Tema 2 – Structuri de date (seria CB)
Operatii cu paranteze.

Responsabili tema:	Cosmin-Dumitru Oprea, Florin-Eugen Iancu
Data publicarii:	**6.04.2017**
Termenul de predare:	**26.04.2017 ora 23:55** Se accepta teme trimise cu penalizare de 10 puncte / zi (din maxim 100 puncte) pana la data de 29.04.2017 ora 23:55

Revizii: 14.04 - clarificari enunt

1. Introducere

O parantezare a unei expresii, de exemplu

```
2 + 5 x 6 - 3
```

este un mod de a pune paranteze in acea expresie pentru a influenta ordinea operatiilor:

```
(2 + 5) x 6 - 3
2 + (5 x 6 - 3)
2 + (5 x (6 - 3))
(((2))) + 5 x (6 - 3)
```

etc.

Cand vorbim de parantezare, nu ne intereseaza expresia insasi, ci doar felul in care se face asezarea parantezelor. Astfel, pentru exemplele de mai sus, avem:

```
()
()
(())
((()))()
```

O parantezare este corecta daca toate parantezele deschise:

- se inchid;
- fac asta in ordinea inversa deschiderii lor.

Urmatoarele sunt gresite, chiar daca numarul de paranteze deschise este egal cu numarul. de paranteze inchise (respecta prima conditie dar nu pe a doua):

```
)(
(()))(
```

Mai mult, daca introducem paranteze de mai multe tipuri: rotunde, patrate, acolade, atunci pe langa cele 2 conditii de mai sus, trebuie ca o paranteza inchisa sa fie de acelasi fel cu cea deschisa cu care face pereche; astfel, urmatoarele cazuri sunt gresite (desi respecta primele 2 conditii):

```
(]
{{]}
{{}]
```

2. Cerintă

In tema, toate parantezele vor fi rotunde, patrate si acolade: `( ) [ ] { }`,
deci 6 tipuri in total.
Veti avea un numar de S stive si C cozi, fiecare dintre acestea memorand cate un sir de paranteze, si un total de N operatii, fiecare dintre acestea prelucrand o anumita stiva sau coada. Cerinta este sa aplicati aceste operatii.

Operatiile sunt:
intrq, extrq, sortq, printq, correctq (pentru cozi)
push, pop, sorts, prints, corrects (pentru stive)
si sunt descrise pe larg in sectiunea urmatoare.

3. Descrierea operaţiilor şi a datelor de intrare

Pe prima linie din fisierul de intrare se afla 3 numere:

- N, un numar natural reprezentand numarul de operatii
- S si C, reprezentand numarul de stive si numarul de cozi asupra carora se vor executa operatii.

Pe urmatoarele N linii se afla operatiile care se vor aplica asupra stivelor si cozilor (descrise mai jos).

push [id_stiva] [id_paranteza] [tip], unde **id_paranteza** este un numar natural si **tip** este un caracter din multimea **{** `(, ), [, ], {, }` **}**

- adauga in stiva cu numarul **id_stiva** perechea (**id_paranteza** - **tip**).

intrq [id_coada] [id_paranteza] [tip_paranteza], unde **id_paranteza** este un numar natural si **tip** este un caracter din multimea **{** `(, ), [, ], {, }` **}**

- adauga in coada cu numarul **id_coada** perechea (**id_paranteza** - **tip**).

pop [id_stiva]

- elimina elementul din varful stivei cu numarul **id_stiva** daca acesta exista.

extrq [id_stiva]

- elimina primul element din coada cu numarul **id_coada** daca acesta exista.

sorts [id_stiva]

- sorteaza crescator stiva cu numarul **id_stiva**, crescator dupa **id_paranteza**
- paranteza cu id-ul cel mai mic se va afla la baza stivei.

sortq [id_coada]

- sorteaza crescator coada cu numarul **id_coada** dupa **id_paranteza**
- paranteza cu id-ul cel mai mic se va afla la inceputul cozii (la capul pe unde se extrag elemente).

prints

- afiseaza toate stivele
- se incepe afisarea de la baza stivei
- nu aveti voie sa iterati prin stiva (trebuie folosite push si pop)
- ordinea de afisare a stivelor este crescatoare in functie de **id_stiva**, de la 0 la S-1.

printq

- afiseaza toate cozile
- se incepe afisarea de la inceputul cozii, adica de la capul de unde se extrag elemente
- nu aveti voie sa iterati prin coada (trebuie folosite intrq si extrq)
- Ordinea afisarii cozilor este crescatoare in functie de **id_coada**, de la 0 la C-1.

Atat pentru **prints**, cat si pentru **printq**:

- se vor afisa doar parantezele, fara id-urile lor
- acestea se printeaza fara spatii intre ele, iar sirul e incadrat de ghilimele (exemplu mai jos)
- cand se trece la o noua stiva/coada, se trece pe linie noua
- exemplu (elemente colorate pentru clarificare):
  - fie coada
    Q = `1 [, 2 (, 3 ),`
    sau stiva
    S =
    `3 )`
    `2 (`
    `1 [`
  - rezultatul afisarii va fi
    `"[()"`

corrects [id_stiva]

- afiseaza lungimea celui mai lung subsir corect parantezat din stiva cu numarul **id_stiva**.

correctq [id_coada]

- Afiseaza lungimea celui mai lung subsir corect parantezat din coada cu numarul **id_coada**.

Atat pentru **corrects** cat si pentru **correctq**:

- intr-un sir corect de paranteze se accepta **orice tip de imbricare**: `([]), [{}], {[()]}` etc
- exemplu:
  - sirul `([{[]([]){}])[]{}()` are subsirul corect cel mai lung `[]([]){}`, deci lungimea este 8
  - sirul `([{[]([]){}}])[]{}()` este corect in intregime, deci lungimea este 20
  - sirul vid este de asemenea corect, lungimea este 0
  - sirurile `(), ()), [[})` sunt gresite complet, lungimea este 0

- sirul `[](){}` are 2 subsiruri corecte de lungime maxima, ambele de 2, deci lungimea este 2.

4. Restricţii şi precizări:

- datele de intrare nu trebuiesc validate
- programul va fi rulat astfel: **./tema2 in_file out_file**
- comenzile se citesc din fişierul **in_file**, iar rezultatele se scriu in fişierul **out_file**
- se garantează că parantezele dintr-o anumita stiva/coada au id-uri unice
- stivele şi cozile vor fi implementate ca liste **generice** simplu inlantuite**.**
- **nu** aveti voie cu variabile globale
- **pentru sortari, afisari si corects/q** se vor folosi doar stive/cozi auxiliare (**nu se va itera prin ele**)
- aveti grija ca tema sa ruleze intr-un timp decent (~1 minut pe un calculator decent pentru testul 9)

5. Exemplu

36 1 1 push 0 6 [push 0 5 (push 0 2) push 0 1 [push 0 7] prints corrects 0 push 0 3] corrects 0 prints intrq 0 8 [intrq 0 9 { intrq 0 0 } intrq 0 4) printq correctq 0 extrq 0 printq correctq 0 extrq 0 sorts 0 prints sortq 0 printq correctq 0 pop 0 pop 0	"[()[]" 4 6 "[()[]]" "[{})" 2 "{})" 2 "[)]([" "})" 0 "" ""

pop 0 pop 0 pop 0 pop 0 extrq 0 extrq 0 **prints** **printq**	

6. Notare

- **85 de puncte** obţinute pe testele de pe vmchecker
- **10 puncte**: coding style; se pot acorda depunctari pentru:
  - warnings (trebuie compilat cu -Wall)
  - linii mai lungi de 80 de caractere
  - folosirea incorectă de pointeri
  - neverificarea codurilor de eroare
  - neeliberarea resurselor folosite (trebuie eliberarea memoriei alocate, inchiderea fisierelor etc)
  - alte situatii nespecificate aici, dar considerate inadecvate
- **5 puncte**: README - va contine detaliile de implementare a temei, precum si punctajul obtinut la teste (la rularea pe calculatorul propriu)
- **bonus: 20 de puncte** pentru soluţiile care nu au memory leak-uri (bonusul se acorda daca testul a trecut cu success)
- temele care nu compileaza, nu ruleaza sau obtin punctaj 0 pe vmchecker, indiferent de motive, vor primi punctaj 0.

7. Reguli de trimitere a temelor

- temele vor trebui incarcate atat pe vmchecker (in secţiunea Structuri de Date **seria CB**) cat si pe cs.curs.pub.ro, in sectiunea aferenta temei 2.
- arhiva cu rezolvarea temei trebuie să fie **.zip** şi să conţină:

- fisiere surse (fiecare fisier sursa va trebui sa inceapa cu un comentariu de forma: **/* NUME Prenume - grupa */**
- fisier **`README`**, denumit obligatoriu astfel, care sa contina detalii despre implementarea temei
- fişier **`Makefile`**, denumit obligatoriu astfel, cu doua reguli:
  build, care va compila sursele si va obtine executabilul, cu numele **tema2**
  clean, care va sterge executabilele si alte fisiere obiect generate
- arhiva nu trebuie să contina decat fisierele sursa (**nu** se accepta fişiere executabile sau obiect)
- daca arhiva nu respecta specificaţiile de mai sus nu va fi acceptata la upload şi tema nu va fi luata in considerare.