3. Εργασία αποθήκευσης & αναζήτησης δεδομένων με χρήση HASHING
Περιγραφή:
Ζητείται η δημιουργία της δομής “hash table” σε ένα πρόβλημα μεγάλου πλήθους κλειδιών.

Ακολουθούμε τα παρακάτω βήματα:
Δημιουργία ενός εκατομμυρίου (1.000.000) επισκέψεων σε πολυκατάστημα και πληρωμή με Πιστωτική Κάρτα.
Από το πολύ μεγάλο πλήθος των διαφορετικών καρτών, δημιουργείται ένα μικρό σχετικά υποσύνολο ως εξής.
Οι πιστωτικές κάρτες των επισκέψεων θα έχουν δεκαέξι (16) συγκεκριμένα σταθερά ψηφία πχ 1234567890123456, αλλά σε τέσσερις (4) από τις δεκαέξι (16) τυχαίες θέσεις θα έχουν και τους τέσσερις χαρακτήρες: A, B, C, D με τυχαία σειρά και σε τυχαία θέση στον αριθμό της κάρτας (πχ 12A45C789012B4D6).
Στις υπόλοιπες θέσεις παραμένουν οι αρχικές τιμές.

Τα ποσά των πληρωμών είναι τυχαίοι ακέραιοι αριθμοί από το 20 μέχρι το 100.
Επίσης επιλέγουμε τυχαία μια ημέρα για τις επισκέψεις μας από Δευτέρα έως και Σάββατο.
Τα δεδομένα των επισκέψεων δεν είναι διαθέσιμα εξ αρχής, αλλά δημιουργούνται ένα-ένα και χρησιμοποιούνται μόνο για την ενημέρωση του HashTable.
Στον πίνακα HashTable μεγέθους N, αποθηκεύονται αναφορές στην δομή της κάρτας που περιέχει τις απαραίτητες για την άσκηση πληροφορίες.

Αρχίζουμε με ένα πρώτο αριθμό N (αρχικό μέγεθος πίνακα) περίπου 1.000.
Καθώς προσθέτουμε νέα στοιχεία το (load factor = n/N), όπου n είναι ο αριθμός των καρτών που βλέπουμε, μεγαλώνει και πρέπει να διπλασιάσουμε τον πίνακα όταν το load factor φτάσει την τιμή 0.6.
Πρέπει πάλι και μετά τον διπλασιασμό το N να παραμείνει πρώτος αριθμός!

Χρησιμοποιούμε μόνο τον πίνακα HashTable[N] (μεθοδολογία open addressing) για την αποθήκευση των καρτών και επιλέγουμε μια τεχνική (δική σας επιλογή) για την τακτοποίηση των συγκρούσεων (collisions)!

Η δημιουργία των τυχαίων αριθμών στις επισκέψεις να γίνει με seed τον αριθμό μητρώου σας.

Μετά την αποθήκευση των δεδομένων των επισκέψεων ανά κάρτα ζητείται να προσδιοριστούν :
1. η κάρτα με το μικρότερο συνολικό ποσό πληρωμών
2. η κάρτα με το μεγαλύτερο πλήθος επισκέψεων
3. η ημέρα με το μικρότερο πλήθος επισκέψεων
4. το πλήθος των συγκρούσεων στον τελικό πίνακα (αφού διαχειριστούμε όλες τις επισκέψεις πελατών)

Προσοχή, οι συγκρούσεις αναφέρονται σε κάρτες που εμφανίζονται για πρώτη φορά, αφού τις επόμενες φορές οι συγκρούσεις είναι αναπόφευκτες.
Επίσης οι συγκρούσεις αναφέρονται στην πρώτη προσπάθεια τοποθέτησης των δεδομένων μιας κάρτας στο HashTable.

Επίσης προσδιορίστε τα εξής:
• Συνολικός χρόνος εκτέλεσης για τη δημιουργία του HashTable και ανάλογη σύγκριση όταν έχουμε 2.000.000 επισκέψεις αντί για 1.000.000 επισκέψεις (χωρίς το χρόνο για την απάντηση στα 4 παραπάνω ερωτήματα)
• Υπολογίστε το πλήθος των συγκρούσεων στον τελικό πίνακα εκτός από την τιμή 0.6 και για τις τιμές 0.7 & 0.8 του load factor!

Επεξήγηση για τους αριθμούς των καρτών:
Για αρχικά σταθερά 16 ψηφία : 1234567890123456, πιθανές κάρτες:
123A56B8901C34D6, 12B45678CD12345A, 1234C67A9DB23456, D23A567B90123C56, …

Παραδίδετε συμπιεσμένο αρχείο με το username σας (π.χ up123456.zip) που περιέχει τα αρχεία του κώδικα (σε source file) και την αναφορά σε pdf ή txt file (παρακαλώ όχι doc), που περιγράφει τη δομή και τους αλγόριθμους που χρησιμοποιήθηκαν για την αποθήκευση των δεδομένων και για την απάντηση των ερωτήσεων.
Ο κώδικας που θα παραδώσετε πρέπει να περιέχει τα απαραίτητα για την καταννόησή του σχόλια και πριν από κάθε συνάρτηση (ή μέθοδο) να υπάρχει μια σύντομη περιγραφή της λειτουργίας της.