Project 1
Simplifying Input
Andrzej Wasylkowski
Your task: use Delta Debugging to simplify demo.xml
Step 1
Write a testing function

- Invoke the XML parser
- Differentiate between three outcomes
  - **Pass**—without errors and warnings
  - **Fail**—the original failing outcome
  - **Unresolved**—another outcome (e.g., a syntax error)
Invoking the Parser

• Invoke directly from Python

• Try-except-else block differentiates between the outcomes

```python
try:
    from xmlproc.xml.parsers.xmlproc import xmlproc
    p = xmlproc.XMLProcessor()
    p.parse_resource(input_filename)
except UnboundLocalError as exc:
    # original failure
except:
    # some other failure
else:
    # parsing finished without errors
```
Step 2

Write a Splitting Function

• Write a `split()` function that splits input into subsets

• Just split the input into smallest parts (for example single characters)

• Use downloadable `split()` function
Step 3
Attach Delta Debugging

- You need `listminus()`
- Download the listsets module
- You need `ddmin()`
- Download the dadmin module
- Hint: It comes with a test function
Step 4
Choose a Representation

• How do you represent the configuration that is to be minimized?

• You have to **uniquely identify** each circumstance (single character)

```
listminus(['d','e','f','e','c','t'], ['e']) = 
['d','f','c','t']
```
Split a String into a List

• Split a string into a list of distinguishable characters

• Add an index to each character

```python
char_list = []
for i in range(0, len(string)):
    char_list.append((i, string[i]))
```
Step 5
Run it

- What is the simplified failure-inducing input in urls.xml?
- Record the number of tests that Delta Debugging needs
Step 6

Is it Documented?

• Be sure to have a docstring for each function

```python
def string_to_list(s):
    """
    Splits a string into a list of distinguishable characters. Returns a list of pairs: (index, character).
    """
    char_list = []
    for i in range(0, len(s)):
        char_list.append((i, s[i]))
    return char_list
```
Step 7
Improve Efficiency

• Add caching to the testing function
• Implement syntactic simplification
• Use an XML parser to split the input along the XML structure
• Narrowing down the failure cause should be faster
Parsing XML Entities

• Use Expat parser to split XML into entities

```python
import xml.parsers.expat

entities = []

def defaultHandler (data):
    entities.append (data)

p = xml.parsers.expat.ParserCreate ()
p.DefaultHandler = defaultHandler
f = file (filename, "r")
p.ParseFile (f)
```
Step 8
Attach Interface

• Your tool should be runnable from the command line exactly as shown here:

  $ python yourtool.py xmlproc/demo(urls).xml
  Relevant input: ...
  Tests performed: ...

• Additional options for caching and XML parsing:

  $ python yourtool.py -cache xmlproc/demo(urls).xml
  ...
  $ python yourtool.py -xml xmlproc/demo(urls).xml
  ...
Project Grading

- Base version: 60%
- Caching: 10%
- XML parsing: 30%
Submission

• 2008-11-19 23:59

• Send .zip archive to:
  wasylkowski@st.cs.uni-sb.de

• Subject should start with [Project 1]

• Ready-to-run as prescribed and documented