ή 'EMPTY QUEUE'.
3. *Σε όσες από τις ασκήσεις θεωρείται δεδομένη η ύπαρξη στοίβας/ουράς θα πρέπει προηγουμένως να τη δημιουργήσετε.*
4. ***ΠΡΟΣΟΧΗ: Οι ασκήσεις θα πρέπει να λύνονται με χρήση του κώδικα που υλοποιεί τον ΑΤΔ στοίβα/ουρά (με δείκτες). Ο κώδικας που σας δίνεται περιλαμβάνεται στο code.zip στην αντίστοιχη διάλεξη. Οι συναρτήσεις που υλοποιούν τις βασικές λειτουργίες του ΑΤΔ στοίβα/ουρά (με δείκτες) δεν τροποποιούνται. Τροποποιήσεις μπορούν να γίνουν ανάλογα με τη άσκηση και εφόσον χρειάζεται στον τύπο του στοιχείου της στοίβα/ουρά (με δείκτες).***

---

8. Δίνονται *N* αριθμοί X_j, *j = 1, 2, …,N*. Να αναπτυχθεί πρόγραμμα που θα διαβάζει το πλήθος *N* των αριθμών και τους *N* αριθμούς και στη συνέχεια θα βρίσκει και θα εμφανίζει το μεγαλύτερο αριθμό και τις θέσεις στις οποίες εμφανίζεται αυτός σε περίπτωση που εμφανίζεται περισσότερες από 1 φορές. Το παραπάνω πρόβλημα να λυθεί με τη χρήση στοίβας με δείκτες όπου θα αποθηκεύονται οι διαφορετικές θέσεις εμφάνισης του μεγαλύτερου αριθμού.

9. Να γραφεί πρόγραμμα που να διαβάζει ένα αλφαριθμητικό και στη συνέχεια να προσθέτει έναν-ένα τους χαρακτήρες του σε μια στοίβα και σε μια ουρά ταυτόχρονα. Μετά το τέλος της εισαγωγής των χαρακτήρων του αλφαριθμητικού στη στοίβα και στην ουρά θα εμφανίζετε τα στοιχεία της στοίβας και της ουράς. Η εμφάνιση των στοιχείων της στοίβας και ουράς θα γίνεται όπως περιγράφεται στις Παρατηρήσεις στο σημείο 2i. Στη συνέχεια θα ελέγχετε αν το αλφαριθμητικό είναι καρκινικό, οπότε κι θα εμφανίζετε το μήνυμα '*ACCEPTED'*, ή όχι, οπότε θα εμφανίζετε το μήνυμα 'REJECTED' (χρησιμοποιήστε τις διαδικασίες: εισαγωγή στοιχείου σε στοίβα, εισαγωγή στοιχείου σε ουρά, εξαγωγή στοιχείου από στοίβα, εξαγωγή στοιχείου από ουρά με δυναμική υλοποίηση).
Στη συνέχεια δύνονται 2 στιγμιότυπα εκτέλεσης (με καρκινικό και μη καρκινικό αλφαριθμητικό).

DWSE TO ALFARITHMITIKO: ANNA -------Stack of characters------- A N N A -------Queue of characters------- A N N A ANNA ACCEPTED	DWSE TO ALFARITHMITIKO: KALIMERA -------Stack of characters------- A R E M I L A K -------Queue of characters------- K A L I M E R A KALIMERA REJECTED

10. Να γίνει πρόγραμμα που θα προσομοιώνει μια ουρά με τη βοήθεια δύο στοιβών, δηλαδή οι λειτουργίες της ουράς θα προσομοιώνονται με τις λειτουργίες της στοίβας. Αντί να χρησιμοποιήσετε μια ουρά αρκεί να χρησιμοποιήσετε 2 στοίβες. Κάθε κόμβος περιέχει έναν ακέραιο αριθμό και η εισαγωγή δεδομένων θα γίνεται ως εξής: πλήθος στοιχείων, στοιχεία. Το πρόγραμμα θα εμφανίζει τα περιεχόμενα και των 2 στοιβών. Η $2^η$ στοίβα έχει καταχωρημένα τα στοιχεία όπως θα ήταν αν είχαμε χρησιμοποιήσει μια ουρά. Δίνεται ένα στιγμιότυπο από τη εκτέλεση του προγράμματος όπου φαίνονται τα δεδομένα εισόδου και η εμφάνιση των αποτελεσμάτων.

PLITHOS STOIXEIWN: 5
DWSE TO 1o STOIXEIO: 23
DWSE TO 2o STOIXEIO: 12
DWSE TO 3o STOIXEIO: 7
DWSE TO 4o STOIXEIO: 35
DWSE TO 5o STOIXEIO: 3
*********1i stoiva**********
3 35 7 12 23
*********2i stoiva**********
23 12 7 35 3

11. Σε ένα υπολογιστικό σύστημα είναι επιτρεπτή η πρόσβαση (login) μόνον αν κάθε χρήστης διαθέτει αντίστοιχο λογαριασμό που εκφράζεται με έναν USER_ID (αριθμός ταυτότητας χρήστη). Τα USER-IDs των χρηστών είναι αποθηκευμένα στο αρχείο κειμένου "I11F4.DAT". Κάθε χρήστης μπορεί να κάνει login από ένα μόνο τερματικό. Κάθε φορά που ένας χρήστης κάνει login τότε το USER_ID του προστίθεται σε μια ουρά. Να γίνει πρόγραμμα που θα εμφανίζει το μήνυμα '*USERNAME: '*, θα δέχεται από το πληκτρολόγιο το USER_ID (*string* μέχρι 8 χαρακτήρες) και θα ελέγχει αν ο χρήστης με το παραπάνω USER_ID έχει λογαριασμό στο σύστημα (υπάρχει δηλαδή τέτοιο USER_ID στο αρχείο). Η εισαγωγή των USER_ID

των χρηστών θα ελέγχεται από το μήνυμα '*New entry Y/N (Y=Yes, N=No)?*'.

i) Αν δεν έχει λογαριασμό, τότε θα εμφανίζει το μήνυμα '*Wrong user ID*' και το πρόγραμμα θα συνεχίζει με νέο USER_ID.

ii) Αν έχει λογαριασμό, τότε θα ελέγχει αν το USER_ID είναι στοιχείο της ουράς. Αν δεν είναι τότε θα προσθέτει το USER_ID στην ουρά. Αν είναι στοιχείο της ουράς, δηλαδή έχει ήδη κάνει login στο σύστημα από άλλο τερματικό, τότε θα εμφανίζει το μήνυμα '*You have logged in to the system from another terminal. New access is forbidden*' και το πρόγραμμα θα συνεχίζει με νέο USER-ID.

iii) Όταν τερματιστεί η εισαγωγή των USER_ID των χρηστών τότε θα εμφανίζει τα στοιχεία της ουράς

12. Μια αεροπορική εταιρεία κρατά τα στοιχεία των ατόμων που θα ταξιδέψουν με μια συγκεκριμένη πτήση. Για κάθε επιβάτη κρατούνται τα παρακάτω στοιχεία:

-επώνυμο και
-όνομα τύπου *String 15 χαρακτήρες*,
-τηλέφωνο τύπου *Longint*,
-τύπος εισιτηρίου (0=ολόκληρο, 1=μισό).

Την ημέρα πραγματοποίησης της πτήσης οι επιβάτες που δεν έχουν κρατήσει θέση και επιθυμούν να ταξιδέψουν εγγράφονται σε ουρά (ουρά αναμονής). Για κάθε επιβάτη που γράφεται στην ουρά αναμονής κρατούνται τα στοιχεία του. Να γίνει πρόγραμμα που θα εκτελεί τις παρακάτω λειτουργίες:

-Εισαγωγή του πλήθους των ατόμων που θα καταχωρηθούν στην ουρά αναμονής.
-Εισαγωγή στοιχείων κάθε ατόμου στην ουρά αναμονής
-Διάσχιση της ουράς αναμονής και εμφάνιση των στοιχείων
-Εισαγωγή ελεύθερων θέσεων πτήσης και διαγραφή των αντίστοιχων επιβατών από την ουρά αναμονής
-Διάσχιση της ουράς αναμονής και εμφάνιση των στοιχείων

Όταν οι διαδικασίες εισόδου/εξόδου δεδομένων και αποτελεσμάτων γίνονται:

- Α' τρόπος:
  i) οι διαδικασίες θα καλούνται από το κυρίως πρόγραμμα σειριακά
- Β' τρόπος:
  i) οι διαδικασίες θα καλούνται μέσω καταλόγου επιλογών (menu driven)
  ii) η καταχώρηση των στοιχείων των επιβατών θα ελέγχεται από το μήνυμα '*New entry Y/N (Y=Yes, N=No)?*', οπότε δε θα διαβάζεται το πλήθος τους από το κυρίως πρόγραμμα.

13. Ο Administrator του δικτύου του εργαστηρίου Η/Υ του τμήματος Εφ. Πληροφορικής, αποφάσισε να δημιουργήσει 2 ουρές εξυπηρέτησης των εργασιών προς εκτύπωση στον LASER εκτυπωτή. Στην 1η ουρά θα οδηγούνται οι εργασίες των χρηστών USERX και FORTX, όπου Χ οποιοσδήποτε χαρακτήρας/ες, ενώ στη 2η ουρά οι εργασίες των διδασκόντων. Θεωρήστε ότι οι πληροφορίες που αφορούν κάθε εργασία για εκτύπωση είναι αποθηκευμένες στο αρχείο κειμένου "I13F4.DAT" που θα δημιουργήσετε με την εξής γραμμογράφηση:

όνομα χρήστη (*αλφαριθμητικό 8 χαρακτήρες*)
όνομα αρχείου (*αλφαριθμητικό 12 χαρακτήρες*)

π.χ.

USER7
ask10.pas
MAYA
text.doc
.....

Ζητείται να γίνει πρόγραμμα που:

i) Θα διαβάζει μία-μία τις γραμμές του αρχείου κειμένου και θα δημιουργεί τις 2 ουρές ως εξής: αν το όνομα του χρήστη που απαιτεί εξυπηρέτηση είναι ένας από τους USERX ή FORTX, τότε η εργασία του θα κατευθύνεται στην ουρά 1, ενώ αν είναι οποιοσδήποτε άλλος η εργασία του θα κατευθύνεται στην ουρά 2.

ii) Θα κάνει προσομοίωση της εξυπηρέτησης των εργασιών ως εξής: θα εμφανίζονται στην οθόνη τα μηνύματα: '*Printing the job of όνομα-χρήστη, όνομα-αρχείου*' και σε άλλη γραμμή '*Press any key to continue*' π.χ.:

Printing the job of USER7, ask10.pas
Press any key to continue

Μόλις πατηθεί ένα οποιοδήποτε πλήκτρο θα γίνεται διαγραφή του αντίστοιχου κόμβου από την αντίστοιχη ουρά. Η εξυπηρέτηση των εργασιών θα γίνεται πρώτα για εκείνες που βρίσκονται στην ουρά 2 και μετά για εκείνες που βρίσκονται στην ουρά 1. Να γίνονται οι σχετικοί έλεγχοι που αφορούν στο αν οι ουρές είναι άδειες και όταν εξαντληθούν τα στοιχεία και των 2 ουρών να εμφανίζεται το μήνυμα '*There are no jobs in the printing queue*'.

14. Τα σύμβολα (,), [,], {, } ονομάζονται παρενθετικά σύμβολα. Μία παράσταση που περιέχει μόνο παρενθέσεις (),

αγκύλες [] και άγκιστρα {} ονομάζεται ορθώς σχηματισμένη αν και μόνο αν υπακούει στους παρακάτω κανόνες:

i) Οι παραστάσεις (), [] και {} είναι ορθώς σχηματισμένες.

ii) Αν μια ακολουθία παρενθετικών συμβόλων *X* είναι ορθώς σχηματισμένη, τότε και οι παραστάσεις (*X*), [*X*] και {*X*} είναι ορθώς σχηματισμένες.

iii) Αν οι ακολουθίες παρενθετικών συμβόλων *X* και *Y* είναι ορθώς σχηματισμένες, τότε είναι και η *XY*.

Να γραφεί ένα πρόγραμμα που για δεδομένη ακολουθία παρενθετικών συμβόλων -που θα εισάγεται σε μορφή αλφαριθμητικού- θα αποφαίνεται αν είναι ορθώς σχηματισμένη ή όχι κι εμφανίζει αντίστοιχα τα μηνύματα '*CORRECT*' ή '*WRONG*'.

(***Υπόδειξη:*** Χρησιμοποιήστε μία στοίβα όπου θα αποθηκεύετε τα διαδοχικά "αριστερά" παρενθετικά σύμβολα που θα συναντάτε σαρώνοντας την ακολουθία από τα αριστερά προς τα δεξιά).

15. Το σύστημα εξυπηρέτησης αφίξεων των φορτίων μιας επιχείρησης τοποθετεί κάθε φορτίο σε μία από τις τρεις διαθέσιμες αποθήκες (κάθε αποθήκη λειτουργεί ως στοίβα).

Για κάθε φορτίο αποθηκεύονται τα παρακάτω στοιχεία:

-αριθμός κιβωτίων που περιλαμβάνονται στο συγκεκριμένο φορτίο

-ημερομηνία λήξης

Κάθε φορτίο τοποθετείται στην αποθήκη που περιέχει τον μικρότερο συνολικό αριθμό κιβωτίων.

Τα στοιχεία κάθε στοίβας δεν είναι ταξινομημένα ως προς την ημερομηνία λήξης. Να γραφεί πρόγραμμα που θα περιλαμβάνει τις παρακάτω λειτουργίες:

i) Εισαγωγή συνολικού αριθμού φορτίων

ii) Εισαγωγή φορτίου στην αντίστοιχη στοίβα. Για κάθε φορτίο θα εισάγετε τον αριθμό κιβωτίων και την ημερομηνία λήξης. Κάθε φορτίο θα τοποθετείται στην αποθήκη που περιέχει τον μικρότερο συνολικό αριθμό κιβωτίων και σε περίπτωση ισοπαλίας (ίδιου συνολικού αριθμού κιβωτίων) επιλέγεται η στοίβα με το μικρότερο αύξοντα αριθμό. Υλοποιήστε 2 συναρτήσεις:

- Μια συνάρτηση που δέχεται τις 3 αποθήκες (στοίβες), προσδιορίζει τη στοίβα με τον μικρότερο συνολικό αριθμό κιβωτίων και επιστρέφει έναν ακέραιο που αντιστοιχεί στον αύξοντα αριθμό της στοίβας (δηλαδή ή 1 ή 2 ή 3) με τον μικρότερο συνολικό αριθμό κιβωτίων.
- Μια συνάρτηση που δέχεται μια αποθήκη (μια στοίβα) και επιστρέφει έναν ακέραιο που είναι ο συνολικός αριθμός κιβωτίων της στοίβας.

iii) Διαδικασία για την εμφάνιση των στοιχείων των φορτίων κάθε αποθήκης.

Δίνεται ένα στιγμιότυπο από τη εκτέλεση του προγράμματος όπου φαίνονται τα δεδομένα εισόδου και η εμφάνιση των αποτελεσμάτων.

```
Plithos fortiwn: 5
Arithmos kivwtiwn gia to 1o fortio: 4
Imeromhnia liksis gia to 1o fortio: 14/05/2016
Arithmos kivwtiwn gia to 2o fortio: 5
Imeromhnia liksis gia to 2o fortio: 21/12/2016
Arithmos kivwtiwn gia to 3o fortio: 2
Imeromhnia liksis gia to 3o fortio: 31/08/2017
Arithmos kivwtiwn gia to 4o fortio: 3
Imeromhnia liksis gia to 4o fortio: 12/12/2016
Arithmos kivwtiwn gia to 5o fortio: 6
Imeromhnia liksis gia to 5o fortio: 06/06/2016
**********WAREHOUSE 1**********
6 06/06/2016
4 14/05/2016
**********WAREHOUSE 2**********
5 21/12/2016
**********WAREHOUSE 3**********
3 12/12/2016
2 31/08/2017
```

16. Σε μία ουρά κάθε στοιχείο εισέρχεται σε κάποια συγκεκριμένη θέση σύμφωνα με το βαθμό προτεραιότητας (σε αύξουσα σειρά). Να γραφεί πρόγραμμα που θα περιλαμβάνει τα παρακάτω:

-Εισαγωγή του πλήθους των κόμβων της ουράς.

- Τη συνάρτηση `void insert_prot(QueueType *Queue, QueueElementType Item)` την εισαγωγή στοιχείων στην ουρά, κάθε κόμβος της οποίας θα περιέχει έναν τριψήφιο κωδικό αριθμό και τον βαθμό προτεραιότητας (1-20). Σε περίπτωση ίδιου βαθμού προτεραιότητας το στοιχείο εισέρχεται τελευταίο στην αντίστοιχη προτεραιότητα.
- Τη συνάρτηση `void TraverseQ(QueueType Queue)` που θα εμφανίζει τα περιεχόμενα της ουράς κατά αύξοντα βαθμό προτεραιότητας. Τα στοιχεία κάθε κόμβου εμφανίζονται σε ξεχωριστή σειρά με ένα κενό μεταξύ τους και πρώτο το βαθμό προτεραιότητας.

Στη συνέχεια δίνεται ένα στιγμιότυπο εκτέλεσης

```
DWSE TO PLITHOS: 6
DWSE TON KODIKO TOU 1ou KOMVOU: 123
DWSE TO VATHMO PROTERAIOTITAS TOU 1ou KOMVOU: 2
DWSE TON KODIKO TOU 2ou KOMVOU: 234
DWSE TO VATHMO PROTERAIOTITAS TOU 2ou KOMVOU: 1
DWSE TON KODIKO TOU 3ou KOMVOU: 345
DWSE TO VATHMO PROTERAIOTITAS TOU 3ou KOMVOU: 6
DWSE TON KODIKO TOU 4ou KOMVOU: 456
DWSE TO VATHMO PROTERAIOTITAS TOU 4ou KOMVOU: 1
DWSE TON KODIKO TOU 5ou KOMVOU: 567
DWSE TO VATHMO PROTERAIOTITAS TOU 5ou KOMVOU: 3
DWSE TON KODIKO TOU 6ou KOMVOU: 678
DWSE TO VATHMO PROTERAIOTITAS TOU 6ou KOMVOU: 2
----------Priority Queue----------
1 234
1 456
2 123
2 678
3 567
6 345
```

17. Υλοποιείστε μία κυκλική ουρά. Σαν εφαρμογή επιλύστε με μία τεχνική κυκλικών ουρών το ακόλουθο πρόβλημα (πρόβλημα του Josephus):

N φυλακισμένοι είναι αριθμημένοι από το 1 ως το *N* και σχηματίζεται ένας κύκλος. Έστω ότι εκτελείται κάθε *M*-οστός φυλακισμένος, πλην του τελευταίου στον οποίο χαρίζουν τη ζωή. Ποιος θα επιζήσει;

Για παράδειγμα έστω ότι οι φυλακισμένοι είναι 5 κι εκτελείται κάθε τρίτος φυλακισμένος. Τότε η ακολουθία των εκτελέσεων και τα μηνύματα που πρέπει να εμφανιστούν θα έχουν ως εξής:

1 2 3 εκτελείται ο φυλακισμένος με αύξοντα αριθμό 3 '*EXECUTION 3*'
4 5 1 εκτελείται ο φυλακισμένος με αύξοντα αριθμό 1 '*EXECUTION 1*'
2 4 5 εκτελείται ο φυλακισμένος με αύξοντα αριθμό 5 '*EXECUTION 5*'
2 4 2 εκτελείται ο φυλακισμένος με αύξοντα αριθμό 2 '*EXECUTION 2*'
Επιζεί ο φυλακισμένος με αύξοντα αριθμό 4 '*SURVIVAL 4*'

Να γραφεί πρόγραμμα που θα διαβάζει το πλήθος *N* των φυλακισμένων και τον αριθμό *M* του κύκλου εκτέλεσης και θα υλοποιεί το πρόβλημα του Josephus.

18. Προσομοιώστε τη λειτουργία μίας ουράς σε μία Τράπεζα. Η προσομοίωση θα γίνει
(a) με τη βοήθεια πινάκων
(b) με τη βοήθεια δεικτών (pointers).

Το κυρίως πρόγραμμα θα έχει τη μορφή ενός βρόγχου όπως ο παρακάτω:

Repeat
    επεξεργασία ταμείου
    επεξεργασία ουράς
Until TELOS

Κάθε ανακύκλωση έχει το νόημα της μεταβολής του χρόνου - έτσι για παράδειγμα κάθε ανακύκλωση μπορεί να αντιστοιχεί σε ένα λεπτό και η συνθήκη τέλος μπορεί να σημαίνει ότι ο βρόγχος εκτελέστηκε 300 φορές - δηλαδή προσομοιώνεται η λειτουργία της Τράπεζας από 60x50 =300 λεπτά. Κάθε πελάτης χαρακτηρίζεται από δύο στοιχεία:

-το όνομα του
-το χρόνο εξυπηρέτησης - δηλαδή το χρόνο που χρειάζεται η εργασία του προκειμένου να διεκπεραιωθεί.

Η επεξεργασία του ταμείου περιλαμβάνει τα εξής:

i) σε κάθε ανακύκλωση, αν υπάρχει πελάτης, ο *XE* (χρόνος εξυπηρέτησης) του πελάτη μειώνεται κατά μία μονάδα έως ότου μηδενιστεί - δηλαδή ο πελάτης εξυπηρετηθεί και φύγει.
ii) αν δεν υπάρχει πελάτης στο ταμείο αλλά υπάρχουν πελάτες στην ουρά αναμονής, ο επόμενος πελάτης προσέρχεται στο ταμείο.

Η επεξεργασία της ουράς περιλαμβάνει τα εξής: σε κάθε ανακύκλωση μπορεί να προστεθεί ένας νέος πελάτης με ένα ορισμένο αριθμό κωδικοποίησης.

Η εμφάνιση περιλαμβάνει την εκτύπωση (στην οθόνη) της ουράς σε κάθε χρονική στιγμή (δηλαδή σε κάθε ανακύκλωση).

19. Σε προγράμματα επεξεργασίας κειμένου (text editors), η σύμβαση που χρησιμοποιείται για να διευκρινιστεί ότι ένα τμήμα κειμένου είναι υποσημείωση ή σχόλιο είναι να σημειωθεί με κάποια ειδικά διαχωριστικά σύμβολα όπως '//', '//', '/*' και '*/'. Όταν το κείμενο διαμορφώνεται προς εκτύπωση, το συγκεκριμένο τμήμα δεν εκτυπώνεται σαν κανονικό κείμενο αλλά αποθηκεύεται σε μια ουρά για το τέλος. Να γραφεί πρόγραμμα που θα εκτελεί την παραπάνω λειτουργία, δηλαδή θα

εμφανίζει το κείμενο χωρίς σχόλια και στο τέλος του κειμένου θα εμφανίζονται τα σχόλια αφού πρώτα σημειωθεί σε ξεχωριστή γραμμή ο χαρακτήρας έναρξης των σχολίων '{'. Αφού προστεθούν όλα τα σχόλια θα κλείσει το αρχείο κειμένου με το χαρακτήρα τέλους σχόλιων '}'. (Ως κείμενο μπορεί να χρησιμοποιηθεί ένα πρόγραμμα C με σχόλια σε μορφή // .. // ή /*... */. Θεωρείστε ότι το μέγιστο μήκος σχολίων δεν μπορεί να υπερβαίνει τους 255 χαρακτήρες).

20. Να τροποποιήσετε-επεκτείνετε την Άσκηση 9 του "Φύλλου Ασκήσεων 3: Ουρές" έτσι ώστε:

- τα στοιχεία των ασθενών για τους οποίους δεν μπορούμε να κλείσουμε ραντεβού στην επιθυμητή κλινική να αποθηκεύονται σε μία λίστα.
- η διαδικασία *showWaitingQ* θα δέχεται πλέον τη λίστα (και όχι μια ουρά) των ασθενών που είναι σε αναμονή και θα εμφανίζει τα στοιχεία τους.
- θα γράψετε διαδικασία *cancelAppointment* που θα προσομοιώνει τη λειτουργία ακύρωσης ενός ραντεβού και εξυπηρέτησης ενός ασθενούς από τη λίστα αναμονής. Συγκεκριμένα, η διαδικασία θα δέχεται τις 5 ουρές των ασθενών και τη λίστα αναμονής, και στο σώμα της: (1) θα διαβάζεται ο κωδικός *code* (*integer*) της κλινικής (1, 2, 3, 4 ή 5), (2) θα διαγράφεται το 1^{ο} ραντεβού από την κατάλληλη ουρά, (3) θα πραγματοποιείται αναζήτηση στη λίστα αναμονής για την εύρεση του πρώτου ασθενούς που είναι σε αναμονή για ραντεβού στη συγκεκριμένη κλινική, (4) ο συγκεκριμένος ασθενής θα διαγράφεται από τη λίστα αναμονής και θα προστίθεται στην κατάλληλη ουρά (το ονοματεπώνυμό του), (5) θα εμφανίζεται το παρακάτω μήνυμα προκειμένου να ενημερώσει ο υπάλληλος του τηλεφωνικού κέντρου των εξωτερικών ιατρείων τον ασθενή που έφυγε από τη λίστα αναμονής για το ραντεβού του: *Call <αριθμός-τηλεφώνου> to inform <ονοματεπώνυμο-ασθενή> for his/her appointment at clinic <κωδικός-κλινικής>*

Οι παραπάνω λειτουργίες θα εκτελούνται μέσω μενού επιλογών:

Appointment scheduling application

1. New appointment
2. Cancel appointment
3. Show scheduled appointments for all clinics
4. Show waiting list
5. Exit

Χρησιμοποιήστε τους ΑΤΔ Συνδεδεμένη Λίστα και Ουρά με δείκτες για την υλοποίηση της ΑΣΛ και των ουρών αντίστοιχα..

21. Σε μια ΑΣΛ θέλουμε να καταχωρούμε τον Αριθμό Μητρώου (int) και τον βαθμό (float) ενός μαθητή. Τροποποιείστε τον τύπο ListElementType ώστε να αποθηκεύει τον ΑΜ και το βαθμό του μαθητή. Κάντε τις κατάλληλες τροποποιήσεις σε όσες συναρτήσεις είναι αναγκαίο (**L_ListADT.c & L_ListADT.h**). Στη συνέχεια, φτιάξτε ένα πρόγραμμα που, χρησιμοποιώντας τις τροποποιημένες συναρτήσεις θα κάνει τα εξής:

i. Θα δημιουργεί μία κενή ΣΛ (μέγιστο 20).
ii. Θα διαβάζει το πλήθος των μαθητών που θέλουμε να εισάγουμε (μέγιστο 20).
iii. Θα διαβάζει και θα εισάγει έναν προς έναν τον κάθε μαθητή και το βαθμό του. Έπειτα από την εισαγωγή του κάθε ένα θα εμφανίζει τα περιεχόμενα της ΣΛ, <u>όπως φαίνεται στο στιγμιότυπο εκτέλεσης</u>.
iv. Θα διαγράφει έναν μαθητή, αφού διαβάσει τη θέση του προηγούμενου στη ΣΛ και θα εμφανίζει το περιεχόμενο αυτής, <u>όπως φαίνεται στο στιγμιότυπο εκτέλεσης.</u>
v. Θα διαβάζει και θα εισάγει δύο νέους μαθητές και τους βαθμούς τους στη λίστα. Έπειτα από την εισαγωγή του κάθε ένα θα εμφανίζει τα περιεχόμενα της ΣΛ, <u>όπως φαίνεται στο στιγμιότυπο εκτέλεσης</u>.

```
DWSE ARI8MO MA8HTWN:3
DWSE ARI8MO MHTRWOU GIA EISAGWGH STH LISTA: 101
DWSE BA8MO GIA EISAGWGH STH LISTA: 17.3
DWSE TH 8ESH META THN OPOIA 8A GINEI H EISAGWGH STOIXEIOY: -1

Plithos soixeiwn sth lista 1
[0: (101,17.3) ->-1]
DWSE ARI8MO MHTRWOU GIA EISAGWGH STH LISTA: 202
DWSE BA8MO GIA EISAGWGH STH LISTA: 12.4
DWSE TH 8ESH META THN OPOIA 8A GINEI H EISAGWGH STOIXEIOY: -1

Plithos soixeiwn sth lista 2
[1: (202,12.4) ->0] [0: (101,17.3) ->-1]
DWSE ARI8MO MHTRWOU GIA EISAGWGH STH LISTA: 303
DWSE BA8MO GIA EISAGWGH STH LISTA: 20
DWSE TH 8ESH META THN OPOIA 8A GINEI H EISAGWGH STOIXEIOY: 1

Plithos soixeiwn sth lista 3
[1: (202,12.4) ->2] [2: (303,20) ->0] [0: (101,17.3) ->-1]
DWSE TH 8ESH TOY PROHGOUMENOY STOIXEIOY GIA DIAGRAFH: 1

Plithos soixeiwn sth lista 2
[1: (202,12.4) ->0] [0: (101,17.3) ->-1]
DWSE ARI8MO MHTRWOU GIA EISAGWGH STH LISTA: 909
DWSE BA8MO GIA EISAGWGH STH LISTA: 14.1
DWSE TH 8ESH META THN OPOIA 8A GINEI H EISAGWGH STOIXEIOY: 0

Plithos soixeiwn sth lista 3
[1: (202,12.4) ->0] [0: (101,17.3) ->2] [2: (909,14.1) ->-1]
DWSE ARI8MO MHTRWOU GIA EISAGWGH STH LISTA: 808
DWSE BA8MO GIA EISAGWGH STH LISTA: 9.9
DWSE TH 8ESH META THN OPOIA 8A GINEI H EISAGWGH STOIXEIOY: -1

Plithos soixeiwn sth lista 4
[3: (808,9.9) ->1] [1: (202,12.4) ->0] [0: (101,17.3) ->2] [2: (909,14.1) ->-1]
```

Η εμφάνιση των στοιχείων της ΣΛ θα είναι ως εξής: [θέση πίνακα: (ΑΜ,Βαθμός) -> NEXT] (**δείτε το στιγμιότυπο εκτέλεσης**)

Θεωρήστε ότι οι αριθμοί μητρώου που εισάγουν οι χρήστες είναι μοναδικοί.

Προσοχή η άσκηση θα λυθεί με την υλοποίηση Συνδεδεμένης Λίστας. Ενότητα "4.3 Υλοποίηση ΑΤΔ Συνδεδεμένη Λίστα με πίνακα". Αρχεία κώδικα που θα τροποποιήσετε είναι τα L_ListADT.c & L_ListADT.h. Επίσης μπορείτε να συμβουλευτείτε και το πρόγραμμα πελάτη με τη main() που βρίσκεται στο φάκελο Project_L_List)

22. Σε μια ΑΣΛ θέλουμε να καταχωρούμε το ονοματεπώνυμο (char[20]) και τον βαθμό (float) ενός μαθητή. Τροποποιείστε τον τύπο ListElementType ώστε να αποθηκεύει το ονοματεπώνυμο και το βαθμό του μαθητή. Κάντε τις κατάλληλες τροποποιήσεις σε όσες συναρτήσεις είναι αναγκαίο (L_ListADT.c & L_ListADT.h). Ελέγξτε όλες τις συναρτήσεις με κώδικα αντίστοιχο της TestL_List.c (φάκελος Project_L_List). (Προσοχή στην ανάθεση αλφαριθμητικών χρήση της συνάρτησης strcpy(..))

23. Σε μια ΑΣΛ θέλουμε να καταχωρούμε τον Αριθμό Μητρώο (int) και τον βαθμό (float) ενός μαθητή. Τροποποιείστε τον τύπο ListElementType ώστε να αποθηκεύει τον ΑΜ και το βαθμό του μαθητή. Κάντε τις κατάλληλες τροποποιήσεις σε όσες συναρτήσεις, που θα αντιγράψετε από τα L_ListADT.c & L_ListADT.h, είναι αναγκαίο. Θέλουμε να βρίσκουμε και να εμφανίζουμε τους μαθητές που έχουν τον μεγαλύτερο βαθμό. Να υλοποιήσετε τη συνάρτηση FindMaxs που θα δέχεται 2 παραμέτρους: τη ΑΣΛ με τα στοιχεία των μαθητών και μια στοίβα όπου θα καταχωρεί τους ΑΜ όλων των μαθητών με τον ίδιο μέγιστο βαθμό, και θα επιστρέφει την τιμή του μέγιστου βαθμού. Αν η λίστα είναι άδεια η FindMaxs θα εμφανίζει Empty list και θα επιστρέφει τιμή max = -1. Το πρωτότυπο της συνάρτησης:

```
float FindMaxs(ListPointer List, NodeType Node[NumberOfNodes], StackType *Stack)
```

Αφού βρεθούν μέσω της FindMaxs οι μαθητές με τον μέγιστο βαθμό, στο κυρίως πρόγραμμα θα εμφανίζονται τα στοιχεία τους στην οθόνη. Η εμφάνιση του ΑΜ των μαθητών θα γίνεται με χρήση των επιτρεπόμενων λειτουργιών της στοίβας και στη συνέχεια θα καλείται η βοηθητική συνάρτηση TraverseStack για την επαλήθευση της ορθής υλοποίησης αυτού του ερωτήματος (η στοίβα θα είναι κενή). Το περιεχόμενο κάθε κόμβου της ΑΣΛ θα εμφανίζεται ως εξής: <ΑΜ, Βαθμός> , πχ αν ο κόμβος της ΑΣΛ είναι στη θέση 0 του πίνακα με ΑΜ 21 και βαθμό 9 και ο επόμενος κόμβος της ΑΣΛ είναι αποθηκευμένος στη θέση 3 του πίνακα, τότε η εμφάνιση του κόμβου θα είναι: (0:<21,9.0> -> 3) που σημαίνει ότι ο κόμβος με περιεχόμενο <21, 9> είναι αποθηκευμένος στη θέση 0 του πίνακα και έχει επόμενο κόμβο αποθηκευμένο στη θέση 3 του πίνακα. Το πρόγραμμα σας θα πρέπει να εκτελεί κατά τη σειρά τις παρακάτω λειτουργίες και οι κλήσεις των συναρτήσεων θα γίνονται από τη main().

1. Αρχικοποίηση storage pool
2. Δημιουργία ΑΣΛ (μέγιστο μέγεθος 10)
3. Εμφάνιση της storage pool
4. Εμφάνιση των στοιχείων της ΑΣΛ
5. Εισαγωγή 5 στοιχείων στην ΑΣΛ (Η εισαγωγή στοιχείου θα γίνεται στην αρχή της ΑΣΛ)
6. Εμφάνιση της storage pool
7. Εμφάνιση των στοιχείων της ΑΣΛ
8. Έλεγχος αν η ΑΣΛ είναι άδεια. Αν η ΑΣΛ είναι άδεια εμφανίζει μήνυμα «Empty List» διαφορετικά «Not an Empty List»
9. Έλεγχος εάν η ΑΣΛ είναι γεμάτη. Αν η ΑΣΛ είναι γεμάτη εμφανίζει μήνυμα «Full List» διαφορετικά «Not a Full List»
10. Εμφάνιση του μέγιστου βαθμού και των ΑΜ των μαθητών με το μέγιστο βαθμό
11. Κλήση της TraverseStack για την επαλήθευση της ορθής υλοποίησης του ερωτήματος 10. Η εμφάνιση των στοιχείων της στοίβας από τη θέση Top και προς «τα κάτω» (Top – 0).
12. Εμφάνιση της storage pool
13. Εμφάνιση των στοιχείων της ΑΣΛ

Στη συνέχεια δίνεται ένα στιγμιότυπο εκτέλεσης:

```
1o STOIXEIO LISTAS=-1, 1H FREE POSITION=0
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<-1,-1.0>  ->1)  (1:<-1,-1.0>  ->2)  (2:<-1,-1.0>  ->3)  (3:<-1,-1.0>  ->4)  (4:<-1,-1.0>  ->5)
(5:<-1,-1.0> ->6) (6:<-1,-1.0> ->7) (7:<-1,-1.0> ->8) (8:<-1,-1.0> ->9) (9:<-1,-1.0> ->-1)
Question 4
Linked list
Empty List ...
Question 5
DWSE AM GIA EISAGWGH STH LISTA: 20
DWSE VATHMO GIA EISAGWGH STH LISTA: 5
DWSE AM GIA EISAGWGH STH LISTA: 30
DWSE VATHMO GIA EISAGWGH STH LISTA: 8
DWSE AM GIA EISAGWGH STH LISTA: 40
DWSE VATHMO GIA EISAGWGH STH LISTA: 10
DWSE AM GIA EISAGWGH STH LISTA: 50
DWSE VATHMO GIA EISAGWGH STH LISTA: 6
DWSE AM GIA EISAGWGH STH LISTA: 60
DWSE VATHMO GIA EISAGWGH STH LISTA: 10
Question 6
Storage pool
1o STOIXEIO LISTAS=4, 1H FREE POSITION=5
```

```
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<20,5.0> ->-1) (1:<30,8.0> ->0) (2:<40,10.0> ->1) (3:<50,6.0> ->2) (4:<60,10.0> ->3) (5:<-
1,-1.0> ->6) (6:<-1,-1.0> ->7) (7:<-1,-1.0> ->8) (8:<-1,-1.0> ->9) (9:<-1,-1.0> ->-1)
Question 7
Linked list
(4:<60,10.0> ->3) (3:<50,6.0> ->2) (2:<40,10.0> ->1) (1:<30,8.0> ->0) (0:<20,5.0> ->-1)
Question 8
Not an Empty List
Question 9
Not a Full List
Question 10
Max vathmos=10.0
AM me megisto vathmo: 40 60
Question 11

plithos sto stack 0

Question 12
Storage pool
1o STOIXEIO LISTAS=4, 1H FREE POSITION=5
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<20,5.0> ->-1) (1:<30,8.0> ->0) (2:<40,10.0> ->1) (3:<50,6.0> ->2) (4:<60,10.0> ->3) (5:<-
1,-1.0> ->6) (6:<-1,-1.0> ->7) (7:<-1,-1.0> ->8) (8:<-1,-1.0> ->9) (9:<-1,-1.0> ->-1)
Question 13
Linked list
(4:<60,10.0> ->3) (3:<50,6.0> ->2) (2:<40,10.0> ->1) (1:<30,8.0> ->0) (0:<20,5.0> ->-1)
```

24. Γράψτε μία συνάρτηση sort_list(ListPointer *List, NodeType Node[], boolean Ascending) η οποία δέχεται μία ΑΣΛ με ακεραίους και την κατατάσσει σε αύξουσα ή φθίνουσα σειρά, ανάλογα με το εάν η μεταβλητή Ascending είναι αληθής ή όχι. Οι αλλαγές των θέσεων να γίνουν με αλλαγή των δεικτών Next και όχι με αλλαγή των δεδομένων κάθε θέσης. Το πρόγραμμα θα ζητάει αρχικά το πλήθος των στοιχείων της λίστας, η οποία θα μπορεί να φιλοξενήσει κατά μέγιστο 20 στοιχεία. Στη συνέχεια, θα διαβάζει τα στοιχεία, ένα-ένα. Αφού διαβαστούν τα στοιχεία, θα ζητά από το χρήστη να κατατάξει τα στοιχεία (1) με αύξουσα ή (2) με φθίνουσα σειρά, εισάγοντας τον αντίστοιχο αριθμό. Στη συνέχεια θα εμφανίζει τη ΣΛ αντίστοιχα ταξινομημένη.
Προσοχή η άσκηση θα λυθεί με την υλοποίηση Συνδεδεμένης Λίστας. Ενότητα "4.3 Υλοποίηση ΑΤΔ Συνδεδεμένη Λίστα με πίνακα". Αρχεία κώδικα που θα τροποποιήσετε είναι τα L_ListADT.c & L_ListADT.h. Επίσης μπορείτε να συμβουλευτείτε και το πρόγραμμα πελάτη με τη main() που βρίσκεται στο φάκελο Project_L_List)

```
Enter number of integers: 5
Enter an integer: 2
Enter an integer: -3
Enter an integer: 1
Enter an integer: 7
Enter an integer: 10
---------------------- Select --------------------------
1. Sort Ascending
2. Sort Descending
1
-3 1 2 7 10
```

```
Enter number of integers: 5
Enter an integer: 2
Enter an integer: -7
Enter an integer: 2
Enter an integer: 1
Enter an integer: 99
---------------------- Select --------------------------
1. Sort Ascending
2. Sort Descending
2
99 2 2 1 -7
```

25. Σε μία εταιρία μεταφορών, διατηρούνται αρχεία σχετικά με τα εμπορεύματα που πρέπει να παραδοθούν. Το κάθε εμπόρευμα χαρακτηρίζεται από το βάρος του (int), το κόστος μεταφοράς του (int) και το όνομά του (string 20 χαρακτήρων). Γράψτε πρόγραμμα, το οποίο να διαβάζει το πλήθος των εμπορευμάτων και στη συνέχεια τα χαρακτηριστικά του καθενός, με την παρακάτω σειρά

Όνομα
Βάρος
κόστος

Το κάθε εμπόρευμα αποθηκεύεται σε μία ΑΣΛ. Στη συνέχεια, δίνεται η δυνατότητα αύξουσας ή φθίνουσας ταξινόμησης της λίστας, με βάση οποιοδήποτε από τα τρία χαρακτηριστικά. Γράψτε μια συνάρτηση sort_list η οποία θα δέχεται ως είσοδο μια ΑΣΛ προς ταξινόμηση, μια λογική μεταβλητή Ascending που αν είναι αληθής θα γίνεται ταξινόμηση σε αύξουσα σειρά και αν είναι ψευδής σε φθίνουσα σειρά, και μια ακέραια μεταβλητή ByFeature που θα καθορίζει το χαρακτηριστικό με βάση το οποίο θα γίνει η ταξινόμηση (1-Όνομα,2-Βάρος,3-Κόστος).
Τα περιεχόμενα κάθε κόμβου της storage pool θα εμφανίζονται ως εξής: <Όνομα, Βάρος, Κόστος> , πχ αν ο κόμβος της ΣΛ είναι στη θέση 0 του πίνακα με όνομα "prwto", βάρος 5 και κόστος 3 και ο επόμενος κόμβος της ΑΣΛ είναι αποθηκευμένος στη θέση 3 του πίνακα, τότε η εμφάνιση του κόμβου θα είναι: (0:<prwto,5,3> -> 3). Η εμφάνιση των στοιχείων της ΑΣΛ θα εμφανίζει μόνο το περιεχόμενο τους θα είναι δηλαδή για τον παραπάνω κόμβο: (prwto,5,3). Στο κυρίως πρόγραμμα θα εκτελούνται οι παρακάτω λειτουργίες στη σειρά.
1. Αρχικοποίηση storage pool
2. Δημιουργία ΑΣΛ (μέγιστο μέγεθος 10)

3. Εμφάνιση της storage pool
4. Εμφάνιση των στοιχείων της ΑΣΛ
5. Ανάγνωση πλήθους εμπορευμάτων (n). (Να γίνεται έλεγχος και να διαβάζεται ο αριθμός μέχρι να δοθεί έγκυρη τιμή)
6. Εισαγωγή n στοιχείων στην ΑΣΛ (Η εισαγωγή στοιχείου θα γίνεται στην αρχή της ΑΣΛ)
7. Εμφάνιση της storage pool
8. Εμφάνιση των στοιχείων της ΑΣΛ
9. Ταξινόμηση της ΑΣΛ σε αύξουσα σειρά κατά όνομα
10. Εμφάνιση των στοιχείων της ΑΣΛ
11. Ταξινόμηση της ΑΣΛ σε αύξουσα σειρά κατά βάρος
12. Εμφάνιση των στοιχείων της ΑΣΛ
13. Ταξινόμηση της ΑΣΛ σε αύξουσα σειρά κατά κόστος
14. Εμφάνιση των στοιχείων της ΑΣΛ
15. Ταξινόμηση της ΑΣΛ σε φθίνουσα σειρά κατά όνομα
16. Εμφάνιση των στοιχείων της ΑΣΛ
17. Ταξινόμηση της ΑΣΛ σε φθίνουσα σειρά κατά βάρος
18. Εμφάνιση των στοιχείων της ΑΣΛ
19. Ταξινόμηση της ΑΣΛ σε φθίνουσα σειρά κατά κόστος
20. Εμφάνιση των στοιχείων της ΑΣΛ

Στη συνέχεια δίνεται ένα στιγμιότυπο εκτέλεσης:

```
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<,0,0> ->1) (1:<,0,0> ->2) (2:<,0,0> ->3) (3:<,0,0> ->4) (4:<,0,0> ->5) (5:<,0,0> ->6)
(6:<,0,0> ->7) (7:<,0,0> ->8) (8:<,0,0> ->9) (9:<,0,0> ->-1)
Empty List ...
Give objects number: 3
Give name: prwto
Give weight: 5
Give cost: 10
Give name: deytero
Give weight: 3
Give cost: 8
Give name: trito
Give weight: 10
Give cost: 3
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<prwto,5,10> ->-1) (1:<deytero,3,8> ->0) (2:<trito,10,3> ->1) (3:<,0,0> ->4) (4:<,0,0> ->5)
(5:<,0,0> ->6) (6:<,0,0> ->7) (7:<,0,0> ->8) (8:<,0,0> ->9) (9:<,0,0> ->-1)
(trito,10,3)
(deytero,3,8)
(prwto,5,10)

Name Ascending
(deytero,3,8)
(prwto,5,10)
(trito,10,3)

Weight Ascending
(deytero,3,8)
(prwto,5,10)
(trito,10,3)

Cost Ascending
(trito,10,3)
(deytero,3,8)
(prwto,5,10)

Name Descending
(trito,10,3)
(prwto,5,10)
(deytero,3,8)

Weight Descending
(trito,10,3)
(prwto,5,10)
(deytero,3,8)

Cost Descending
(prwto,5,10)
```

```
(deytero,3,8)
(trito,10,3)
```

26. Η συχνότητα μίας τιμής είναι η συχνότητα εμφάνισής της σε ένα σύνολο παρατηρήσεων.
Χρησιμοποιείστε δύο απλά συνδεδεμένες λίστες, για να υπολογίσετε τις αθροιστικές συχνότητες, με χρήση N ακέραιων παρατηρήσεων, για τιμές (κλάσεις) από 0 έως 9. Αφού δώσει τον αριθμό των παρατηρήσεων, ο χρήστης θα εισάγει μία – μία τις τιμές των παρατηρήσεων, οι οποίες θα εισάγονται στην πρώτη λίστα. Στο τέλος θα υπολογίζεται και θα εκτυπώνεται η συχνότητα για κάθε κλάση 0-9 (2η λίστα) με φθίνουσα σειρά. Για τη μοντελοποίηση της κάθε κλάσης χρησιμοποιείστε ένα struct με δύο ακέραια πεδία. Το 1ο είναι η κλάση (τιμές 0-9) και το 2ο η συχνότητα εμφάνισης.

```
Enter observations number: 10
Enter an observation: 1
Enter an observation: 2
Enter an observation: 3
Enter an observation: 4
Enter an observation: 5
Enter an observation: 9
Enter an observation: 8
Enter an observation: 7
Enter an observation: 1
Enter an observation: 2

(2:2) (1:2) (9:1) (8:1) (7:1) (5:1) (4:1) (3:1) (6:0) (0:0)
Press any key to continue
```

27. Στην κρυπτογραφία, μία μέθοδος ώστε να αποκρυφθεί πληροφορία, είναι τα δεδομένα να επεξεργάζονται από μία συνάρτηση η οποία τα μετασχηματίζει γραμμικά, δηλαδή με τη μορφή

$$f(x) = ax + b \bmod c$$

Όπου a,b σταθερές και το $\bmod c$ σημαίνει πως παίρνουμε το υπόλοιπο της διαίρεσης με το c (τελεστής %). Για να εισάγουμε και σύγχυση στα δεδομένα, μπορούμε να θεωρήσουμε ότι δεδομένα εισάγονται σε μία ουρά, και καθώς εξέρχονται από αυτήν, περνούν από την πράξη κρυπτογράφησης

$$f(x) = ax + b + di \bmod c$$

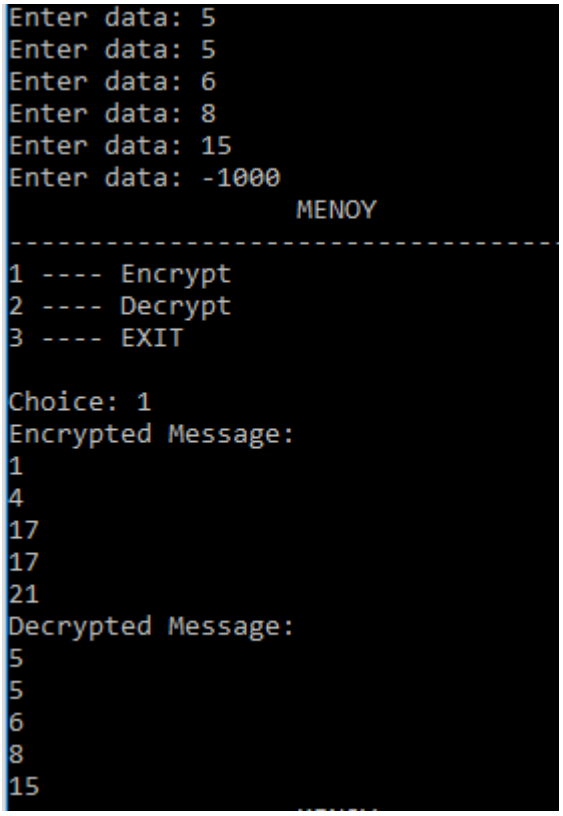
Όπου d είναι σταθερά και το i είναι η θέση του στοιχείου στην ουρά (ή αλλιώς πόσα έχουν εξαχθεί πριν από αυτό).
Γράψτε πρόγραμμα το οποίο δέχεται άγνωστο αριθμό δεδομένων (το πολύ 100), τα αποθηκεύει σε μία ουρά και όταν δεχθεί την τιμή -1000 τα κρυπτογραφεί και εμφανίζει το αρχικό και το κρυπτογραφημένο μήνυμα, με a=10,b=20,d=3,c=23.
ΠΡΟΣΟΧΗ: Το πρόγραμμα πρέπει να δέχεται μόνο μη αρνητικούς αριθμούς μικρότερους του c!!

```
Enter data: -1000
Original Message:
5
5
6
8
Encrypted Message:
1
4
17
17
```

28. Επεκτείνετε την άσκηση 27 ως εξής:
Ο χρήστης θα εισάγει τα δεδομένα με τον ίδιο τρόπο, αλλά πλέον θα μπορεί να επιλέξει ανάμεσα σε κρυπτογράφηση και αποκρυπτογράφηση.
Για την αποκρυπτογράφηση χρησιμοποιείστε την συνάρτηση

$$f(x) = a * (x + b + di) \bmod c$$

με τις τιμές:
a=7,b=3,d=20,c=23.
Μαζί με το (απο)κρυπτογραφημένο μήνυμα, εμφανίστε και το αρχικό, εφαρμόζοντας την αντίστροφη πράξη (δηλαδή εάν ο χρήστης θέλει κρυπτογράφηση, εφαρμόστε κρυπτογράφηση και στο κρυπτογραφημένο μήνυμα εφαρμόστε αποκρυπτογράφηση).



29. Μία εταιρία μεταφορών έχει διαθέσει ένα φορτηγό για να εκτελέσει μεταφορές από την Αθήνα στη Θεσσαλονίκη. Το φορτηγό λειτουργεί ως στοίβα (φορτώνει και εκφορτώνει από την πίσω πλευρά). Η συνολική του χωρητικότητα είναι 10 τόνοι. Τα πιο βαριά αντικείμενα πρέπει να βρίσκονται κάτω από τα ελαφριά. Η εταιρεία έχει ήδη φορτώσει από άλλους πελάτες τα εξής αντικείμενα:

Αντικείμενο	Βάρος (τόνοι)
1	3
2	2
3	1
4	0.5
5	0.4

Για να φορτωθεί ένα επιπλέον αντικείμενο πρέπει να χωράει στο φορτηγό και να συμφέρει την εταιρεία. Για κάθε αντικείμενο που θα εισαχθεί, πρέπει να ξεφορτωθούν πρώτα όλα τα ελαφρύτερα στην πλατφόρμα(δεύτερη στοίβα), με σκοπό να βρίσκεται κάτω από τα πιο ελαφριά και πάνω από τα πιο βαριά. Για κάθε αντικείμενο που αναδιατάσσεται (ξεφορτώνεται και φορτώνεται) η εταιρεία έχει κόστος το **βάρος του σε τόνους*300 ευρώ**. Γράψτε πρόγραμμα το οποίο διαβάζει αντικείμενα και την αξία μεταφοράς τους και αποφασίζει εάν συμφέρει την εταιρία ή όχι η εισαγωγή τους στο φορτηγό (αξιολόγηση γίνεται χωρίς να μεταφερθούν τα αντικείμενα στην πλατφόρμα και εφόσον συμφέρει, μόνο τότε μεταφέρονται). Το πρόγραμμα τερματίζει όταν εισαχθεί αρνητικό βάρος ή αξία. Με την εισαγωγή ενός στοιχείου, εκτυπώστε τα περιεχόμενα της πλατφόρμας και του φορτηγού, κατά τη διάρκεια φορτο-εκφόρτωσης και τα περιεχόμενα του φορτηγού στο τέλος.

```
Give the weight: 1
Give the value: 10
Not enough value
Give the weight: 1
Give the value: 1000
--Platform--
0.5
0.4
--Truck--
1
1
2
3
Give the weight: 0.1
Give the value: 30
--Platform--
EMPTY Stack
--Truck--
0.1
0.4
0.5
1
1
2
3
Give the weight: -1
--Truck--
0.1
0.4
0.5
1
1
2
3
```

30. Σε μια ΣΛ θέλουμε να καταχωρούμε τον Αριθμό Μητρώο (int) και τον βαθμό (float) ενός μαθητή. Τροποποιήστε τον τύπο ListElementType ώστε να αποθηκεύει τον ΑΜ και το βαθμό του μαθητή. Κάντε τις κατάλληλες τροποποιήσεις όπου είναι απαραίτητο, στις συναρτήσεις που θα αντιγράψετε από τα αρχεία L_ListADT.c & L_ListADT.h. Θέλουμε να εντοπίζουμε και να εμφανίσουμε τα στοιχεία των μαθητών που έχουν τον ελάχιστο βαθμό. Να υλοποιήσετε τη συνάρτηση FindMins που θα δέχεται 2 παραμέτρους: τη ΣΛ με τα στοιχεία των μαθητών και μια στοίβα όπου θα καταχωρεί τους ΑΜ όλων των μαθητών με τον ίδιο ελάχιστο βαθμό, και θα επιστρέφει την τιμή του ελάχιστου βαθμού. Το πρωτότυπο της συνάρτησης:

```
float FindMins(ListPointer List, NodeType Node[NumberOfNodes], StackType *Stack)
```

Αφού εντοπιστούν οι μαθητές με τους ελάχιστους βαθμούς, στη συνέχεια θα εμφανιστούν στην οθόνη.

Η εμφάνιση των ΑΜ των μαθητών με τον ίδιο ελάχιστο βαθμό θα γίνεται στη main() με χρήση των επιτρεπόμενων λειτουργιών της στοίβας (άδειασμα στοίβας). Στη συνέχεια θα καλείται η βοηθητική συνάρτηση TraverseStack για την επαλήθευση της ορθής υλοποίησης αυτού του ερωτήματος (η στοίβα θα πρέπει να είναι κενή μετά την «εμφάνιση»-διαγραφή από τη στοίβα των ΑΜ). Το περιεχόμενο κάθε κόμβου της ΣΛ θα εμφανίζεται ως εξής: <ΑΜ, Βαθμός> πχ αν ο κόμβος της ΣΛ είναι στη θέση 0 του πίνακα με ΑΜ 5 και βαθμό 10 και ο επόμενος κόμβος της ΣΛ είναι αποθηκευμένος στη θέση 2 του πίνακα, τότε η εμφάνιση του κόμβου θα είναι: (0|<5,10.0> -> 2) που σημαίνει ότι ο κόμβος με περιεχόμενο 5, 10 είναι αποθηκευμένος στη θέση 0 του πίνακα και έχει επόμενο κόμβο αποθηκευμένο στη θέση 2 του πίνακα.

Το πρόγραμμα θα πρέπει να εκτελεί κατά σειρά τις παρακάτω λειτουργίες και οι κλήσεις των συναρτήσεων θα γίνονται από τη main().

A. Αρχικοποίηση storage pool
B. Δημιουργία ΣΛ (μέγιστο μέγεθος 10)
C. Εμφάνιση της storage pool
D. Εμφάνιση των στοιχείων της ΣΛ
E. Εισαγωγή 5 στοιχείων στην ΣΛ. (Η εισαγωγή στοιχείου θα γίνεται πάντα στην αρχή της ΣΛ)
F. Εμφάνιση της storage pool
G. Εμφάνιση των στοιχείων της ΣΛ

H. Έλεγχος εάν η ΣΛ είναι άδεια. Αν η ΣΛ είναι άδεια εμφανίζει μήνυμα «Empty List» διαφορετικά «Not an Empty List»
I. Έλεγχος εάν η ΣΛ είναι γεμάτη. Αν η ΣΛ είναι γεμάτη εμφανίζει μήνυμα «Full List» διαφορετικά «Not a Full List»
J. Εμφάνιση του ελάχιστου βαθμού και των ΑΜ των μαθητών με το ελάχιστο βαθμό
K. Κλήση της TraverseStack για την επαλήθευση της ορθής υλοποίησης του ερωτήματος J
L. Εμφάνιση της storage pool
M. Εμφάνιση των στοιχείων της ΣΛ

Στη συνέχεια δίνεται ένα στιγμιότυπο εκτέλεσης:

```
--------------Question C--------------------
--------------Storage pool------------------
1o STOIXEIO LISTAS=-1, 1H FREE POSITION=0
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<-1,-1.0> ->1) (1:<-1,-1.0> ->2) (2:<-1,-1.0> ->3) (3:<-1,-1.0> ->4) (4:<-1,-1.0> ->5)
(5:<-1,-1.0> ->6) (6:<-1,-1.0> ->7) (7:<-1,-1.0> ->8) (8:<-1,-1.0> ->9) (9:<-1,-1.0> ->-1)
--------------Question D--------------------
--------------Linked list--------------------
Empty List ...
--------------Question E--------------------
DWSE AM GIA EISAGWGH STH LISTA: 3
DWSE VATHMO GIA EISAGWGH STH LISTA: 10
DWSE AM GIA EISAGWGH STH LISTA: 9
DWSE VATHMO GIA EISAGWGH STH LISTA: 5
DWSE AM GIA EISAGWGH STH LISTA: 1
DWSE VATHMO GIA EISAGWGH STH LISTA: 8
DWSE AM GIA EISAGWGH STH LISTA: 2
DWSE VATHMO GIA EISAGWGH STH LISTA: 5
DWSE AM GIA EISAGWGH STH LISTA: 7
DWSE VATHMO GIA EISAGWGH STH LISTA: 5
--------------Question F--------------------
--------------Storage pool------------------
1o STOIXEIO LISTAS=4, 1H FREE POSITION=5
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<3,10.0> ->-1) (1:<9,5.0> ->0) (2:<1,8.0> ->1) (3:<2,5.0> ->2) (4:<7,5.0> ->3) (5:<-1,-1.0>
->6) (6:<-1,-1.0> ->7) (7:<-1,-1.0> ->8) (8:<-1,-1.0> ->9) (9:<-1,-1.0> ->-1)
--------------Question G--------------------
--------------Linked list--------------------
(4:<7,5.0> ->3) (3:<2,5.0> ->2) (2:<1,8.0> ->1) (1:<9,5.0> ->0) (0:<3,10.0> ->-1)
--------------Question H--------------------
Not an Empty List
--------------Question I--------------------
Not a Full List
--------------Question J--------------------
Min value=5.0
AM with min grade are: 9 2 7
--------------Question K--------------------

plithos sto stack 0

--------------Question L--------------------
--------------Storage pool------------------
1o STOIXEIO LISTAS=4, 1H FREE POSITION=5
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<3,10.0> ->-1) (1:<9,5.0> ->0) (2:<1,8.0> ->1) (3:<2,5.0> ->2) (4:<7,5.0> ->3) (5:<-1,-1.0>
->6) (6:<-1,-1.0> ->7) (7:<-1,-1.0> ->8) (8:<-1,-1.0> ->9) (9:<-1,-1.0> ->-1)
--------------Question M--------------------
--------------Linked list--------------------
(4:<7,5.0> ->3) (3:<2,5.0> ->2) (2:<1,8.0> ->1) (1:<9,5.0> ->0) (0:<3,10.0> ->-1)
```

31. Η συχνότητα μίας τιμής είναι η συχνότητα εμφάνισής της σε ένα σύνολο παρατηρήσεων. Χρησιμοποιείστε δύο απλά συνδεδεμένες λίστες, για να υπολογίσετε τις αθροιστικές συχνότητες Ν ακέραιων παρατηρήσεων (μέγιστο πλήθος 25). Η πρώτη λίστα θα αποθηκεύει τις παρατηρήσεις. Οι παρατηρήσεις θα ανήκουν σε 10 κλάσεις, δηλαδή θα έχουν τιμές από 0 έως 9. Στη δεύτερη λίστα θα υπολογίζεται για κάθε τιμή η συχνότητα εμφάνισης της. Για τη μοντελοποίηση της κάθε κλάσης στη δεύτερη λίστα θα χρησιμοποιήσετε ένα struct με δύο ακέραια πεδία. Το 1ο είναι η κλάση (τιμές 0-9) και το 2ο η συχνότητα εμφάνισης. Ο χρήστης θα δίνει αρχικά το πλήθος των παρατηρήσεων και στη συνέχεια, θα εισάγει μία – μία τις τιμές των παρατηρήσεων, οι οποίες θα εισάγονται στην πρώτη λίστα (η εισαγωγή θα γίνεται στην αρχή της λίστας). Να

γίνεται έλεγχος εγκυρότητας τόσο κατά το διάβασμα του πλήθος των παρατηρήσεων όσο και κατά το διάβασμα της κάθε παρατήρησης. Μετά την εισαγωγή των στοιχείων στη 1η λίστα θα εμφανίζονται τα περιεχόμενα της λίστας, και μετά θα υπολογίζονται με τη βοήθεια της συνάρτησης GetFrequencies οι συχνότητες εμφάνισης της κάθε τιμής και το αντίστοιχο στοιχείο της δεύτερης λίστας θα ενημερώνεται. Στο τέλος, θα εκτυπώνεται η συχνότητα για κάθε κλάση 0-9 (2η λίστα). Θα πρέπει να λυθεί με τις υλοποιήσεις L_ListADT.c & L_ListADT.h (Project_L_ListADT)
Ακολουθεί στιγμιότυπο εκτέλεσης.

```
Enter observations number: 12
Enter an observation in [0,9]: 7
Enter an observation in [0,9]: 7
Enter an observation in [0,9]: 1
Enter an observation in [0,9]: 3
Enter an observation in [0,9]: 6
Enter an observation in [0,9]: 9
Enter an observation in [0,9]: 9
Enter an observation in [0,9]: 0
Enter an observation in [0,9]: 1
Enter an observation in [0,9]: 9
Enter an observation in [0,9]: 6
Enter an observation in [0,9]: 5

1h Lista Parathrhsewn
(5) (6) (9) (1) (0) (9) (9) (6) (3) (1) (7) (7)

2h Lista Syxnothtwn
(9:3) (8:0) (7:2) (6:2) (5:1) (4:0) (3:1) (2:0) (1:2) (0:1)
```

32. Μαθητές 10 γυμνασίων της Ανατολικής Θεσσαλονίκης συμμετείχαν σε διαγωνισμό μαθηματικών. α) Οι μαθητές που πρώτευσαν από κάθε γυμνάσιο καταχωρούνται σε ΑΣΛ της Α' φάσης του διαγωνισμού. Η εισαγωγή γίνεται κάθε φορά στην αρχή της ΑΣΛ, και καταχωρείται το όνομα (20 χαρακτήρες) και το γυμνάσιο (ακέραιος). Τα στοιχεία των 10 μαθητών θα τα διαβάσετε από το αρχείο students.dat, το οποίο σας δίνεται. β) Απ' αυτούς τους 10 κληρώνονται 2 μαθητές και 2 αναπληρωματικοί που θα λάβουν μέρος στη Β' φάση του διαγωνισμού. Τους μαθητές (τις θέσεις των μαθητών στον πίνακα με τους κόμβους της ΑΣΛ) που κληρώνονται τις διαβάζει το πρόγραμμα από το χρήστη, διενεργώντας σχετικό έλεγχο εγκυρότητας του αριθμού. Οι 4 αυτοί μαθητές διαγράφονται από την ΑΣΛ της Α' φάσης του διαγωνισμού και καταχωρούνται σε ΑΣΛ της Β' φάσης του διαγωνισμού (θεωρείστε ίδιο μέγεθος για τις δύο ΑΣΛ). Η εισαγωγή στη 2η ΑΣΛ γίνεται πάντα στο τέλος. γ) 2 μέρες πριν το διαγωνισμό ο 2ος μαθητής αρρώστησε και θα τον αντικαταστήσει ο 1ος αναπληρωματικός. Διαγράψτε από την 2η ΑΣΛ τον μαθητή αυτόν και να το εισάγετε στην 1η ΑΣΛ (εισαγωγή στην αρχή της ΑΣΛ). Μετά τα α), β) γ) εμφανίζεστε τη storage pool και στη συνέχεια τα στοιχεία της ΑΣΛ όπως φαίνεται στο στιγμιότυπο εκτέλεσης. Θα πρέπει να τροποποιήσετε τη λειτουργία της διαγραφής ώστε να επιστρέφει το στοιχείο που διαγράφεται. Θα πρέπει να λυθεί με τις υλοποιήσεις L_ListADT.c & L_ListADT.h (Project_L_ListADT)
Ακολουθεί στιγμιότυπο εκτέλεσης.

```
Question a:
1o STOIXEIO LISTAS=9, 1H FREE POSITION=-1
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<TAKIS,3> ->-1) (1:<MANOS,20> ->0) (2:<ANNA,18> ->1) (3:<HLIAS,31> ->2) (4:<NIKOS,9> ->3)
(5:<ASPA,22> ->4) (6:<MARIA,2> ->5) (7:<TASOS,7> ->6) (8:<ALIKI,1> ->7) (9:<KOSTAS,11> ->8)
Lista A
(9:<KOSTAS,11> ->8) (8:<ALIKI,1> ->7) (7:<TASOS,7> ->6) (6:<MARIA,2> ->5) (5:<ASPA,22> ->4)
(4:<NIKOS,9> ->3) (3:<HLIAS,31> ->2) (2:<ANNA,18> ->1) (1:<MANOS,20> ->0) (0:<TAKIS,3> ->-1)
Dwse thn thesh toy ma8hth poy klhrw8hke:4
Dwse thn thesh toy ma8hth poy klhrw8hke:6
Dwse thn thesh toy ma8hth poy klhrw8hke:0
Dwse thn thesh toy ma8hth poy klhrw8hke:9
Question b:
1o STOIXEIO LISTAS=8, 1H FREE POSITION=9
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<,-1> ->6) (1:<MANOS,20> ->-1) (2:<ANNA,18> ->1) (3:<HLIAS,31> ->2) (4:<,-1> ->-1)
(5:<ASPA,22> ->3) (6:<,-1> ->4) (7:<TASOS,7> ->5) (8:<ALIKI,1> ->7) (9:<,-1> ->0)
Lista A
(8:<ALIKI,1> ->7) (7:<TASOS,7> ->5) (5:<ASPA,22> ->3) (3:<HLIAS,31> ->2) (2:<ANNA,18> ->1)
(1:<MANOS,20> ->-1)
1o STOIXEIO LISTAS=0, 1H FREE POSITION=4
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<NIKOS,9> ->1) (1:<MARIA,2> ->2) (2:<TAKIS,3> ->3) (3:<KOSTAS,11> ->-1) (4:<,-1> ->5)
(5:<,-1> ->6) (6:<,-1> ->7) (7:<,-1> ->8) (8:<,-1> ->9) (9:<,-1> ->-1)
Lista B
(0:<NIKOS,9> ->1) (1:<MARIA,2> ->2) (2:<TAKIS,3> ->3) (3:<KOSTAS,11> ->-1)
Question c:
```

```
1o STOIXEIO LISTAS=9, 1H FREE POSITION=0
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<,-1> ->6) (1:<MANOS,20> ->-1) (2:<ANNA,18> ->1) (3:<HLIAS,31> ->2) (4:<,-1> ->-1)
(5:<ASPA,22> ->3) (6:<,-1> ->4) (7:<TASOS,7> ->5) (8:<ALIKI,1> ->7) (9:<MARIA,2> ->8)
Lista A
(9:<MARIA,2> ->8) (8:<ALIKI,1> ->7) (7:<TASOS,7> ->5) (5:<ASPA,22> ->3) (3:<HLIAS,31> ->2)
(2:<ANNA,18> ->1) (1:<MANOS,20> ->-1)
1o STOIXEIO LISTAS=0, 1H FREE POSITION=1
H STORAGE POOL EXEI TA EJHS STOIXEIA
(0:<NIKOS,9> ->2) (1:<,-1> ->4) (2:<TAKIS,3> ->3) (3:<KOSTAS,11> ->-1) (4:<,-1> ->5) (5:<,-
1> ->6) (6:<,-1> ->7) (7:<,-1> ->8) (8:<,-1> ->9) (9:<,-1> ->-1)
Lista B
(0:<NIKOS,9> ->2) (2:<TAKIS,3> ->3) (3:<KOSTAS,11> ->-1)
```