KAUNO TECHNOLOGIJOS UNIVERSITETAS
FAKULTETAS

Programavimo kalbų teorija
Laboratorinio darbo ataskaita

Atliko:
IFF-2 gr. studentas
Žygimantas Benetis
2015-02-25
Priėmė:
Doc. Tomas Neverdauskas

KAUNAS 2015

TURINYS

Python laboratorinis darbas

1. Darbo užduotis

This problem requires you to write a program to syntactically validate some simple text written using HTML, the HyperText Markup Language in which all documents available on the World Wide Web are written. We'll not consider the semantics, or meaning, of these documents, and will only consider simplified syntactical rules. In these documents you'll find ordinary text (with arbitrary line lengths) interspersed with markup tags. The markup tags we consider will always occur in pairs. An example will illustrate this point:

```
This is some ordinary text. <ATAG>  Here's some additional
text. <BOLD>  This will be boldface. </BOLD>  Still more
</ATAG>
```

There are two pair of markup tags in the example. The meaning of ATAG and BOLD is unimportant to us in this problem, but typically a markup tag requests particular treatment of the text to which it applies. The markup tags are easily identified because they always appear in angle brackets (that is, a less than symbol and a greater than symbol). The tags we'll consider will always be written as a sequence of no more than 10 upper-case alphabetic characters. The end of the document region affected by the tag is indicated by a tag with the same name preceded by a forward slash, \`/' As illustrated, the tagged regions may encompass more than one line of text. Also as shown in the example, the HTML tags must nest properly, just like BEGIN...END pairs in Pascal, or \`{' and \`}' in C/C++.

2. Programos tekstas

```
def list_append(lst, item):
    lst.append(item)
    return lst

def getInput():
    def readFile():
        f = open('input_data', 'r')
        return f.read().split('\n')
    fileContent = readFile()
    def groupLinesIntoCases(currentArrayIndex, caseList):
        caseSize = int(fileContent[currentArrayIndex])
        if caseSize == 0:
            return caseList
        caseStartingIndex = currentArrayIndex + 1
        caseEndingIndex = caseStartingIndex + caseSize
        caseString = fileContent[caseStartingIndex:caseEndingIndex]
            return groupLinesIntoCases(caseEndingIndex, list_append(caseList, '\n'.join([str(x) for x in
caseString])))

    return groupLinesIntoCases(0, [])

def problemSolution(cases):
    import re
    caseError = ''

    def isCaseCorrect(caseString):
        def isTagCorrect(tag):
            insideTag = re.search('\<\/?([a-zA-Z]{1,10}?)\>', tag)
            if insideTag:
                return True
            else:
                problemSolution.caseError = 'Tag ' + tag + ' is incorrect'
                return False

        def isTagsOrderValid(tagList):
            def isOpenTag(tag):
                if tag[1] == '/':
                    return False
                else:
                    return True
            def makeOpenTag(tag):
                return tag.replace('/', '')
            tagStack = []
            for itTag in tagList:
                if isOpenTag(itTag):
                    tagStack.append(itTag)
                else:
                    if not tagStack:
                        problemSolution.caseError = 'No opening tag of '+ itTag
                        return False
                    removedTag = tagStack.pop()
                    if removedTag != makeOpenTag(itTag):
                        problemSolution.caseError = 'Expected ending tag of '+removedTag
                        return False
            return True

        problemSolution.caseError = ''
        tagIterator = re.finditer('\<\/?.*?\>', caseString)
        tagList = list(map(lambda x: x.group(), tagIterator))
        areTagsValid = tagList == filter(isTagCorrect, tagList)
        return areTagsValid and isTagsOrderValid(tagList)

    def checkCases(casesToCheck):
        solution = map(lambda (i,x): "Case #{} : Correct".format(i+1) \
            if isCaseCorrect(x) else "Case #{} : Incorrect. Error: ".format(i+1)+problemSolution.caseError,\
            enumerate(casesToCheck))

        print('\n'.join(solution))

    checkCases(cases)

problemSolution(getInput())
```

3. Pradiniai duomenys ir rezultatai

inputdata

```
5
This is some ordinary text.
<BEGIN> This is included in the BEGIN tag &lt;/BEGIN>
<START> Here's some stuff
and so is this
more stuff. </START>
2
This has a null tag <>
And an extra line after the error
5
This has some good stuff <OKAY> and some bad stuff later on.
<GOOD> All is still okay, but later on we'll have an error.
</GOOD> We're still in the pink! <THISISTOOLONG>
This line will be skipped.
As will this one.
1
This one is okay <IN> </IN>
1
Mismatch <START> </STOP>
1
```

```
Missing start symbol: <OK></OK></NOTOK> more garbage...
17
<nav id="app-switcher" class="aui-dropdown2 aui-style-default aui-dropdown2-disable-active-class">
    <div class="aui-dropdown2-section blank-slate">
      <h2>Connect Bitbucket with other great Atlassian products:</h2>
      <dl>
        <dt class="jira">JIRA</dt>
        <dd>Project and issue tracking</dd>
        <dt class="confluence">Confluence</dt>
        <dd>Collaboration and content sharing</dd>
        <dt class="bamboo">Bamboo</dt>
        <dd>Continuous integration</dd>
      </dl>
                                <a href="https://www.atlassian.com/ondemand/signup/?product=jira.ondemand,com.atlassian.bitbucket&utm_source=bitbucket&utm_medium=button&utm_campaign=app_switcher&utm_content=trial_button"
          class="aui-button aui-button-primary" target="_blank" id="ondemand-trial">Free trial</a>
                                <a href="https://www.atlassian.com/software?utm_source=bitbucket&utm_medium=button&utm_campaign=app_switcher&utm_content=learn_more_button#cloud-products"
          class="aui-button" target="_blank" id="ondemand-learn-more">Learn more</a>
    </div>
</nav>
0
```

Outputdata

```
Case #1 : Correct
Case #2 : Incorrect. Error: Tag <> is incorrect
Case #3 : Incorrect. Error: Tag <THISISTOOLONG> is incorrect
Case #4 : Correct
Case #5 : Incorrect. Error: Expected ending tag of <START>
Case #6 : Incorrect. Error: No opening tag of </NOTOK>
Case #7 : Incorrect. Error: Tag <dt class="bamboo"> is incorrect
```