TEMA 1 STRUCTURI DE DATE

Dicționar pentru ”autocorrect”

Alexandra-Ștefania Ghiță, Oana Bălan

Orice sistem de operare mobil modern beneficiază de un dicționar de cuvinte integrat ce permite utilizatorului scrierea rapidă de texte, mesaje și email-uri, prin funcția de ”autocorrect”. Asfel, ca urmarea a introducerii unei forme greșite ale unui cuvânt, se generează opțiuni de corectare, utilizatorul putând selecta varianta corectă din punct de vedere gramatical și sintactic. De asemenea, atunci când utilizatorul scrie un cuvânt nou, acesta se salvează în dicționar pentru a putea fi folosit și dat ca sugestie la editări următoare de text.

În vederea implementării unei aplicații de ”autocorrect”, dorim să integrăm ca funcționalități determinarea cuvintelor de căutare cu frecvență maximă și lista de sugestii ale unui anumit cuvânt la căutare. Atunci când utilizatorul introduce un cuvânt nou, el va fi stocat de către aplicație și i se va atribui frecvența de apariție egală cu 1. Pentru aceasta, vom utiliza o structură de date de tip dicționar ce reține în memorie doar cele mai recente înregistrări, împărțite după cheia de căutare.

Structura necesară pentru implementarea temei este alcatuită dintr-un vector cu N elemente, fiecare element fiind reprezentat printr-o listă dublu înlănțuită circulară. Elementele listei sunt tupluri de forma (Key, Value, Frequency), unde Key și Value sunt șiruri de caractere alocate dinamic, iar Frequency este un număr întreg. Vom folosi liste dublu înlănțuite circulare pentru a eficientiza accesarea elementelor și operațiile de adăugare și ștergere. În memorie se va reține primul element al fiecărei liste, iar accesarea ultimului element se va face prin pointerul ”prev” al primului.

Exemple de tupluri pot fi:

(”mmaa”, ”mama”, 5) -> Cheia ”mmaa” a fost corectată cu valoarea ”mama” și are frecvența de apariție 5
(”fcultate”, ”facultate”, 9) -> Cheia ”fcultate” a fost corectată cu valoarea ”facultate” și are frecvența de apariție 9

Atunci când vrem să inserăm un nou tuplu în dicționar, vom calcula un număr întreg r (0<=r<N), care reprezintă numărul listei asociate tuplului și care este restul împărțirii sumei tuturor caracterelor din Key la N. Fiecare listă va fi sortată descrescător după Frequency și pentru elementele cu aceeași valoare a lui Frequency, alfabetic, în funcție de Value și apoi în funcție de Key.

CERINȚE

Operațiile ce vor fi implementate sunt următoarele:

addElement (Dictionary, Key, Value, Frequency)

- Adaugă un nou tuplu în lista corespunzătoare, asigurându-se faptul că lista rămâne ordonată conform criteriilor enunțate mai sus.
- Dacă se încearcă adăugarea unui element cu aceleași Key și Value ca unul existent deja în dicționar, se va aduna la parametrul Frequency al tuplului deja existent valoarea Frequency a tuplului care se încearcă a fi adăugat.
- Dacă se încearcă adăugarea unui element cu un Key existent în dicționar, dar cu un Value inexistent, acesta va fi adăugat ca element nou al listei.
- Pentru un răspuns rapid din partea aplicației, vom limita numărul de elemente reținute de dicționar. Astfel că, atunci când se adaugă un nou tuplu, dacă lista corespunzătoare cheii conține deja N elemente, se va șterge ultimul element din lista respectivă înainte de adăugare.
- Tot din motive de eficientizare, vom limita memoria dicționarului la 2*N elemente. În cazul în care numărul total de elemente din toate listele este mai mare decât 2*N, se va șterge ultimul element al fiecărei liste. Astfel, va dispărea căutarea cea mai puțin frecventă din toate listele (întâi se va testa dacă în lista asociată tuplului sunt deja N elemente, se va opera corespunzător, și apoi se va testa dacă numărul total de tupluri din toate listele este mai mare ca 2*N).

get (Dictionary, Key)

- Întoarce o listă dublu înlănțuită necirculară, ce conține elementele cu cheia Key. După crearea listei, pentru fiecare element găsit se va incrementa Frequency și se va reordona în mod corespunzător lista din care face parte în cadrul dicționarului.Acestă listă va conține toate sugestiile de corectare date de dicționar pentru cheia Key. Frequency crește cu 1 fiindcă s-a realizat o căutare și frecvența de apariție a cuvântului trebuie actualizată.
- În cazul în care nu există cheia Key în dicționar, se va întoarce NULL.

removeElement (Dictionary, Key, Value)

- Șterge elementul (Key, Value, Frequency) din dicționar.

printDictionary (Dictionary)

- Afișează elementele celor N liste din dicționar sub forma:
  List 0: (Key, Value, Frequency) … (Key, Value, Frequency)
  List 1: (Key, Value, Frequency) … (Key, Value, Frequency)
  …
  List N-1: (Key, Value, Frequency) … (Key, Value, Frequency)

printValue (Dictionary, Value)

- Afișează toate tuplurile care au valoarea Value, în ordinea în care se găsesc în dicționar. Această listă conține toate formele greșite salvate pentru un anumit cuvânt.

printFrequency (Dictionary, Frequency)

- Afișează toate elementele cu frecvența Frequency, în ordinea în care se găsesc în dicționar.

unionValues (Dictionary, Value)

- Creează o listă dublu înlănțuită necirculară cu elementele având valoarea Value, în ordinea în care se găsesc în dicționar.

unionMaxFrequencies (Dictionary)

- Creează o listă dublu înlănțuită necirculară cu elementele de frecvență maximă din fiecare listă, în ordinea în care se găsesc în dicționar.

printList (List)

- Afișează o listă dublu înlănțuită necirculară.

reverseLists (Dictionary)

- Creează un nou dicționar bazat pe cel primit ca parametru, având listele în ordine inversă. Astfel, tuplul de frecvență minimă se va regăsi pe prima poziție în listă.

În funcțiile de afișare, *printList*, *printValue* și *printFrequency*, dacă nu există elemente se va scrie o linie goală (”\n”). Afișarea se va face în fișierul dat ca parametru fiecărei funcții.

NOTARE

- 85 puncte obținute pe teste.

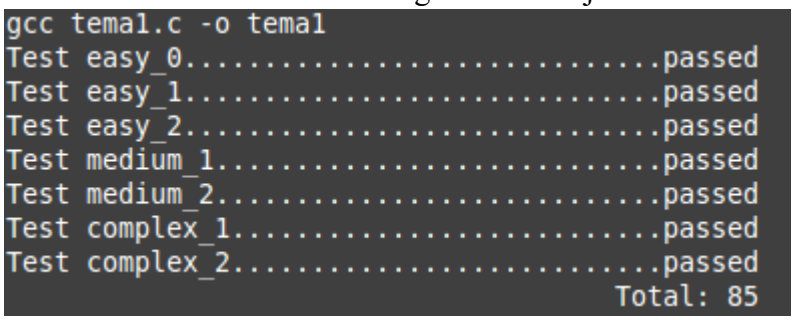
- 10 puncte: coding style - codul trebuie să fie comentat, consistent și ușor de citit.

  Tema **nu** trebuie să conțină: warninguri la compilare, linii mai lungi de 80 de caractere, tab-uri amestecate cu spații, o denumire neadecvată a funcțiilor sau a variabilelor, folosirea incorectă de pointeri, neverificarea codurilor de eroare, utilizarea unor metode ce consumă resurse în mod inutil, neeliberarea resurselor folosite, alte situații nespecificate aici, dar considerate inadecvate.

- 5 puncte: README – va conține detalii despre implementarea temei.

Temele care nu compilează, nu rulează sau obțin punctaj 0 la teste, indiferent de motive, vor primi punctaj 0.

Implementarea funcțiilor se va face în fișierul Dictionary.h. Pentru a testa implementarea, rulați scriptul run.sh (*./run.sh*). Scriptul testează implementarea funcțiilor pe 3 teste ușoare, 2 medii și 2 complexe. Rezultatul ar trebui să arate ca în imaginea de mai jos.



Pentru a verifica leak-urile de memorie puteți folosi targetul check-memory-leaks din Makefile, care se folosește de utilitarul valgrind (*make check-memory-leaks*).

Nu aveți voie să modificați structurile din fișierul Dictionary.h și nici să folosiți variabile statice pentru a reține numărul de elemente din dicționar. De asemenea nu aveți voie să modificați testele din fișierul Tema1.c. Trebuie să completați funcțiile unde scrie TODO și dacă aveți nevoie, puteți să vă definiți funcții auxiliare.

Tema se va încărca pe cs.curs sub forma unei arhive zip, care va conține toate fișierele disponibile în schelet completate cu soluția voastră. Numele acestei arhive va avea forma: *Grupa_Nume_Prenume_Tema1SD.zip*.
Exemplu: 317CD_Ionescu_IonIonica_Tema1SD.zip

Tema se va încărca pe Moodle până la data de 25.03.2018 ora 23:55 și se va prezenta asistentului în laboratorul 6.