Univerza v Ljubljani, Fakulteta za elektrotehniko
Podiplomski magistrski študijski program 2. stopnje Elektrotehnika
Biomedicinska tehnika - Biomedicinska informatika

NAVODILA

Vaja 6: Elektronski zdravstveni zapis

Pripravila: Žiga Bizjak & Tomaž Vrtovec

Navodila

Elektronski zdravstveni zapis - EZZ (ang. electronic health record, EHR) je koncept sistematičnega zbiranja zdravstvenih podatkov o posameznih bolnikih v elektronski obliki. Predpostavimo, da imamo EZZ zapisan z razširljivim označevalnim jezikom (ang. extensible markup language, XML), pri čemer je zapis nekoliko poenostavljen (značke ne vsebujejo atributov, ime značke se ne ponovi na istem nivoju, vrednost značke je vedno besedilo), npr. kot:

```
<medications>
    <medication2>
        <name>Ultop</name>
        <fullname>Ultop 10 mg trde kapsule</fullname>
        <usage>1 kapsula vsakih 24 ur</usage>
        <code>040762</code>
        <dateOfPrescription>
            <dd>15</dd>
            <mm>06</mm>
            <yyyy>2012</yyyy>
        </dateOfPrescription>
        <dateOfExpiration>
            <dd>06</dd>
            <mm>07</mm>
            <yyyy>2012</yyyy>
        </dateOfExpiration>
    </medication2>
</medications>
```

Zapis v obliki Pythonove strukture, ki je enakovreden zgornjemu XML besedilu, je enak:

```
\textbf{{'medications':{
    'medication2':{
        'name': 'Ultop',
        'fullname':'Ultop 10 mg trde kapsule',
        'usage': '1 kapsula vsakih 24 ur',
        'code':'040762',
        'dateOfPrescription':{
            'dd':'15',
            'mm':'06',
            'yyyy':'2012'},
        'dateOfExpiration':{
            'dd':'06'
            'mm':'07'
            'yyyy':'2012'}}}}
```

Naloga vaje je, da na osnovi XML razčlenjevanja (ang. parsing) dano XML besedilo pretvorite v zapis v obliki Pythonovega slovarja, nato temu slovarju dodate izbran podslovar ter tako dobljen slovar pretvorite nazaj v XML besedilo.

1. Naložite XML besedilo iz datoteke sampleEHR.xml.:
2. Napišite funkcijo za pretvorbo XML besedila v strukturo:

```
        def xml2dict(iXml, iDict, iTag):
            # ...
            # your code goes here
            # ...
            return oDict
```

kjer vhodni argument `iXml` predstavlja XML besedilo, `iDict` trenutni slovar, `iTag` pa celice z imeni vseh XML značk do trenutne značke. Izhodni argument `oDict` predstavlja novi slovar. Zasnova funkcije naj omogoča rekurzivno klicanje v primeru gnezdenih XML značk, pri čemer je začetni klic enak `EHR = xml2dict(XML, start_dict,[])`. Rekurzija je metoda programiranja, kjer je rešitev danega problema sestavljena iz rešitev posameznih podproblemov, kar dosežemo s klicem funkcije znotraj funkcije same,pri čemer vsakič podamo drugačne argumente in zagotovimo zaustavitveni kriterij.

3. V dobljen slovar dodajte slovar medications iz uvoda.

```
newEHR = EHR
newEHR['medications'].update({'medication2':{}})
newEHR['medications']['medication2'].update({'name':'Ultop'})
newEHR['medications']['medication2'].update({'fullname':'Ultop 10
    mg trde kapsule'})
newEHR['medications']['medication2'].update({'usage':'1 kapsula
    vsakih 24 ur'})
newEHR['medications']['medication2'].update({'code':'040762'})
newEHR['medications']['medication2'].update({'dateOfPrescription'
    :{}})
newEHR['medications']['medication2']['dateOfPrescription'].update({
    'dd':'15'})
newEHR['medications']['medication2']['dateOfPrescription'].update({
    'mm':'06'})
newEHR['medications']['medication2']['dateOfPrescription'].update({
    'yyyy':'2012'})
newEHR['medications']['medication2'].update({'dateOfExpiration'
    :{}})
newEHR['medications']['medication2']['dateOfExpiration'].update({'
    dd':'15'})
newEHR['medications']['medication2']['dateOfExpiration'].update({'
    mm':'06'})
newEHR['medications']['medication2']['dateOfExpiration'].update({'
    yyyy':'2012'})
```

4. Napišite funkcijo za pretvorbo strukture v XML besedilo:

```
        def dict2xml(iDict, iXml, iOffset),
            # ...
            # your code goes here
            # ...
            return oXml
```

kjer vhodni argument `iDict` predstavlja slovar, `iXml` trenutno XML besedilo, `iOffset` pa zamik trenutne XML značke. Izhodni argument `oXML` predstavlja novo XML besedilo. Zasnova funkcije naj omogoča rekurzivno klicanje v primeru gnezdenih XML značk, kjer je začetni klic funkcije enak `XML = dict2xml(EHR, '', '')`. Za prehod v novo vrstico XML besedila uporabite ASCII kodo 10 oz. char(10).

5. Shranite XML besedilo v datoteko newEHR.xml.

Vprašanja

Odgovore na sledeča vprašanja zapišite v poročilo, v katerega vstavite zahtevane izrise in programske kode.

1. Naštejte prednosti in slabosti zapisa podatkov v XML obliki ter vsako izmed njih na kratko obrazložite.

2. Napišite iterativno funkcijo `factI` ter rekurzivno funkcijo `factR`:

```
        def factI(iValue),
            # your code goes here
            return oValue
        def factR(iValue),
            # your code goes here
            return oValue
```

   kjer vhodni argument iValue predstavlja naravno število N, izhodni argument oValue pa fakulteto (faktorielo) števila N, torej N! .Priložite programsko kodo obeh funkcij.

3. Napišite funkciji `xml2dict_attr` in `dict2xml_attr`, in sicer spremenite funkciji `xml2dict` in `dict2xml` tako, da bosta omogočali branje in pisanje atributa type, ki predstavlja vrsto vrednosti, npr.:

   (a) značka ¡id type=”number»123456¡/id¿ naj predstavlja število EHR.id = 123456,

   (b) značka ¡dd type=štring»12¡/dd¿ naj predstavlja besedilo EHR.dd = ’12’.

   V primeru, da atribut type ne obstaja, potem je vsebina značke gnezdeno XML besedilo. Atribut je vedno vsebovan znotraj XML značke, od njenega imena pa je ločen z enim ali več presledkov. Sestavljen je iz imena atributa (besedilo levo od enačaja =, npr. type) in vrednosti atributa (besedilo desno od enačaja =, vsebovano znotraj navednic ”, npr. number). Za preizkus pravilnosti uporabite XML besedilo iz datoteke sampleEHR_attr.xml. Priložite programsko kodo spremenjenih funkcij xml2struct attr in struct2xml attr.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati k poročilu, prispevajo pa naj k boljšemu razumevanju vsebine.

Obravnavani XML zapis je nekoliko poenostavljen, npr. značke niso vsebovale atributov, ime značke se ne ponovi na istem nivoju, vrednost značke je vedno besedilo. V splošnem ima pa lahko vsaka XML značka poljubno število atributov:

```
    <tag attr1="value1" attr2="value2" ... attrN="valueN">value</tag>,
```

kjer je tag ime značke, value je vrednost značke, attrN je ime n-tega atributa, valueN pa je vrednost n-tega atributa. Napišite funkcijo za razčlenjevanje dane vrstice XML besedila v strukturo:

```
        def xmlLineParse(iXml),
            # your code goes here
            return oDict
```

kjer vhodni argument iXml predstavlja dano vrstico XML besedila, ki vsebuje eno značko in poljubno število atributov, izhodni argument oDict pa predstavlja razčlenjen slovar oblike:

```
oStruct['tag'] = 'value'
oStruct['attr1'] = 'value1' % ime in vrednost atributa 1
oStruct['attr2'] = 'value2' % ime in vrednost atributa 2
...
oStruct['attrN'] = 'valueN' % ime in vrednost atributa N
```

Preizkusite delovanje funkcije na naslednji vrstici XML besedila:

```
    <medication dd="15" mm="06" yyyy="2012">Ultop</medication>,
```