Zadání úlohy do projektu z předmětu IPP 2016/2017
(Obecné a společné pokyny všech úloh jsou v `proj2017.pdf`)

XQR: XML Query

Zodpovědný cvičící: Radim Krčmář (ikrcmar@fit.vutbr.cz)

1 Detailní zadání úlohy

Skript provádí vyhodnocení zadaného dotazu, jenž je podobný příkazu SELECT jazyka SQL, nad vstupem ve formátu XML. Výstupem je XML obsahující elementy splňující požadavky dané dotazem. Dotazovací jazyk má zjednodušené podmínky a syntaxi.

Skript bude pracovat s následujícímmi parametry:

- `--help` viz společné zadání všech úloh
- `--input=filename` zadaný vstupní soubor ve formátu XML
- `--output=filename` zadaný výstupní soubor ve formátu XML s obsahem podle zadaného dotazu
- `--query='dotaz'` zadaný dotaz v dotazovacím jazyce definovaném níže (v případě zadání tímto způsobem nebude dotaz obsahovat symbol apostrof)
- `--qf=filename` dotaz v dotazovacím jazyce definovaném níže zadaný v externím textovém souboru (nelze kombinovat s `--query`)
- `-n` negenerovat XML hlavičku na výstup skriptu
- `--root=element` jméno párového kořenového elementu obalující výsledky. Pokud nebude zadán, tak se výsledky neobalují kořenovým elementem, ač to porušuje validitu XML.

Dotazovací jazyk Nejprve uveďme neformální zápis struktury dotazovacího jazyka:

```
SELECT element FROM element|element.attribute|ROOT WHERE condition LIMIT n
```

Celá bezkontextová gramatika (včetně povolených rozšíření) je definována takto (neterminály jsou v úhlových závorkách, `<QUERY>` je startující neterminál, tokeny jsou odděleny bílým znakem[1] (je-li to nezbytné) a jazyk je case-sensitive[2]):

```
<QUERY> --> SELECT element FROM <FROM-ELM> <WHERE-CLAUSE> <ORDER-CLAUSE> <LIMITn>
<LIMITn> --> empty
<LIMITn> --> LIMIT number
<FROM-ELM> --> empty
<FROM-ELM> --> <ELEMENT-OR-ATTRIBUTE>
<FROM-ELM> --> ROOT
<WHERE-CLAUSE> --> empty
<WHERE-CLAUSE> --> WHERE <CONDITION>
```

[1] Za *bílý znak* je považována neprázdná posloupnost mezer, tabulátorů a nových řádků.
[2] *case-sensitive* znamená, že v dotazech záleží na velikosti písmen u klíčových slov i u identifikátorů.

```
<CONDITION> --> NOT <CONDITION>
<CONDITION> --> <ELEMENT-OR-ATTRIBUTE> <RELATION-OPERATOR> <LITERAL>
<LITERAL> --> string
<LITERAL> --> number
<RELATION-OPERATOR> --> CONTAINS
<RELATION-OPERATOR> --> =
<RELATION-OPERATOR> --> >
<RELATION-OPERATOR> --> <
<ELEMENT-OR-ATTRIBUTE> --> element
<ELEMENT-OR-ATTRIBUTE> --> element.attribute
<ELEMENT-OR-ATTRIBUTE> --> .attribute
<ORDER-CLAUSE> --> empty
```

Lexémy jsou definovány takto: `empty` je prázdný řetězec. `number` je celé číslo v běžném 32-bitovém celočíselném rozsahu implementačního jazyka. `element` resp. `attribute` jsou identifikátory elementu resp. atributu jazyka XML (bez ohraničujících znaků `<` a `>`). `string` je řetězec zapsaný v uvozovkách, který neobsahuje žádné netisknutelné znaky, escape sekvence, konec řádku, ani uvozovky (nebude testováno).

Sémantika dotazovacího jazyka: Dotaz v klauzuli `FROM` definuje zdrojový element (viz neterminál `<FROM-ELM>`), kde se následně hledají vnořené výstupní elementy z klauzule `SELECT`, které splňují podmínky dané v klauzuli `WHERE`. Poté může být výsledný seznam elementů ořezán omezením `LIMIT` na požadovaný maximální počet elementů.

Hledání zdrojového elementu provádějte hledáním do hloubky, dokud nenarazíte na první výskyt zdrojového elementu dle následujících podmínek pro klauzuli `FROM`:

- Pokud je klauzule tvaru `FROM element`, je hledán první výskyt elementu `element`.
- Pokud je klauzule tvaru `FROM element.attribute`, je hledán první výskyt elementu `element` obsahující atribut `attribute`.
- Pokud je klauzule tvaru `FROM .attribute`, je hledán první element obsahující atribut `attribute`.

Teprve zde bude prováděno další zpracování (již se neuvažuje další nepřekrývající zdrojový element jinde na vstupu). Vzájemné zanoření totožných výstupních elementů neřešte. Nicméně uvažujte, že výstupní element může být pokaždé zanořen do jiné úrovně ve zdrojovém elementu (nepřekrývajícím způsobem). Klíčové slovo `ROOT` zastupuje virtuální kořenový element zastupující celý XML dokument, který pak obsahuje skutečný kořenový element. Entita (element nebo atribut elementu) z podmínky v klauzuli `WHERE` se hledá opět do hloubky po první svůj výskyt elementu dle stejných vlastností pro výběr vhodné entity jako v případě hledání zdrojového elementu s tím rozdílem, že `.attribute` a `element.attribute` vrací hodnotu atributu, nikoliv elementu. Není-li hledaná entita v aktuálně kontrolovaném výstupním elementu nikde nalezena nebo je její první výskyt špatného typu, je výsledek porovnání (viz neterminál `<RELATION-OPERATOR>`) nepravdivý.

Výstupní elementy jsou na výstup kopírovány v nezměněné podobě (včetně všech atributů, hodnot i podelementů[3]), případně obaleny kořenovým elementem v závislosti na parametrech skriptu.

V případě kolize jmen atributů či elementů uvažujte první načtený.

[3]Není třeba kopírovat komentáře.

Pokud bude v dotazu syntaktická nebo sémantická chyba, tak ukončete skript s chybovým hlášením vypsaným na standardní chybový výstup a vraťte návratový kód 80.

Příklady k definici sémantiky dotazovacího jazyka:

```
SELECT book FROM library WHERE title CONTAINS "XML"
```

Projdi všechny elementy `<book>` v elementu `<library>` a vyber ty knihy, které v prvním podelementu `<title>` obsahují podřetězec `XML` (case-sensitive[4]). Pokud `<title>` obsahuje další elementy místo textové hodnoty, tak skonči s chybou 4 (chybný formát vstupního souboru). Vybrané knihy vypisujte na výstup jako kopie vybraných elementů `<book>` včetně všech atributů a podelementů (až na formátování). Podle argumentů skriptu případně doplňte párový kořenový element a XML hlavičku.

```
SELECT title FROM library WHERE NOT title CONTAINS "Duna"
```

Tento dotaz vypíše pomocí XML všechny názvy knih z knihovny, které neobsahují v názvu řetězec `Duna`.

```
SELECT book FROM library WHERE author.name CONTAINS "Herbert"
```

Tento dotaz vypíše knihy z knihovny, kde první nalezený element `author` obsahující atribut `name` obsahuje ve jménu řetězec `Herbert`.

```
SELECT book FROM library WHERE title CONTAINS "Duna" LIMIT 8
```

Vypište XML elementy (vč. podelementů a atributů) pro prvních 8 knih, jejichž název obsahuje podřetězec `Duna`. Pořadí knih je určeno určeno pořadím ve vstupu.

```
SELECT author FROM library WHERE year > 1980
```

Poslední příklad vypíše autory z knihovny, kteří publikovali po roce 1980.

Poznámky:

- Porovnávání řetězců provádějte lexikograficky.
- Při porovnávání rozhoduje typ literálu. Tedy bude-li literál číselný, převeďte element či atribut na číslo.
- Na celočíselný literál nelze aplikovat relační operátor `CONTAINS` (chyba dotazu).
- Celočíselný literál může začínat znaménkem `+` nebo `-`.
- Záporné hodnoty pro klíčové slovo `LIMIT` jsou chybou vstupu.
- Ve vstupním XML souboru se mohou vyskytovat i desetinná čísla. U desetinných čísel na vstupu uvažujte pouze formát: volitelné znaménko (`+` nebo `-`), neprázdná posloupnost číslic 0 až 9, tečka (`.`), neprázdná posloupnost číslic 0 až 9.

[4] *case-sensitive* vyhledávání řetězců vyžaduje shodu i ve velikosti všech písmen.

- Je-li u `FROM <FROM-ELM>` vynechán `<FROM-ELM>`, výstupem bude prázdný soubor (respektive, v závislosti na zadaných parametrech, hlavička či kořenový element).
- `NOT` se může vyskytovat vícekrát a vyhodnoťte jej dle algebraických zvyklostí: `NOT NOT x` = `x`.
- Klíčová slova (`SELECT`, `FROM`, ...) se nebudou vyskytovat v identifikátorech elementů a identifikátory elementů a atributů nebudou obsahovat tečku.
- Chybí-li argument `--query` i `--qf`, považujte dotaz za prázdný a tedy chybný. Stejně tak nelze-li soubor s dotazem z nějakého důvodu číst.

2 Bonusová rozšíření

- **ORD** (1 bod): Jedním z rozšíření je podpora klauzule `ORDER BY`, která lexikograficky seřadí vracené elementy, přičemž ke každému řazenému elementu přidá atribut `order` s hodnotou odpovídající pozici v seřazeném seznamu. První vracený element má index 1. Řazení je stabilní a proběhne před možným ořezáním výsledků pomocí `LIMIT`.

  Neformální zápis pravidel s rozšířením:

  ```
  SELECT element FROM element|element.attribute|ROOT WHERE condition
                 ORDER BY element|element.attribute ASC|DESC LIMIT n
  ```

  Gramatika je rozšířena o následující pravidla

  ```
  <ORDER-CLAUSE> --> ORDER BY <ELEMENT-OR-ATTRIBUTE> <ORDERING>
  <ORDERING> --> ASC
  <ORDERING> --> DESC
  ```

  Za chybu vstupního souboru považujte situaci, kdy ve vstupu není element či atribut podle kterého se má řadit.

- **LOG** (2 body): Dalším z možných rozšíření je podpora skládání podmínek pomocí závorek a klíčových slov `AND`, `OR` a `NOT`. Nejvyšší prioritu má operátor `NOT`, nižší má `AND` a nejnižší `OR`. Operátory `AND` a `OR` jsou levě asociativní.

  Gramatika je rozšířena o následující pravidla

  ```
  <CONDITION> --> ( <CONDITION> )
  <CONDITION> --> <CONDITION> AND <CONDITION>
  <CONDITION> --> <CONDITION> OR <CONDITION>
  ```

Příklady k bonusovým rozšířením:

```
SELECT book FROM library WHERE title CONTAINS "Duna" AND
                               (.name = "Frank Herbert" OR year > 1980)
           ORDER BY year DESC LIMIT 8
```

Vypište XML elementy (vč. podelementů a atributů) pro 8 knih, jejichž název obsahuje podřetězec `Duna` a jejichž autor uvedený v libovolném prvně načteném atributu `name` uvnitř libovolného elementu uvnitř elementu `<book>` (v libovolném zanoření, tedy i přímo v elementu `<book>`) je `Frank Herbert` nebo rok (element `<year>`) je větší jak číslo `1980`. Výsledek seřaďte sestupně podle obsahu elementu `<year>` a až poté vyberte prvních 8 knih (dáno klauzulí `LIMIT`). Pokud dojde ke kolizi jmen, například atributu `name`, ve více elementech, tak uvažujte první nalezený atribut v elementu `<book>` nebo jeho podelementech. Relační operátor pracující s neexistujícím (nedefinovaným) elementem nebo atributem vždy vrací nepravdivou hodnotu.

```
SELECT author FROM library WHERE (year > 1980 OR year = 1980) AND year < 1990
```

Příklad vypíše autory z knihovny, kteří publikovali v letech 1980 až 1989 včetně.

3 Poznámky k hodnocení

Výsledný XML soubor bude porovnáván nástrojem JExamXML pro porovnání XML souborů, který se umí správně vypořádat s rozdílným uspořádáním podelementů v rámci stejného elementu, různým odsazením, či s vypuštěním komentářů z původního vstupního XML souboru.