Tema 3 SD-CB 2020

Bianca-Mihaela Cauc, Florin Iancu, Roxana Știucă

29 Aprilie 2020

Contents

Detalii

- **Deadline: 14.05.2020** (se acceptă teme trimise cu penalizare de 10 puncte / zi (din max 100 puncte) până la data de **17.05.2020**.
- **Versiune enunț:** 3

Introducere

HyperText Markup Language (HTML) este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser.

Paginile HTML sunt formate din etichete, denumite și tag-uri, și au extensia `.html` sau `.htm`. În marea lor majoritate, aceste etichete sunt pereche, una de deschidere `<eticheta>` și alta de închidere `</eticheta>`, însă mai există și cazuri în care nu se închid, cazuri în care se folosește structura: `<eticheta />`. În plus, unele etichete permit utilizarea de atribute care pot avea anumite valori:

```
<eticheta atribut="valoare"> ... </eticheta>
```

Componența unui document HTML este următoarea:

- versiunea HTML a documentului
- zona antetului, cu etichetele `<head> ... </head>`
- zona informației utile, cu etichetele `<body> ... </body>`

Cerință

Scopul temei constă în implementarea unui program care aplică o serie de comenzi asupra unui cod HTML, cod care nu este neapărat bine formatat (de exemplu, poate fi indentat incorect sau chiar deloc indentat, tot codul fiind scris pe un singur rând).

Structura datelor de intrare

Codul HTML din cadrul fișierului de intrare va fi stocat sub forma unui ***arbore***, în care fiecare nod stochează informațiile corespunzătoare unui tag din codul inițial.

Ținând cont de structura unui cod HTML (menționată mai sus), arborele corespunzător unui cod HTML va fi construit astfel:

- nodul rădăcină al arborelui va fi reprezentat de tag-ul `<html>`
- nodul rădăcină va avea 2 copii care corespund tag-urilor `<head>` și `<body>`
- nodurile intermediare sunt reprezentate de tag-urile care conțin la rândul lor alte tag-uri, acestea urmând să fie stocate drept copii ai nodului curent
- nodurile frunză din cadrul arborelui corespund tag-urilor care nu mai conțin la rândul lor alte tag-uri.

Exemplu

Se consideră următorul cod HTML:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Page title</title>
</head>
<body>
```

```
    <div class="class1"><h1>This is a big text</h1></div>
    <div>
        <p style="color: red;font-size: 50px;" class="class2">Red and big font</p>
        <ul>
            <li>item 1</li>
            <li>item 2</li>
            <li>item 3</li>
        </ul>
        <p class="class2">simple text</p>
    </div>
    <div>
        <img style="float: right;border: 1px;"
            class="imageClass"
            src="../img/purple.png" alt="this is an image"
            width="500" height="600" />
    </div>
</body>
</html>
```

Arborele corespunzător acestui cod este următorul:

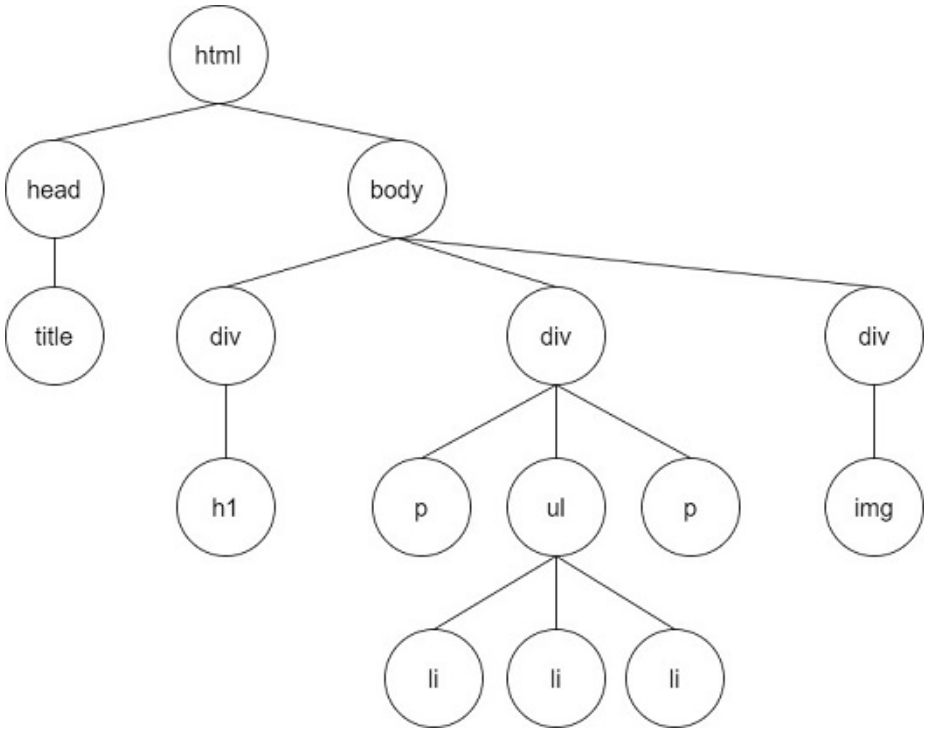


Figura 1: Stocarea codului HTML sub forma unui arbore

Structura unui tag

Pentru simplificare, se consideră că toate tag-urile HTML din cadrul fișierelor de intrare vor respecta una dintre următoarele 2 sintaxe:

- pentru tag-uri normale:

```
<tagName attribute1="valueAttribute1" ... attributeN="valueAttributeN">
    <content>
</tagName>
```

unde `<content>` constă fie într-o listă de alte tag-uri, fie într-un string reprezentând conținutul acelui tag.

- pentru tag-uri self-closing:

```
<tagName attribute1="valueAttribute1" ... attributeN="valueAttributeN"/>
```

Observație: Nu se garantează modul în care elementele componente ale unui tag HTML sunt indentate!

Exemple de tag-uri valide

```
<img class="class1" src="image.png" alt="Text"/>
```

```
<div class="class2" style="font-size:25px;color:blue">
    <p>
        This is a text.
    </p>
    <p>
        This is another text.
    </p>
</div>
```

```
<p style="color:red;margin-right:50px;">This is a red text.</p>
```

Exemple de tag-uri invalide

```
<div class="class2" style="font-size:25px;color:blue">
    <p>
        This is a text.
    </p>
        This is another text.
</div>
```

Detaliile unui tag

Pentru fiecare nod din cadrul arborelui se vor stoca în mod obligatoriu următoarele câmpuri:

- tipul nodului (exemplu: `div`, `p`, `img`, `ul` etc.)
- identificatorul (modul de identificare a nodurilor va fi detaliat mai jos)
- dacă este self-closing tag sau nu
- stil
- conținut

Restul atributelor (în cazul în care există alte atribute) se vor stoca într-o listă simplu înlănțuită, în ordinea apariției.

Generarea identificatorilor nodurilor se face în modul următor:

- rădăcina arborelui nu va avea stocat un ID
- copiii direcți ai rădăcinii (nodurile `head` și `body`) au ID-urile `1` și `2`

- pentru restul nodurilor, ID-urile se calculează pe baza ID-ului nodului părinte la care se vor concatena un separator (punct), respectiv numărul de ordine al copilului, după cum urmează:
  - ID nod părinte: `X.Y.Z`
  - ID nod curent: `X.Y.Z.<număr_ordine_copil>`

Figura următoare prezintă modul de etichetare al ID-urilor nodurilor pentru arborele anterior:

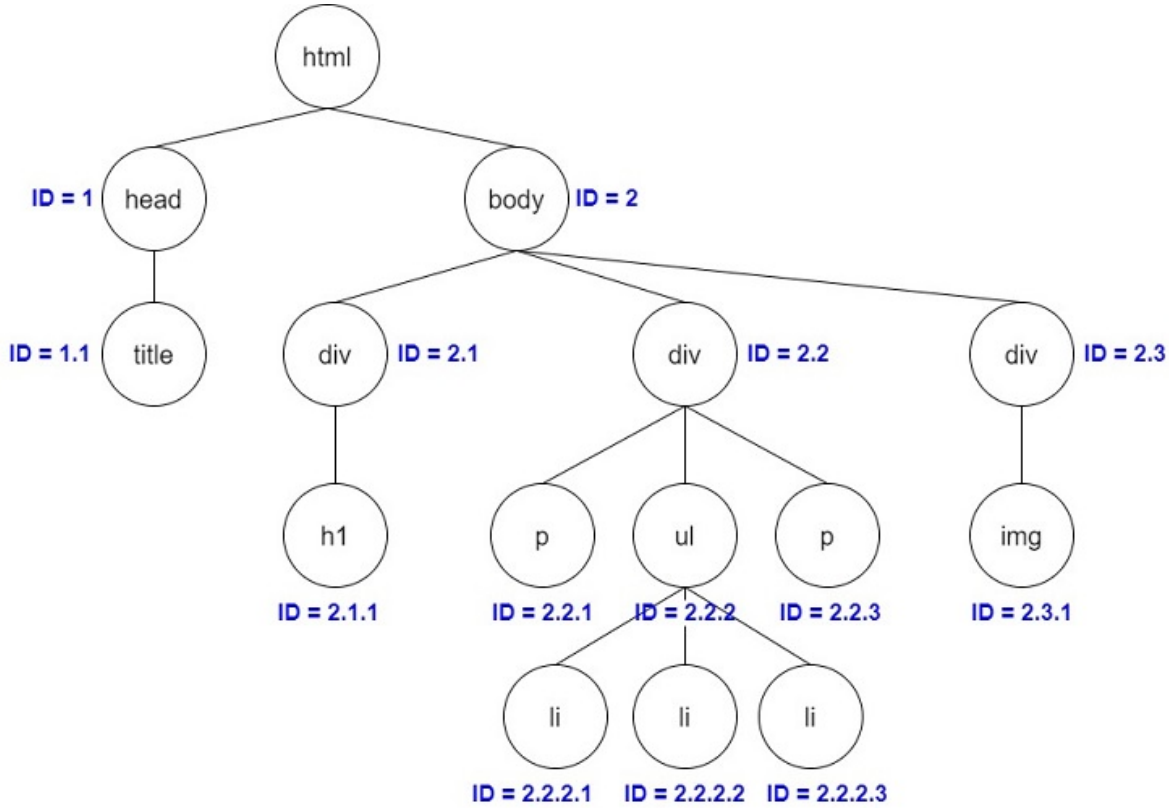


Figura 2: Modul de etichetare al nodurilor din arbore

Implementare

În cadrul implementării, se vor utiliza structurile următoare pentru stocarea codului HTML sub formă de arbore.

```
typedef struct TNodAttr
{
    char *name;
    char *value;
    struct TNodAttr *next;
} TNodAttr, *TAttr;

typedef struct TNodInfo
{
    char *type;
    char *id;
    TAttr style;
    TAttr otherAttributes;
```

```
    int isSelfClosing;
    char *contents;
} TNodInfo, *TInfo;

typedef struct TNodArb
{
    TInfo info;
    struct TNodArb *nextSibling;
    struct TNodArb *firstChild;
} TNodArb, *TArb;
```

Câmpurile principale sunt detaliate în continuare:

Structura **TNodArb**:

- ***info***: reține informațiile referitoare la nodul curent (tip, atribute, conținut etc.)
- ***firstChild***: reține primul nod copil al nodului curent
- ***nextSibling***: reține nodul frate al nodului curent (echivalent cu următorul nod copil al nodului părinte)

Exemplu: În cazul nodului cu ID = 2.2, *firstChild* va fi reprezentat de nodul cu ID = 2.2.1, iar *nextSibling* va fi reprezentat de nodul cu ID = 2.3.

Structura **TNodInfo**:

- ***type***: reține tipul tag-ului HTML (Exemplu: `div`, `p`, `img`, `ul` etc)
- ***id***: reține ID-ul nodului (calculat în modul detaliat anterior)
- ***style***: listă simplu înlănțuită care reține toate subatributele definite în cadrul atributului *style* al tag-ului HTML
- ***otherAttributes***: listă simplu înlănțuită care conține restul atributelor din cadrul tag-ului HTML (cu excepția atributului *style*)
- ***isSelfClosing***: reține dacă tag-ul HTML este self-closing sau nu
- ***contents***: reține conținutul tag-ului HTML

Observații!

- Conținutul tag-ului HTML obținut în urma parsării nu va conține leading whitespaces sau trailing whitespaces. Exemplu: În cazul următorului tag HTML, conținutul este reprezentat de "This is the tag content".

```
<p>      This is the tag content  </p>
```

- Inserarea în liste se va face mereu la finalul acestora, astfel încât acestea vor stoca atributele fix în ordinea apariției în cadrul tag-ului HTML.
- În cadrul fișierelor input, atributul style respectă următoarea sintaxă (**Nu** se garantează totuși modalitatea în care subatributele sunt indentate):

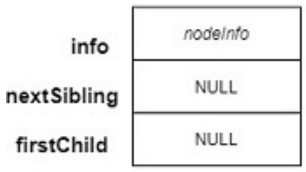
```
style="attribute1:valueAttribute1; ... subattributeN:valueAttributeN;"
```

Exemplu stocare informatii

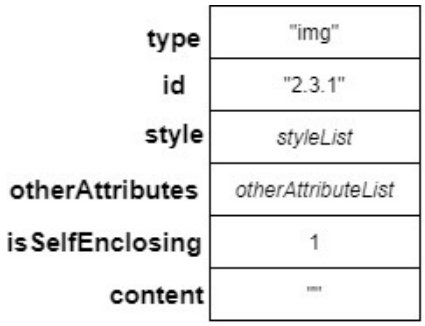
Exemplu 1

În figurile de mai jos este reprezentat modul de stocare al informațiilor pentru nodul cu ID-ul 2.3.1 din arborele anterior, al cărui tag-ul HTML corespunzător este:

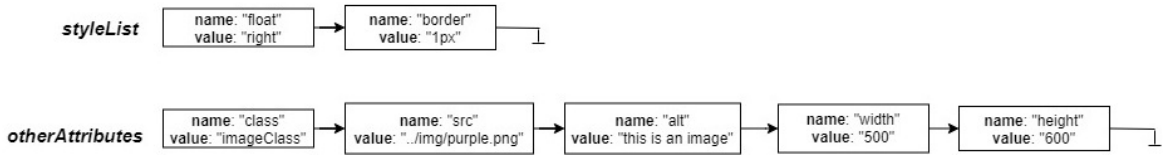
```
<img style="float: right;border: 1px;" class="imageClass" src="../img/purple.png"
        alt="this is an image"  width="500" height="600" />
```



Explicație: Deoarece nodul cu ID-ul 2.3.1 este un nod frunză, câmpul *firstChild* este NULL. În plus, nodul este singurul copil al nodului cu ID-ul 2.3, astfel că valoarea câmpului *nextSibling* este, de asemenea, NULL. Câmpul *info* reține următoarele informații:



unde *styleList* și *otherAttributeList* sunt următoarele:

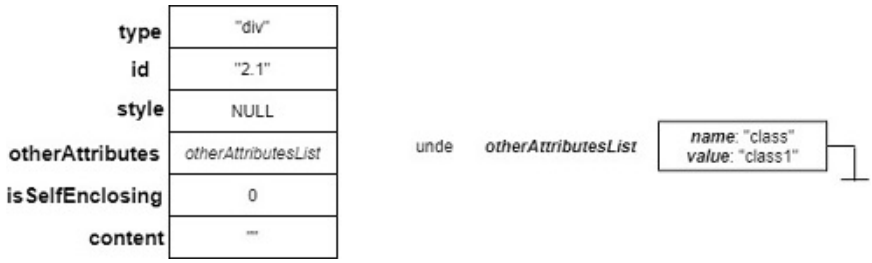


Exemplu 2

Se consideră nodul cu ID-ul 2.1 din arborele anterior, al cărui tag-ul HTML corespunzător este:

```
<div class="class1"><h1>This is a big text</h1></div>
```

În cazul acestui nod, *nextSibling* este reprezentat de nodul cu ID = 2.2, *firstChild* de nodul cu ID = 2.1.1 (tag-ul `<h1>`), iar câmpul *info* va memora următoarele informații:



Exemplu 3

Se consideră nodul cu ID-ul 2.1.1 din arborele anterior, al cărui tag-ul HTML corespunzător este:

```
<h1>This is a big text</h1>
```

În cazul acestui nod, atât *nextSibling*, cât și *firstChild* au valoarea NULL, iar câmpul *info* va memora următoarele informații:

type	"h1"
id	"2.1.1"
style	NULL
otherAttributes	NULL
isSelfEnclosing	0
content	"This is a big text"

Selectori

În continuare, o parte dintre comenzi se folosesc de selectori CSS pentru a determina tag-urile asupra cărora se aplică operațiile. Selectorii posibili împreună cu semnificația lor se regăsesc în tabelul de mai jos:

Selector	Exemplu	Interpretare
`#ID`	`#2.2.1.3`	Caută tag-ul cu ID-ul `2.2.1.3`
`.class`	`.className`	Caută toate tag-urile care au clasa `className`
`element`	`p`	Caută toate tag-urile `<p>`
`element.class`	`p.className`	Caută toate tag-urile `<p>` care au clasa `className`
`element1>element2`	`div>p`	Caută toate tag-urile `<p>` care au părinte un tag `<div>`
`element1 element2`	`div p`	Caută toate tag-urile `<p>` care se află în interiorul unui tag `<div>` (tag-uri `<p>` care au un strămoș cu tipul `<div>`)

Ordinea în care se verifică nodurile din arbore pentru a determina nodurile care respectă selectorul dat este următoarea:

- În cazul selectorului `#ID`, parcurgerea arborelui trebuie să se folosească în mod **eficient** de modul de etichetare al nodurilor, pentru a reduce numărul de noduri vizitate
- În cazul celorlalți selectori, parcurgerea arborelui se va face pe nivele (se utilizează o parcurgere în lățime a arborelui).

Comenzi

Observație: În continuare, în cadrul exemplelor, comenzile vor fi aplicate asupra arborelui reprezentat în cadrul figurii 1.

Comenzile posibile care se pot aplica asupra codului HTML sunt următoarele:

Formatare cod

Comandă: `format`

Comanda formatează un cod HTML stocat sub forma unui arbore în modul prezentat anterior. Formatarea se va realiza prin parcurgerea arborelui în adâncime, respectând următoarele criterii:

- Toate tag-urile HTML (cu excepția rădăcinii) vor fi indentate la un tab suplimentar față de tag-ul HTML părinte
- În cazul în care tag-ul HTML are atribute, se va afișa prima dată atributul `style` și după aceea restul atributelor în ordinea în care acestea apăreau și în codul inițial.
- Atributele vor fi afișate în modul următor: `attributeName="attributeValue"`
- Subatributele din cadrul atributului *style* vor fi afișate în modul următor: `subattribute1: value1; subattribute2: value2; ... subattributeN: valueN;`

Exemplu:

Se consideră următorul cod ca fiind codul inițial care se dorește a fi formatat:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Page title</title></head>
    <body>
        <div class="class1"><h1>This is a big text</h1>
        </div>
        <div>
            <p style="color:red;font-size: 50px;" class="class2">
                Red and big font
            </p>
            <ul><li>item 1</li><li>item 2</li><li>item 3</li></ul>
            <p class="class2">simple text</p>
        </div>
        <div><img style="float: right;border: 1px;"
        class="imageClass"
              src="../img/purple.png" alt="this is an image"
              width="500" height="600"
      />
        </div></body>
</html>
```

După apelul `format`, codul HTML rezultat este următorul:

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>
            Page title
        </title>
    </head>
```

```
    <body>
        <div class="class1">
            <h1>
                This is a big text
            </h1>
        </div>
        <div>
            <p style="color: red; font-size: 50px;" class="class2">
                Red and big font
            </p>
            <ul>
                <li>
                    item 1
                </li>
                <li>
                    item 2
                </li>
                <li>
                    item 3
                </li>
            </ul>
            <p class="class2">
                simple text
            </p>
        </div>
        <div>
            <img style="float: right; border: 1px;" class="imageClass"
src="../img/purple.png" alt="this is an image" width="500" height="600" />
        </div>
    </body>
</html>
```

Adăugare tag

Comandă: `add ID=<ID_părinte> tagHTML="<tagHTML>"`

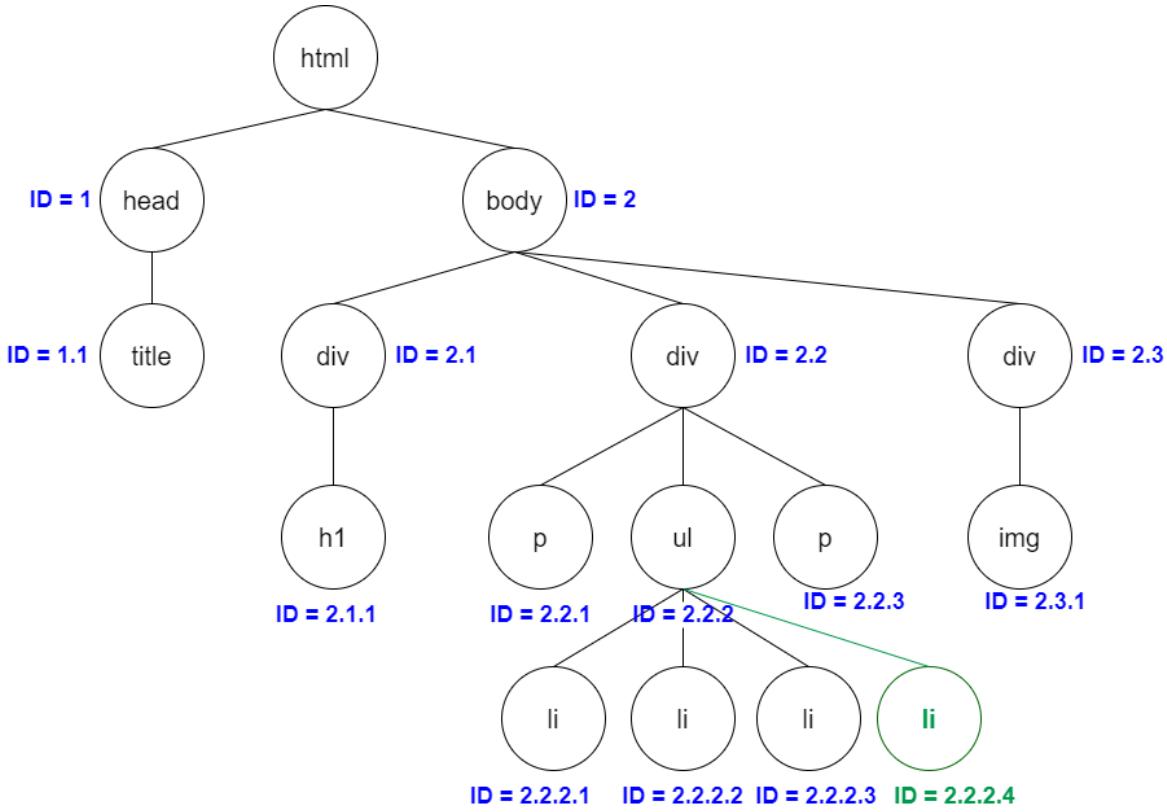
Comanda adaugă în arbore un nou nod copil pentru nodul cu ID-ul dat. Adăugarea acestui nod se va face astfel încât noul nod să reprezinte **ultimul** nod copil al nodului cu ID-ul dat. În cazul în care nu există în arbore un nod cu id-ul `<ID_părinte>`, se va afișa următorul mesaj:

`Add tag failed: node with id <ID_părinte> not found!`

Exemplu:

Comandă: `add ID=2.2.2 tagHTML="<li>This is item4</li>"`

După adăugarea tag-ului, arborele se modifică astfel (nodul adăugat a fost marcat cu verde):



Ștergere recursivă tag

Comandă: `deleteRecursively selector="<CSS_Selector>"`

Comanda caută în arbore nodurile care respectă selectorul dat, iar în cazul în care există, șterge nodurile găsite din arbore. Dacă tag-ul corespunzător nodului care se șterge are la rândul său alte tag-uri HTML în componență (adică nodul curent are unul sau mai multe noduri fiu), atunci nodurile corespunzătoare vor fi șterse de asemenea, împreună cu toate nodurile copii ale acestora în mod recursiv, până se ajunge la nodurile frunze. În cazul în care nodul părinte al nodului șters mai are alte noduri copil, ID-urile lor împreună cu ID-urile nodurilor copii trebuie renumerotate conform noilor poziții.

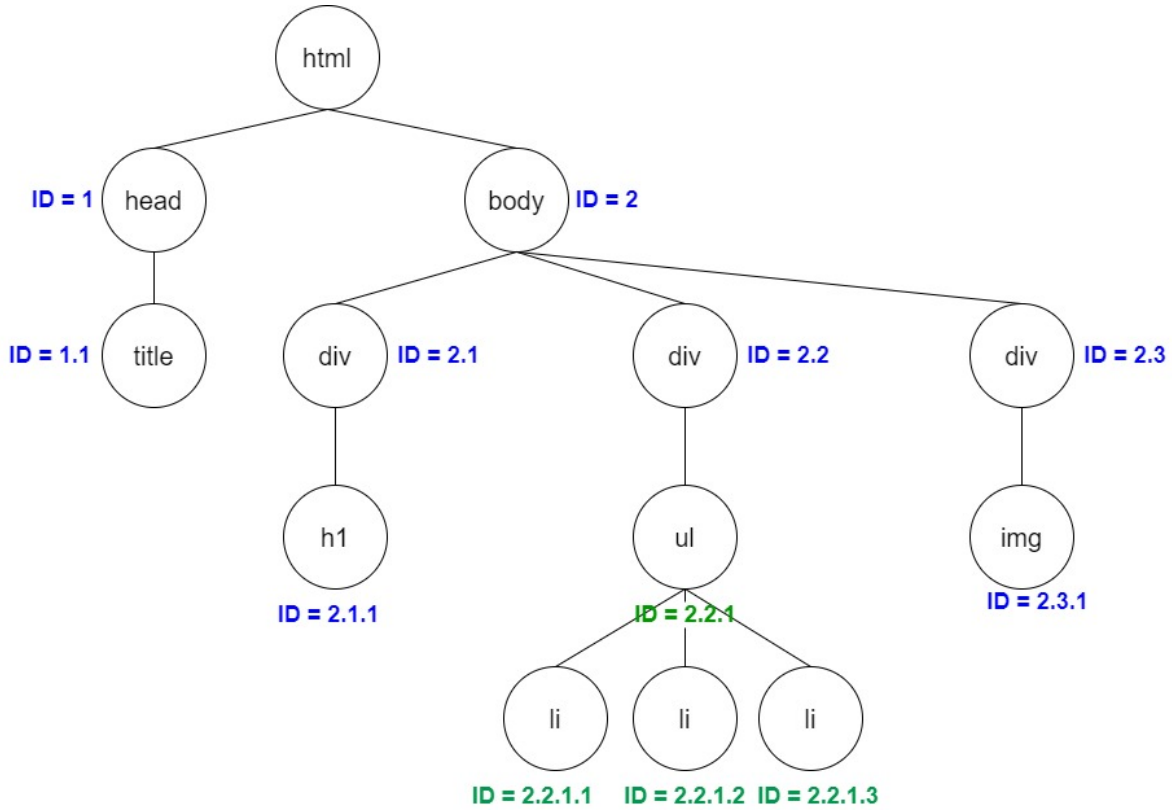
În cazul în care nu există niciun nod care respectă selectorul CSS dat, comanda va afișa următorul mesaj:

`Delete recursively failed: no node found for selector <CSS_Selector>!`

Exemplu:

Comandă: `deleteRecursively selector="p.class2"`

Comanda de mai sus șterge toate tag-urile `<p>` care au clasa `class2`. Astfel, nodurile care vor fi șterse sunt reprezentate de nodurile cu ID-urile 2.2.1 și 2.2.3 din cadrul arborelui inițial. În urma execuției comenzii, arborele inițial se va modifica în modul următor (ID-urile modificate au fost marcate cu verde):



Observație: Comanda `deleteRecursively` nu va fi niciodată apelată cu ID-urile corespunzătoare tag-urilor `<head>` și `<body>`.

Override style

Comandă: `overrideStyle selector="<CSS_Selector>" style="<newStyle>"`

Comanda suprascrie atributul `style` al nodurilor care respectă selectorul dat cu noile atribute memorate în `newStyle`, unde `newStyle` reprezintă un string cu următoarea formă:

```
attributeName1:value1;attributeName2:value2;...attributeNameN:valueN;
```

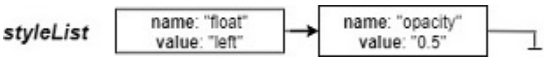
În cazul în care nu există niciun nod care respectă selectorul CSS dat, comanda va afișa următorul mesaj:

`Override style failed: no node found for selector <CSS_Selector>!`

Exemplu:

Comandă: `overrideStyle selector="div>img" style="float:left;opacity:0.5;"`

Comanda de mai sus suprascrie stilul tuturor tag-urilor `<img>` care au drept părinte un tag `<div>`. În cazul arborelui din figura 1, doar nodul cu ID = 2.3.1 respectă selectorul dat, astfel că doar stilul acestuia va fi suprascris. După execuția comenzii, câmpul *style* corespunzător acestui nod va conține o listă cu următoarele elemente:



Append to style

Comandă: `appendStyle selector="<CSS_Selector>" style="<newStyle>"`

Comanda actualizează stilul nodurilor care respectă selectorul dat în modul următor: pentru fiecare subatribut din cadrul câmpului `style` dat ca parametru se verifică dacă există definită deja acea proprietate în cadrul câmpului `style` al nodului. În caz afirmativ, valoarea atributului este actualizată cu noua valoare, iar în caz contrar aceasta se va adăuga la finalul listei de atribute de stil. `<newStyle>` reprezintă un string cu următoarea formă:

```
attributeName1:value1;attributeName2:value2;...attributeNameN:valueN;
```

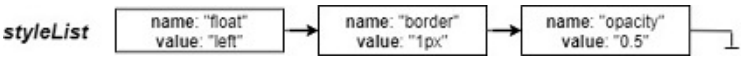
În cazul în care nu există niciun nod care respectă selectorul CSS dat, comanda va afișa următorul mesaj:

`Append to style failed: no node found for selector <CSS_Selector>!`

Exemplu:

Comandă: `appendStyle selector="img" style="float:left;opacity:0.5;"`

Comanda actualizează stilul tuturor tag-urilor `<img>`. În cazul arborelui din figura 1, doar nodul cu ID = 2.3.1 respectă selectorul dat, astfel că doar stilul acestuia va fi actualizat. După execuția comenzii, câmpul *style* corespunzător acestui nod va conține o listă cu următoarele elemente:



Se observă că deoarece exista deja subatributul *float* în listă, valoarea acestuia a fost actualizată; în schimb, în listă nu exista niciun element cu numele *opacity*, astfel că s-a adăugat un nou element la sfârșitul listei corespunzător acestui subatribut.

Citirea fișierului de cod

Pentru a ușura citirea codului HTML din fișier, vi se pune la dispoziție funcția `Interpret`, disponibilă în fișierul `interpret.c`. Aceasta primește un singur caracter (fișierul va fi citit caracter cu caracter în afara acestei funcții, iar ea va fi apelată de fiecare dată) și, în funcție de caracter, trece dintr-o stare în alta. Comentariile din cod arată o sugestie (neobligatorie) despre cum ar trebui completată funcția pentru a construi arborele.

Date de intrare și de ieșire

Programul va primi ca argumente numele a 3 fișiere după cum urmează:

```
./tema3 input.html commands.in commands.out
```

`input.html`

Fișierul conține un cod care respectă sintaxa XHTML 1.1 (un dialect mai strict al HTML standard). Asupra lui se vor aplica toate operațiile din cadrul fișierului `commands.in`.

Se garantează următoarele aspecte asupra fișierelor input:

- Nu vor exista linii in fisier mai lungi de 100 * 1024 caractere
- Conținutul fișierelor input reprezintă un cod HTML valid, nefiind necesară validarea acestora
- Orice element component al codului HTML (nume tag, atribut tag, continut tag) nu va depasi 1000 de caractere.

- Nu vor exista comentarii în cadrul codului HTML
- Orice tag din cadrul fișierului input va avea definită maxim o clasă; de exemplu, nu vor exista tag-uri care au definite mai multe clase:

```
<div class="class1 class2 class3"></div>
```

- Niciun tag HTML din cadrul fișierului input nu va avea definit atributul id; de exemplu:

```
<div id="divID"></div>
```

- Doar nodurile frunză pot conține informație utilă.
- Nu vor exista tag-uri `<script>` care să conțină cod Javascript; de exemplu:

```
<script>
    document.getElementById("demo").innerHTML = "Hello";
</script>
```

- Nu vor exista tag-uri `<style>` care să conțină cod CSS, de exemplu:

```
<style>
    body {background-color: powderblue;}
    h1   {color: blue;}
    p    {color: red;}
</style>
```

Nu se garantează următoarele aspecte asupra fișierelor input:

- Indentarea codului HTML: acesta poate fi corect indentat, parțial indentat sau chiar deloc (fiind posibilă și existența întregului cod HTML pe o singură linie). Astfel, următoarele 2 variante reprezintă fișiere input valide:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Page title</title>
</head>
<body>
    <p>This a text!</p>
</body>
</html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Page
    title</title></head>
<body><p>This
    a text!</p></body></html>
```

`commands.in`

Fișierul conține pe prima linie un număr `N` care reprezintă numărul de comenzi, iar următoarele `N` linii vor conține comenzile efective care vor fi executate asupra codului din cadrul fișierului `input.html`.

Exemplu fișier:

```
6
format
add ID=2.2.2 tagHTML="<li>This is item4</li>"
appendStyle selector="#2.2.2.4" style="color:red;font-size:50px;"
```

```
deleteRecursively selector="p.class2"
overrideStyle selector=".imageClass" style="float:left;"
format
```

`commands.out`

Conține rezultatul produs de fiecare comandă din cele `N` menționate în cadrul fișierului `commands.in`.

Notare

- **85 puncte** obținute pe testele de pe vmchecker
- **10 puncte: coding style**, codul trebuie să fie comentat, consistent și ușor de citit. De exemplu, tema nu trebuie sa conțină:
  - warning-uri la compilare
  - linii mai lungi de 80 de caractere
  - tab-uri amestecate cu spații
  - denumire neadecvată a funcțiilor sau a variabilelor
  - folosirea incorectă de pointeri
  - neverificarea codurilor de eroare
  - utilizarea unor metode ce consumă resurse în mod inutil (alocare de memorie)
  - neeliberarea resurselor folosite (eliberare memoriei alocate, ștergerea fișierelor temporare, închiderea fișierelor)
  - alte situații nespecificate aici, dar considerate inadecvate
- **5 puncte: README**, care va conține detalii despre implementarea temei, precum și punctajul obținut la teste (la rularea pe calculatorul propriu)
- **Bonus: 20 puncte** pentru soluțiile ce nu au memory leak-uri (bonusul se va considera numai în cazul în care a fost obținut punctajul aferent testului)
- **Temele care nu compilează, nu rulează sau obțin punctaj 0 la teste, indiferent de motive, vor primi punctaj 0.**
- **Temele care nu folosesc implementarea cu arbore vor primi punctaj 0.**

Reguli de trimitere a temelor

Temele vor fi încărcate pe vmchecker (în secțiunea Structuri de Date seria CB: SD-CB), dar și pe `cs.curs.pub.ro`, în secțiunea destinată assignment-ului `Tema3`. Arhiva finală a temei rezolvate trebuie sa conțină:

- fișierele sursă
  - fiecare fișier sursă creat sau modificat trebuie să înceapă cu un comentariu de forma: `/* NUME Prenume - grupa */`
- fișierul README în care va fi detaliat modul de implementare al rezolvării
- fișierul Makefile cu două reguli (`build` și `clean`)
  - fișierul trebuie obligatoriu denumit `Makefile` și trebuie sa conțină cele 2 reguli menționate
  - regula `build` va compila sursele și va crea executabilul numit `tema3`
  - regula `clean` va șterge executabilele create

Arhiva va conține numai fișierele menționate mai sus (nu se acceptă fișiere executabile sau obiect). Dacă arhiva nu respectă aceste specificații, aceasta nu va fi acceptată la upload și implicit tema nu va fi luată în considerare.

Referințe

[1] https://ro.wikipedia.org/wiki/HyperText_Markup_Language

[2] https://www.w3schools.com/cssref/css_selectors.asp

[3] https://ocw.cs.pub.ro/courses/programare/coding-style

[4] http://acs.curs.pub.ro/ - curs SD - seria CB - secțiunea Regulament SD - Reguli de realizare, verificare și trimitere a temelor