ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Τμήμα Εφαρμοσμένης Πληροφορικής

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Εξάμηνο Β'

Φύλλο Ασκήσεων 6: ΚΑΤΑΚΕΡΜΑΤΙΣΜΟΣ

Μάγια Σατρατζέμη, Γεωργία Κολωνιάρη, Αλέξανδρος Καρακασίδης

Παρατηρήσεις:

1. *Τα δεδομένα εισόδου διαβάζονται πάντα με ξεχωριστές εντολές scanf( ) το καθένα (εκτός αν ορίζεται διαφορετικά στην άσκηση) 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. **ΠΡΟΣΟΧΗ: Οι ασκήσεις θα πρέπει να λύνονται με χρήση του κώδικα που υλοποιεί τον ΑΤΔ Hashing. Ο κώδικας που σας δίνεται περιλαμβάνεται στο code.zip στην αντίστοιχη διάλεξη. Οι συναρτήσεις που υλοποιούν τις βασικές λειτουργίες του ΑΤΔ Hashing δεν τροποποιούνται. Τροποποιήσεις μπορούν να γίνουν ανάλογα με την άσκηση και εφόσον χρειάζεται στον τύπο του στοιχείου, στην εμφάνιση των στοιχείων της ΔΔ και εφόσον το κλειδί είναι αλφαριθμητικό και όπου γίνονται συγκρίσεις μ' αυτό θα πρέπει να τροποποιηθούν οι εντολές σύγκρισης.**

1. Θεωρήστε ότι έχουμε έναν πίνακα κατακερματισμού με έντεκα θέσεις και τη συνάρτηση κατακερματισμού
h(i) = i % 11
Σχεδιάστε τον πίνακα κατακερματισμού που προκύπτει αν εισαχθούν οι ακόλουθοι ακέραιοι αριθμοί με τη σειρά που δίνονται: 26, 42, 5, 44, 92, 59, 40, 36, 12, 60, 80. Για το χειρισμό των συγκρούσεων χρησιμοποιείται η μέθοδος της αλυσιδωτής σύνδεσης.

2. Θεωρώντας ότι η παρακάτω δομή δεδομένων δημιουργήθηκε με την τεχνική του κατακερματισμού με αλυσίδες συνωνύμων, να προσδιορίσετε ποιο είναι το περιεχόμενο της κάθε υπολίστας συνωνύμων:

HashTable

1	7
2	1
3	4
4	2
5	6

List

1	6	3
2	8	5
3	11	8
4	12	9
5	13	-1
6	9	12
7	10	11
8	21	-1
9	32	10
10	22	-1
11	5	-1
12	24	-1

3. Γράψτε ένα πρόγραμμα για τη δημιουργία και επεξεργασία μιας δομή δεδομένων που δημιουργήθηκε με την τεχνική του κατακερματισμού με αλυσίδες συνωνύμων, στην οποία αποθηκεύονται τα στοιχεία των χρηστών μιας Πανεπιστημιακής βιβλιοθήκης. Κάθε εγγραφή περιλαμβάνει τα εξής στοιχεία της κάρτας που εκδίδεται από τη βιβλιοθήκη και χρησιμοποιούν οι χρήστες για το δανεισμό βιβλίων:

- τον κωδικό του χρήστη (ακέραιος αριθμός – κλειδί κατακερματισμού)
- το ονοματεπώνυμο του χρήστη (*αλφαριθμητικό 20 θέσεων*)
- την ιδιότητα του χρήστη (ακέραιος αριθμός, 1 = postgraduate student, 2 = PhD student, 3 = Professor, 4 = External user)

Η συνάρτηση κατακερματισμού να είναι: h(i) = (i % 10) +1.
Το πρόγραμμα καθοδηγείται από το ακόλουθο μενού επιλογών:

1. *Create HashList* (Δημιουργία της δομής δεδομένων)
2. *Insert new user* (Εισαγωγή νέου χρήστη της βιβλιοθήκης)
3. *Delete a user* (Διαγραφή χρήστη)
4. *Search for a user* (Αναζήτηση χρήστη – αν υπάρχει χρήστης με το συγκεκριμένο κωδικό θα εμφανίζονται τα στοιχεία του, αλλιώς θα εμφανίζεται

το μήνυμα '*There is no user with code x*', όπου *x* ο κωδικός που δόθηκε προς αναζήτηση)

5. *Print list of all users* (Εκτύπωση των στοιχείων όλων των χρηστών – τα στοιχεία κάθε χρήστη θα εμφανίζονται σε διαφορετική γραμμή χωρισμένα με κόμμα)
7. *Quit* (Έξοδος)

Για τις λειτουργίες 2, 3 και 4 ο χρήστης του προγράμματος θα έχει τη δυνατότητα εισαγωγής, διαγραφής ή αναζήτησης αντίστοιχα όσων χρηστών της βιβλιοθήκης επιθυμεί. Η συνέχιση ή όχι κάθε μιας από τις τρεις αυτές λειτουργίες θα ελέγχεται από σχετικό μήνυμα '*Continue Y/N?*'.
Για τις λειτουργίες 4 και 5 θα εμφανίζεται η ιδιότητα του χρήστη (postgraduate student, PhD student, Professor, External user) και όχι ο κωδικός 1, 2, 3 ή 4 που αποθηκεύεται στη δομή.
Για τη λειτουργία 5 η εμφάνιση των στοιχείων ομαδοποιημένα σε συνώνυμα, δηλαδή θα γίνεται ως εξής;
Synonyms, collision at position: <θέση κατακερματισμού>
Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]
Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]
….
Synonyms, collision at position: <θέση κατακερματισμού>
Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]
Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]
…..

Το μήνυμα «Synonyms, collision at position: <θέση κατακερματισμού>» θα εμφανίζεται ακόμη και αν υπάρχει μόνο ένα κλειδί στη θέση <θέση κατακερματισμού>

Σας δίνεται ένα στιγμιότυπο, όπου τα στοιχεία καταχωρήθηκαν με την εξής σειρά: 4, 2, 7, 14, 12, 17 (δίνονται μόνο τα κλειδιά των εγγραφών):
LIBRARY USERS:
Synonyms, collision at position: 3
2: [2, VASILEIOY VASILIS, PhD student]
5: [12, KWNSTANTINOY KWSTAS, PhD student]
Synonyms, collision at position: 5
1: [4, ANASTASIOY ANASTASIS, External user]
4: [14, DHMHTRIOY DIMITRIS, External user]
Synonyms, collision at position: 8
3: [7, NIKOLAOY NIKOS, Professor]
6: [17, LIOLIOY ASPA, postgraduate student]

4. Σε έναν εκπαιδευτικό οργανισμό εργάζονται εκπαιδευτικοί διαφόρων ειδικοτήτων. Τα βασικά τους στοιχεία υπάρχουν σε ένα αρχείου κειμένου 'i4f6.txt' (σε διαφορετικές γραμμές για κάθε εκπαιδευτικό):

Ονομα	*αλφαριθμητικό 10 χαρακτήρες*
Επώνυμο	*αλφαριθμητικό 20 χαρακτήρες*
Τηλέφωνο	*αλφαριθμητικό 10 θέσεων*
Κωδικός ειδικότητας	*byte* (1=Θεολόγοι, 2=Φιλόλογοι, …20=Πληροφορικοί)

Στο κυρίως πρόγραμμα θα υλοποιούνται στη σειρά οι παρακάτω λειτουργίες:

1. BuildHashList Διάβασμα των στοιχείων από το αρχείο κειμένου και δημιουργία Δομής Δεδομένων (ΔΔ) που αποθηκεύει και επεξεργάζεται τα στοιχεία της με την τεχνική του κατακερματισμού με αλυσίδες συνωνύμων. Το κλειδί σχηματίζεται από το όνομα+κενό χαρακτήρα+επώνυμο. Πχ αν το όνομα είναι «nikos» και το επώνυμο «dimitriou», τότε το κλειδί κατακερματισμού που θα σχηματιστεί είναι: «nikos dimitriou»

```
void BuildHashList(HashListType *HList);
```

2. Insert new teacher Εισαγωγή των στοιχείων ενός νέου εκπαιδευτικού στη ΔΔ κατακερματισμού με αλυσίδες συνωνύμων

3. Delete a teacher Διαγραφή ενός εκπαιδευτικού από τη ΔΔ κατακερματισμού με αλυσίδες συνωνύμων

4. Search for a teacher Αναζήτηση και εμφάνιση των στοιχείων ενός εκπαιδευτικού βάσει

ονοματεπωνύμου στη ΔΔ κατακερματισμού με αλυσίδες συνωνύμων

5. Search by subject Αναζήτηση και εμφάνιση των στοιχείων των εκπαιδευτικών μιας συγκεκριμένης ειδικότητας (ο κωδικός της ειδικότητας [1..20] αποτελεί παράμετρο της διαδικασίας) στη ΔΔ κατακερματισμού με αλυσίδες συνωνύμων

```
void Search_HashList_By_Subject(HashListType HList, int code);
```

Μετά τις λειτουργίες 1, 2, 3 θα καλείτε την PrintPinakes(HList).

Ο πίνακας κατακερματισμού θα έχει 9 θέσεις και ως συνάρτηση κατακερματισμού θα χρησιμοποιηθεί η εξής:

h(i) = average % 9

όπου

average = (κωδικός_πρώτου_ χαρακτήρα + κωδικός_τελευταίου_χαρακτήρα) / 2

Θεωρήστε ότι χρησιμοποιούνται οι ακόλουθοι κωδικοί για τους χαρακτήρες: 'A' = 1, 'B' = 2, …, 'Z' = 26. Ο πρώτος και ο τελευταίος χαρακτήρας του ονοματεπωνύμου θα μετατρέπεται στον αντίστοιχο κεφαλαίο χαρακτήρα, εφόσον είναι πεζός. Ο μέσος όρος *average* θα υπολογίζεται με μια συνάρτηση *findAverage*, η οποία θα καλείται από τη συνάρτηση *HashKey*.

Δίνεται ένα στιγμιότυπο εκτέλεσης

```
1. Create HashList
Hash table
-1, -1, 1, -1, 5, -1, -1, 6, 0,
Hash List
0) nikos dimitriou, 2
1) maya satratzemi, 4
2) marios nikolaou, 3
3) maria giannou, -1
4) dimitra totsika, -1
5) giannis pappas, -1
6) nikos ploskas, -1

2. Insert new teacher
Enter teacher's name: nikos
Enter teacher's surname: nikolaou
Enter teacher's phone: 888888888
Enter teacher code: 1

Continue Y/N: y
Enter teacher's name: dimis
Enter teacher's surname: tolis
Enter teacher's phone: 6977777777
Enter teacher code: 1

Continue Y/N: n
Hash table
-1, -1, 1, -1, 5, -1, -1, 6, 0,
Hash List
0) nikos dimitriou, 2
1) maya satratzemi, 4
2) marios nikolaou, 3
3) maria giannou, 7
4) dimitra totsika, 8
5) giannis pappas, -1
6) nikos ploskas, -1
7) nikos nikolaou, -1
8) dimis tolis, -1
```

Συνεχίζεται στη δεύτερη στήλη

```
3. Delete a teacher
Enter teacher's name: giannis
Enter teacher's surname: papas
DEN YPARXEI  EGGRAFH ME KLEIDI giannis papas
Hash table
-1, -1, 1, -1, 5, -1, -1, 6, 0,
Hash List
0) nikos dimitriou, 2
1) maya satratzemi, 4
2) marios nikolaou, 3
3) maria giannou, 7
4) dimitra totsika, 8
5) giannis pappas, -1
6) nikos ploskas, -1
7) nikos nikolaou, -1
8) dimis tolis, -1

4. Search for a teacher
Enter teacher's name: nikos
Enter teacher's surname: dimitriou
[nikos dimitriou, 6911111112, 1]

5. Search by subject
Enter code: 1
List of teachers with subject code 1:
8: [dimis tolis, 6977777777, 1]
0: [nikos dimitriou, 6911111112, 1]
3: [maria giannou, 6914441112, 1]
7: [nikos nikolaou, 888888888, 1]
```

5. Σε ένα υπολογιστικό σύστημα είναι επιτρεπτή η πρόσβαση (login) μόνον αν κάθε χρήστης διαθέτει αντίστοιχο λογαριασμό που αποτελείται από:

- μια ταυτότητα χρήστη *user_id* (*αλφαριθμητικό 8 θέσεων*), η οποία θα αποτελεί και το κλειδί κατακερματισμού και
- έναν κωδικό πρόσβασης *password* (*αλφαριθμητικό 6 θέσεων)*.

Τα *user_id* και τα *password* των χρηστών είναι αποθηκεμένα στο αρχείο κειμένου "I5F6.txt" (σε διαφορετικές γραμμές).

Να γίνει πρόγραμμα που:

- θα διαβάζει τα περιεχόμενα του αρχείου κειμένου και θα δημιουργεί μία ΔΔ που αποθηκεύει και επεξεργάζεται τα στοιχεία της με την τεχνική του κατακερματισμού με αλυσίδες συνωνύμων
- θα εμφανίζει το μηνύματα '*USERNAME: '* και '*PASSWORD:*', και θα δέχεται από το πληκτρολόγιο την ταυτότητα χρήστη και τον κωδικό πρόσβασης αντίστοιχα. Η εισαγωγή των στοιχείων των χρηστών θα ελέγχεται από το μήνυμα '*New entry Y/N (Y=Yes, N=No)?*'.
- Για κάθε χρήστη θα ελέγχεται αν τα στοιχεία του λογαριασμού είναι σωστά και θα εμφανίζονται τα κατάλληλα μηνύματα:
    i) Αν υπάρχει στη δομή **ΔΔ** λογαριασμός με τα στοιχεία που έδωσε ο χρήστης θα εμφανίζεται το μήνυμα '*You have logged in to the system*'
    ii) Αν η ταυτότητα χρήστη δεν είναι σωστή θα εμφανίζει το μήνυμα '*Access is forbidden: Wrong user ID*'.

iii) Αν η ταυτότητα χρήστη είναι σωστή, αλλά όχι και ο κωδικός πρόσβασης τότε θα εμφανίζει το μήνυμα'*Access is forbidden: Wrong password*'.

Ο πίνακας κατακερματισμού θα έχει 10 θέσεις και ως συνάρτηση κατακερματισμού θα χρησιμοποιηθεί η εξής:

h(i) = average % 10

όπου

average = (κωδικός_ASCII_πρώτου_ χαρακτήρα + κωδικός_ASCII_τελευταίου_χαρακτήρα) / 2

Ο μέσος όρος *average* θα υπολογίζεται με μια συνάρτηση *findAverage*, η οποία θα καλείται από τη συνάρτηση *HashKey*. Η υλοποίηση του ΑΤΔ θα πρέπει να τροποποιηθεί κατάλληλα, ώστε το κλειδί να είναι ένα αλφαριθμητικό 8 χαρακτήρων.

6. Μια συνάρτηση κατακερματισμού θεωρείται καλή όταν αποτιμάται εύκολα και κατανέμει τα στοιχεία σε όλο το εύρος του πίνακα κατακερματισμού, ελαχιστοποιώντας έτσι την πιθανότητα συγκρούσεων. Αν και δεν έχει εντοπιστεί μια μέθοδος κατακερματισμού που να λειτουργεί τέλεια σε όλες τις καταστάσεις, μία δημοφιλής μέθοδος, γνωστή ως *τυχαίος κατακερματισμός*, χρησιμοποιεί μια απλή τεχνική δημιουργίας αριθμών για την 'τυχαία' κατανομή των στοιχείων στον πίνακα κατακερματισμού. Το στοιχείο που αποτελεί το κλειδί *key* μετατρέπεται πρώτα σε ένα μεγάλο τυχαίο ακέραιο χρησιμοποιώντας μια εντολή της μορφής

*RandomInt = ((Multiplier * key) + Addend) % Modulus*

και έπειτα υπολογίζεται η θέση στον πίνακα κατακερματισμού με τη γνωστή συνάρτηση

Location = (RandomInt % HMax) + 1

Κατάλληλες τιμές για τις σταθερές *Multiplier*, *Addend*, και *Modulus* είναι οι εξής:

Modulus = 65536
Multiplier = 25173
Addend = 13849

Η σταθερά *Modulus* έχει την τιμή 65536 εφόσον ο υπολογιστής έχει λέξεις των 32 bit. Για έναν υπολογιστή με λέξεις των *M*-bit, πρέπει να αντικατασταθεί με την τιμή του $2^{M/2}$.

- Προκειμένου το πρόγραμμα σας να λειτουργεί σε έναν οποιοδήποτε υπολογιστή, κατά την έναρξη της εκτέλεσης θα ζητάτε από τον χρήστη τον αριθμό των bits της λέξης του υπολογιστή και θα υπολογίζετε την τιμή της (μεταβλητής του κυρίως προγράμματος) *Modulus*.
- Στη συνέχεια, χρησιμοποιώντας την παραπάνω συνάρτηση κατακερματισμού δημιουργήστε μια δομή ΔΔ που αποθηκεύει και επεξεργάζεται τα στοιχεία της με την τεχνική του κατακερματισμού με αλυσίδες συνωνύμων με ακέραια κλειδιά. Για λόγους απλότητας αποθηκεύστε στη δομή ακέραια κλειδιά με τιμές [1..100]. Θεωρήστε ότι ο πίνακας κατακερματισμού έχει μέγεθος 10 και η λίστα αποθήκευσης των κλειδιών 100. Τέλος, εμφανίστε τα περιεχόμενα του πίνακα κατακερματισμού και των υπολιστών συνωνύμων. Η εμφάνιση θα γίνεται όπως στην άσκηση 3, λειτουργία 5.

Παρατήρηση: αυτή η συνάρτηση κατακερματισμού μπορεί να χρησιμοποιηθεί και με κλειδιά που δεν είναι ακέραιοι αριθμοί, αν πρώτα τα κωδικοποιήσουμε ως ακέραιους. Για παράδειγμα, ένα όνομα μπορεί να κωδικοποιηθεί ως το άθροισμα των κωδικών ASCII ορισμένων ή όλων των χαρακτήρων του.

7. Γράψτε ένα πρόγραμμα για τη δημιουργία και επεξεργασία μιας ΔΔ που αποθηκεύει και επεξεργάζεται τα στοιχεία της με την τεχνική του κατακερματισμού με αλυσίδες συνωνύμων, στην οποία αποθηκεύονται τα στοιχεία των μελών ενός γυμναστηρίου. Κάθε εγγραφή περιλαμβάνει τα εξής στοιχεία της κάρτας μέλους που δίνεται σε κάθε μέλος όταν εγγράφεται στο γυμναστήριο:

- τον κωδικό (ακέραιος αριθμός – κλειδί κατακερματισμού)
- το όνομα μέλους (username) (*αλφαριθμητικό 20 θέσεων*)
- το ποσό οφειλής του μέλους στο γυμναστήριο (*ακέραιος*)

Η συνάρτηση κατακερματισμού να είναι: h(i) = i % 5.

Στο κυρίως πρόγραμμα θα υλοποιούνται στη σειρά οι παρακάτω λειτουργίες:

1. Create HashList	Δημιουργία της δομής δεδομένων
2. Insert new member	Εισαγωγή νέου μέλους
3. Search for a member	Αναζήτηση μέλους – αν υπάρχει μέλος με το συγκεκριμένο κωδικό θα εμφανίζονται τα στοιχεία του, αλλιώς θα εμφανίζεται το μήνυμα '*DEN YPARXEI EGGRAFH ME KLEIDI x*', όπου *x* ο κωδικός που δόθηκε προς αναζήτηση
4. Update member amount	Ενημέρωση της οφειλής του μέλους – ο χρήστης δίνει τον κωδικό του μέλους και το ποσό και ενημερώνεται κατάλληλα το ποσό της οφειλής. Κατ' αρχήν θα γίνεται έλεγχος αν υπάρχει μέλος με το συγκεκριμένο κωδικό, αν δεν υπάρχει θα εμφανίζεται το μήνυμα '*DEN YPARXEI EGGRAFH ME KLEIDI x*', όπου *x* ο κωδικός που δόθηκε. Αν ο κωδικός υπάρχει θα διαβάζεται το ποσό που θα πληρώσει. Θα γίνεται έλεγχος ώστε το ποσό που δίνεται να είναι μικρότερο ή ίσο της καταχωρημένης οφειλής.

5. Delete a member — Διαγραφή μέλους – η διαγραφή δεν μπορεί να πραγματοποιηθεί αν το ποσό οφειλής του μέλους δεν έχει εξοφληθεί. Σε αυτή την περίπτωση η διαγραφή δεν πραγματοποιείται και εμφανίζεται το μήνυμα '*Not deleted arrange amount'*. Κατ' αρχήν θα γίνεται έλεγχος αν υπάρχει μέλος με το συγκεκριμένο κωδικό, αν δεν υπάρχει θα εμφανίζεται το μήνυμα '*DEN YPARXEI EGGRAFH ME KLEIDI x*', όπου *x* ο κωδικός που δόθηκε.

6. Print list of synonyms — Ο χρήστης δίνει τον κωδικό ενός μέλους του γυμναστηρίου και εμφανίζονται τα περιεχόμενα της υπολίστας των συνωνύμων στην οποία ανήκει. Θα γίνεται έλεγχος αν υπάρχει μέλος με το συγκεκριμένο κωδικό, αν δεν υπάρχει θα εμφανίζεται το μήνυμα '*DEN YPARXEI EGGRAFH ME KLEIDI x*', όπου *x* ο κωδικός που δόθηκε.
Συνάρτηση `void PrintListOfSynonyms(HashListType HList, int key);`

Για τις λειτουργίες 2 έως και 6 ο χρήστης του προγράμματος θα έχει τη δυνατότητα εισαγωγής, αναζήτησης, ενημέρωσης, διαγραφής και εμφάνισης της λίστας συνωνύμων για όσα μέλη του γυμναστηρίου επιθυμεί μέσω σχετικού μηνύματος '*Continue Y/N?*' Μετά τις λειτουργίες 1, 2, 4, 5 θα καλείτε τις Print_HashList(HList) και PrintPinakes(HList). Στην Print_HashList(HList) δε θα εμφανίζετε τη λίστα με τις ελεύθερες θέσεις της δομής.
Δίνεται ένα στιγμιότυπο εκτέλεσης

```
1. Create HashList
HASHLIST STRUCTURE with SYNONYM CHAINING
=======================================
PINAKAS DEIKTWN STIS YPO-LISTES SYNWNYMWN EGGRAFWN:
0| -1
1| -1
2| -1
3| -1
4| -1
OI YPO-LISTES TWN SYNWNYMWN EGGRAFWN:
TELOS  0HS YPO-LISTAS
TELOS  1HS YPO-LISTAS
TELOS  2HS YPO-LISTAS
TELOS  3HS YPO-LISTAS
TELOS  4HS YPO-LISTAS
MEGE8OS THS LISTAS = 0
=======================================
Hash table
0| -1
1| -1
2| -1
3| -1
4| -1
Hash List

2. Insert new member
Give code: 12
Give name: DWDEKAS
Give amount: 0

Continue Y/N: Y
Give code: 1
Give name: ENAS
Give amount: 100

Continue Y/N: Y
Give code: 2
Give name: DUOS
Give amount: 100

Continue Y/N: Y
Give code: 11
Give name: ENTEKAS
Give amount: 100

Continue Y/N: Y
Give code: 3
Give name: TRIOS
Give amount: 100

Continue Y/N: Y
Give code: 22
Give name: EIKOSIDIOS
```

```
3. Search for a member
Give code: 11
[11, ENTEKAS, 100, -1]

Continue Y/N: N
4. Update member amount
Give code: 1
[1, ENAS, 100, 3]
Give amount: 100

Continue Y/N: N
HASHLIST STRUCTURE with SYNONYM CHAINING
=======================================
PINAKAS DEIKTWN STIS YPO-LISTES SYNWNYMWN EGGRAFWN:
0| -1
1| 1
2| 0
3| 4
4| -1
OI YPO-LISTES TWN SYNWNYMWN EGGRAFWN:
TELOS  0HS YPO-LISTAS
[1, ENAS, 0, 3] -> [11, ENTEKAS, 100, -1] -> TELOS  1HS YPO-LISTAS
[12, DWDEKAS, 0, 2] -> [2, DUOS, 100, 5] -> [22, EIKOSIDIOS, 100, -1] ->
TELOS  2HS YPO-LISTAS
[3, TRIOS, 100, -1] -> TELOS  3HS YPO-LISTAS
TELOS  4HS YPO-LISTAS
MEGE8OS THS LISTAS = 6
=======================================
Hash table
0| -1
1| 1
2| 0
3| 4
4| -1
Hash List
0) 12, DWDEKAS, 0, 2
1) 1, ENAS, 0, 3
2) 2, DUOS, 100, 5
3) 11, ENTEKAS, 100, -1
4) 3, TRIOS, 100, -1
5) 22, EIKOSIDIOS, 100, -1

5. Delete a member
Give code: 3
Not deleted arrange amount

Continue Y/N: Y
Give code: 1

Continue Y/N: N
HASHLIST STRUCTURE with SYNONYM CHAINING
=======================================
PINAKAS DEIKTWN STIS YPO-LISTES SYNWNYMWN EGGRAFWN:
```

```
Give amount: 100

Continue Y/N: N
HASHLIST STRUCTURE with SYNONYM CHAINING
=======================================
PINAKAS DEIKTWN STIS YPO-LISTES SYNWNYMWN EGGRAFWN:
0| -1
1| 1
2| 0
3| 4
4| -1
OI YPO-LISTES TWN SYNWNYMWN EGGRAFWN:
TELOS  0HS YPO-LISTAS
[1, ENAS, 100, 3] -> [11, ENTEKAS, 100, -1] -> TELOS  1HS YPO-LISTAS
[12, DWDEKAS, 0, 2] -> [2, DUOS, 100, 5] -> [22, EIKOSIDIOS, 100, -1] ->
TELOS  2HS YPO-LISTAS
[3, TRIOS, 100, -1] -> TELOS  3HS YPO-LISTAS
TELOS  4HS YPO-LISTAS
MEGE8OS THS LISTAS = 6
=======================================
Hash table
0| -1
1| 1
2| 0
3| 4
4| -1
Hash List
0) 12, DWDEKAS, 0, 2
1) 1, ENAS, 100, 3
2) 2, DUOS, 100, 5
3) 11, ENTEKAS, 100, -1
4) 3, TRIOS, 100, -1
5) 22, EIKOSIDIOS, 100, -1
```

Συνεχίζεται στη δεύτερη στήλη

```
0| -1
1| 3
2| 0
3| 4
4| -1
OI YPO-LISTES TWN SYNWNYMWN EGGRAFWN:
TELOS  0HS YPO-LISTAS
[11, ENTEKAS, 100, -1] -> TELOS  1HS YPO-LISTAS
[12, DWDEKAS, 0, 2] -> [2, DUOS, 100, 5] -> [22, EIKOSIDIOS, 100, -1] ->
TELOS  2HS YPO-LISTAS
[3, TRIOS, 100, -1] -> TELOS  3HS YPO-LISTAS
TELOS  4HS YPO-LISTAS
MEGE8OS THS LISTAS = 5
=======================================
Hash table
0| -1
1| 3
2| 0
3| 4
4| -1
Hash List
0) 12, DWDEKAS, 0, 2
1) 1, ENAS, 0, 6
2) 2, DUOS, 100, 5
3) 11, ENTEKAS, 100, -1
4) 3, TRIOS, 100, -1

6. Print list of synonyms
Give code: 12
[12, DWDEKAS, 0, 2]
0: [12, DWDEKAS, 0]
2: [2, DUOS, 100]
5: [22, EIKOSIDIOS, 100]

Continue Y/N: N
```

8. Γράψτε ένα πρόγραμμα για τη δημιουργία και επεξεργασία μιας δομή δεδομένων που δημιουργήθηκε με την τεχνική του κατακερματισμού με αλυσίδες συνωνύμων, στην οποία αποθηκεύονται τα στοιχεία των ατόμων που σιτίζονται στο εστιατόριο του πανεπιστημίου. Κάθε εγγραφή περιλαμβάνει τα εξής στοιχεία της κάρτας που εκδίδεται από το εστιατόριο και χρησιμοποιούν οι χρήστες για τη σίτισή τους::

- τον κωδικό του χρήστη (ακέραιος αριθμός – κλειδί κατακερματισμού)
- το ονοματεπώνυμο του χρήστη (*αλφαριθμητικό 20 θέσεων*)
- την ιδιότητα του χρήστη (ακέραιος αριθμός, 1 = student, 2 = postgraduate student, 3 = teacher, 4 = visitor)

Η συνάρτηση κατακερματισμού να είναι: h(i) = (i % 10).
Το πρόγραμμα καθοδηγείται από το ακόλουθο μενού επιλογών:

1.	*Create HashList*	(Δημιουργία της δομής δεδομένων)
2.	*Insert new user*	(Εισαγωγή νέου χρήστη στο εστιατόριο)
3.	*Delete a user*	(Διαγραφή χρήστη)
4.	*Search for a user*	(Αναζήτηση χρήστη – αν υπάρχει χρήστης με το συγκεκριμένο κωδικό θα εμφανίζονται τα στοιχεία του, αλλιώς θα εμφανίζεται το μήνυμα '*There is no user with code x*', όπου *x* ο κωδικός που δόθηκε προς αναζήτηση)
5.	*Print list of all users*	(Εκτύπωση των στοιχείων όλων των χρηστών – τα στοιχεία κάθε χρήστη θα εμφανίζονται σε διαφορετική γραμμή χωρισμένα με κόμμα)
7.	*Quit*	(Έξοδος)

Για τις λειτουργίες 2, 3 και 4 ο χρήστης του προγράμματος θα έχει τη δυνατότητα εισαγωγής, διαγραφής ή αναζήτησης αντίστοιχα όσων χρηστών του εστιατορίου επιθυμεί. Η συνέχιση ή όχι κάθε μιας από τις τρεις αυτές λειτουργίες θα ελέγχεται από σχετικό μήνυμα '*Continue Y/N?*'.
Για τις λειτουργίες 4 και 5 θα εμφανίζεται η ιδιότητα του χρήστη (προπτυχιακός φοιτητής, Μεταπτυχιακός φοιτητής, Διδάσκων, Επισκέπτης) και όχι ο κωδικός 1, 2, 3 ή 4 που αποθηκεύεται στη δομή.
Για τη λειτουργία 5 η εμφάνιση των στοιχείων ομαδοποιημένα σε συνώνυμα, δηλαδή θα γίνεται ως εξής;
Synonyms, collision at :position <θέση κατακερματισμού>
Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]

Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]
….
Synonyms, collision at :position <θέση κατακερματισμού>
Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]
Θέση στη λίστα εγγραφών: [<κλειδί>, <ονοματεπώνυμο>, <ιδιότητα>]
…..
Σας δίνεται ένα στιγμιότυπο, όπου τα στοιχεία καταχωρήθηκαν με την εξής σειρά: 2, 4, 7, 12, 14, 17 (δίνονται μόνο τα κλειδιά των εγγραφών):
USERS:
Synonyms, collision at position: 2
0: [2, ANDREOY, student]
3: [12, KWSTAKIS, teacher]
Synonyms, collision at position: 4
1: [4, NINOY, postgraduate student]
4: [14, LEONIDOY, visitor]
Synonyms, collision at position: 7
2: [7, APOSTOLOY, postgraduate student]
5: [17, TOTHS, postgraduate student]

9. Σε ένα οδοντιατρείο, διατηρείται ένας κατάλογος με τις επισκέψεις των ασθενών για κάθε ημέρα, όπου κάθε εγγραφή έχει τα εξής στοιχεία:
Όνομα (Αλφαριθμητικό 20 θέσεων, αποθηκεύεται η τιμή κατακερματισμού)
Είδος υπηρεσίας (int: 1-Λεύκανση, 2-Καθαρισμός,3-Εξαγωγή)
Ποσό πληρωμής(double)
Προκειμένου να προστατεύονται τα ιδιωτικά δεδομένα των πελατών, το όνομα αποθηκεύεται ως ένας αριθμός. Γράψτε ένα πρόγραμμα το οποίο διαχειρίζεται τη βάση δεδομένων του οδοντιατρείου, και το οποίο διαθέτει μενού για:

1. Δημιουργία της βάσης.
2. Εισαγωγή εγγραφής επίσκεψης.
3. Εμφάνιση επίσκεψης πελάτη, διαβάζοντας το όνομά του.
4. Έξοδο

Η συνάρτηση κατακερματισμού υπολογίζεται ως

$$H(x)=x\%p,\ p=1543$$

Όπου x είναι το άθροισμα των γραμμάτων του ονόματος, κωδικοποιημένα ως

$$A\rightarrow 1, B\rightarrow 2 \ldots Z\rightarrow 26$$

Και πολλαπλασιασμένα με τη θέση τους στο string. Π.Χ. το αλφαριθμητικό NICK έχει ως άθροισμα το
$1\cdot 14+2\cdot 9+3\cdot 3+4\cdot 11$
ΠΡΟΣΟΧΗ Το όνομα δεν αποθηκεύεται ως αλφαριθμητικό, αλλά ως το άθροισμα των γραμμάτων του!

```
            MENOY
-------------------------------------------------
1. CREATE DATABASE
2. INSERT APPOINTMENT
3. PRINT CLIENT'S APPOINTMENTS
4. EXIT

CHOICE: 1
            MENOY
-------------------------------------------------
1. CREATE DATABASE
2. INSERT APPOINTMENT
3. PRINT CLIENT'S APPOINTMENTS
4. EXIT

CHOICE: 2
Enter the client's Name:GEORGE
Enter the service:
1-Whitening
2-Cleaning
3-Extraction
1
Enter the amount paid:50

Continue Y/N:Y
Enter the client's Name:NICK
Enter the service:
1-Whitening
2-Cleaning
3-Extraction
2
Enter the amount paid:20.50
```

```
Enter the client's Name:GREGEO
Enter the service:
1-Whitening
2-Cleaning
3-Extraction
3
Enter the amount paid:100

Continue Y/N:N
            MENOY
-------------------------------------------------
1. CREATE DATABASE
2. INSERT APPOINTMENT
3. PRINT CLIENT'S APPOINTMENTS
4. EXIT

CHOICE: 3
Enter the client's Name:GEORGE
Service: 1
Amount Paid: 50.00
            MENOY
-------------------------------------------------
1. CREATE DATABASE
2. INSERT APPOINTMENT
3. PRINT CLIENT'S APPOINTMENTS
4. EXIT

CHOICE: 3
Enter the client's Name:GREGEO
Service: 3
Amount Paid: 100.00
            MENOY
-------------------------------------------------
```

Continue Y/N:Y	1. CREATE DATABASE 2. INSERT APPOINTMENT 3. PRINT CLIENT'S APPOINTMENTS 4. EXIT CHOICE:

10. Προσομοιώστε τη λειτουργία μίας απλής γλώσσας προγραμματισμού η οποία διαθέτει τις εξής δυνατότητες, οι οποίες καλούνται μέσω μενού:

1. Δήλωση μεταβλητής μήκους το πολύ 5 χαρακτήρων και ανάθεση τιμής στη μεταβλητή.
2. Πρόσθεση δύο μεταβλητών
3. Πολλαπλασιασμός δύο μεταβλητών.

Η υλοποίηση να γίνει χρησιμοποιώντας λίστα κατακερματισμού με συνώνυμα και η συνάρτηση κατακερματισμού να είναι η εξής:

$$H(x)=x\%p,\ p=31$$

Όπου x είναι το άθροισμα των γραμμάτων του ονόματος της μεταβλητής, κωδικοποιημένα ως

$$A\rightarrow 1, B\rightarrow 2 \ldots Z\rightarrow 26$$

Και πολλαπλασιασμένα με τη θέση τους στο string. Π.Χ. το αλφαριθμητικό NICK έχει ως άθροισμα το $1\cdot 14+2\cdot 9+3\cdot 3+4\cdot 11$

```
                MENOY
-------------------------------------------------
1. DECLARE VARIABLES
2. MULTIPLY VARIABLES
3. ADD VARIABLES
4. EXIT

CHOICE: 1
Enter the variable name: A
Enter variable value: 20

Continue Y/N: Y
Enter the variable name: B
Enter variable value: 15

Continue Y/N: N
                MENOY
-------------------------------------------------
1. DECLARE VARIABLES
2. MULTIPLY VARIABLES
3. ADD VARIABLES
4. EXIT

CHOICE: 2
Enter the first variable: A
Enter the second variable: B
A*B=300
                MENOY
-------------------------------------------------
1. DECLARE VARIABLES
2. MULTIPLY VARIABLES
3. ADD VARIABLES
4. EXIT

CHOICE: 3
Enter the first variable: A
Enter the second variable: B
A+B=35
                MENOY
-------------------------------------------------
1. DECLARE VARIABLES
2. MULTIPLY VARIABLES
3. ADD VARIABLES
4. EXIT
```