Ling 473 Project 3
Due 11:45pm on Thursday, August 25, 2015

For this project you will implement a finite state machine (FSM) that identifies syllables in Thai text. Thai uses a phonetic alphabet of consonants, vowels, and tone marks, but does not use space to separate words or syllables. Therefore, a fundamental problem in text processing for Thai—and many other East Asian languages—is to automatically identify syllable and word boundaries in text.

Normally, an FSM syllable-breaker for Thai is non-deterministic, and requires backtracking or look-ahead in order to resolve ambiguities in the input stream. For this project, we will consider a simplified structure which allows for a (mostly) greedy, deterministic FSM. The input text has been selected so that it falls fully within the reduced model.

At first, I was going to give you a *description* of the simplified version of the Thai orthography, and have you figure out the state machine, but this proved to be a bit ambitious for this project. Instead, I will give you the correct FSM design for you to implement. In addition, because this project does not refer to any copyrighted corpora, you can develop it on your computer at home. However, *your final code must still run on `patas/dryas`*.

Input

Another objective of this project is to gain familiarity with foreign-language encoding systems. Programming languages offer varying degrees of support for either Unicode, UTF-8, or legacy (8-bit) character encodings. For example, to use Unicode in Python 2.4.3 (which is the default version installed on `patas/dryas`), you must use the "unicode" object, which is distinct from the default 8-bit string object. See http://docs.python.org/howto/unicode.html for more details.

To give you maximum flexibility, the Thai input document will be available in three different encodings: Unicode, UTF-8, and a legacy (8-bit) Thai encoding. **It is preferred that you use the UTF-8 or Unicode file**, but I do not expect anybody to write a UTF-8 decoder, or endure hardship if your programming language cannot elegantly deal with 16-bit characters. The files use UNIX line endings.

Another issue to consider is whether you can save your source code files using the UTF-8 encoding. This makes programming easier, since you can just use the Thai characters as literals in your program. For example, if you are using C#, you can just use the following initialization. You'd need to make sure that your editor can save the source file in UTF-8.

```
HashSet<Char> V1 = new HashSet<Char>("แโใไ");
HashSet<Char> C1 = new HashSet<Char>("กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถทธนบปผฝพฟภมยรฤลฦวศษสหฬอฮ");
HashSet<Char> C2 = new HashSet<Char>("รลวนม");
HashSet<Char> V2 = new HashSet<Char>("ิีึืุูั็");
HashSet<Char> T = new HashSet<Char> { '\u0E48', '\u0E49', '\u0E4A', '\u0E4B' };
HashSet<Char> V3 = new HashSet<Char>("าอยว");
HashSet<Char> C3 = new HashSet<Char>("งนมดบกยว");
```

The three versions of the input document '`fsm-input.`*`format`*`.txt`' are located in the following dropbox location: **`/opt/dropbox/15-16/473/project3`**. You will also find a copy of the above code snippet in a C# skeleton source file which you may use in any way you wish.

Processing

When considered as a stream of code points, the structure of our (simplified) maximal orthographic syllable is as follows. Square brackets show optional elements. Additional constraints are encoded in the FSM state transitions, below.

$$[V_1]\ C_1\ [C_2]\ [V_2]\ [T]\ [V_3]\ [C_3]$$

Category		Members	Unicode	TIS-620
V_1	Preposed vowel	เ แ โ ใ ไ	`0E40, 0E41, 0E42, 0E43, 0E44`	`E0, E1, E2, E3, E4`
C_1	Initial consonant	ก ข ฃ ค ฅ ฆ ง จ ฉ ช ซ ฌ ญ ฎ ฏ ฐ ฑ ฒ ณ ด ต ถ ท ธ น บ ป ผ ฝ พ ฟ ภ ม ย ร ฤ ล ฦ ว ศ ษ ส ห ฬ อ ฮ	`0E01 - 0E2E (inclusive)`	`A1 – CE (inclusive)`
C_2	Clustered consonant	ร ล ว น ม	`0E19, 0E21, 0E23, 0E25, 0E27`	`B9, C1, C3, C5, C7`
V_2	Super- or subscript vowel	◌ิ ◌ี ◌ึ ◌ื ◌ุ ◌ู ◌ั ◌็	`0E31, 0E34, 0E35, 0E36, 0E37, 0E38, 0E39, 0E47`	`D1, D4, D5, D6, D7, D8, D9, E7`
T	Superscript tone mark	◌่ ◌้ ◌๊ ◌๋	`0E48, 0E49, 0E4A, 0E4B`	`E8, E9, EA, EB`
V_3	Postposed vowel or glide	า อ ย ว	`0E22, 0E27, 0E2D, 0E32`	`C2, C7, CD, D2`
C_3	Final consonant	ง น ม ด บ ก ย ว	`0E01, 0E07, 0E14, 0E19, 0E1A, 0E21, 0E22, 0E27`	`A1, A7, B4, B9, BA, C1, C2, C7`

States and transitions of the finite state machine

State #	Action	Transition

0	`Accept V1/C1`	`V1 → 1`
		`C1 → 2`

1	`Accept C1`	`C1 → 2`

2	`Accept C2/V2/T/V3/C3/V1/C1`	`C2 → 3`
		`V2 → 4`
		`T → 5`
		`V3 → 6`
		`C3 → 9`
		`V1 → 7`
		`C1 → 8`

3	`Accept V2/T/V3/C3`	`V2 → 4`
		`T → 5`
		`V3 → 6`
		`C3 → 9`

4	`Accept T/V3/C3/V1/C1`	`T → 5`
		`V3 → 6`
		`C3 → 9`
		`V1 → 7`
		`C1 → 8`

5	`Accept V3/C3/V1/C1`	`V3 → 6`
		`C3 → 9`
		`V1 → 7`
		`C1 → 8`

6	`Accept C3/V1/C1`	`C3 → 9`
		`V1 → 7`
		`C1 → 8`

7	`Break before previous character`	`→ 1`
8	`Break before previous character`	`→ 2`
9	`Break now`	`→ 0`

```
Start state: 0
Accept (final) states: 7, 8, 9
```

If there is no valid transition out of a state based on the current character, then the input would be rejected. Note, however, that *the test data for this project has been selected so that a correct FSM will not reject any lines*, and will correctly identify all Thai syllables.

For states 0 through 9, echo the input character to the output. States 7, 8, and 9 are the cases where a syllable break has been detected. To keep the FSM simple as simple as possible, there is one exception to its greedy behavior: for states 7 and 8, the character that triggered the state actually belongs to the *next* syllable, so you will need to figure out how to emit a space *before* the character that caused the transition. The easier case is for state 9, where the character that triggered the state was the last character of a syllable. Here, you will simply emit a space after it.

Because some characters appear in more than one category, the order in which the current character is matched within each state is important. For example, for the state “Accept V3/C3/V1/C1,” you will take the first category that matches, reading from left-to-right.

The input file contains several lines of Thai text. In accordance with Thai prescriptive convention, they contain no spaces. Process each line of the input file separately and in order. For each line, feed it into your FSM, determine the syllable breaks, and write one line to the output file, separating the syllables with a space character.

Output Format

Console terminal output is not suitable for displaying Thai characters. For this assignment, your program will use the file I/O operations in your programming language to write an HTML file which can be displayed in a web browser. As with the input, it is preferred that you create a Unicode or UTF-8 file, although TIS-620 is acceptable if necessary (in this case, you’d change the `<meta>` tag to be `TIS-620`). The following template shows the HTML markup that you should use:

```
<html>
<meta http-equiv='Content-Type' content='text/html; charset=UTF-8' />
<body>
แบ่ง แผ่น ดิน ออก เป็น สอง ส่วน<br />
หลุม พอ ต้น ให ญ่ง อก งาม ชิด ตลิ่ง<br />
...etc...
</body>
</html>
```

When you run your program, it will produce this file with the output inserted as shown. You will be able to view the results by loading this HTML file in a web browser.

Submission

Include the following files in your submission:

`compile.sh`	Contains command(s) that compile your program. If you are using python, shell scripts, or any other interpreted language that does not require compiling, then this file will be empty, or contain just the single line: `#!/bin/sh`
`run.sh`	The command(s) that run your program. Be sure to include compiled binaries in your submission so that this script will execute without first running compile.sh
`your-uw-netid``.html`	This is the HTML file created by running your program, and must be formatted according to the "Output Format" instructions above. If you wish, your program can use the hard-coded path `/home2/www-uakari/html/473` for this file.
`readme.{pdf, txt}`	Your write-up of the project. Describe your approach, any problems or special features, or anything else you'd like me to review. If you could not complete some or all of the project's goals, please explain what you were able to complete.
`(source code and binary files)`	All source code and binary files (jar, a.out, etc., if any) required to run and compile your program

Gather together all the required files, making sure that, for example, any PDF or other binary files are transferred from your local machine using a binary transmission format. Then, from within the directory containing your files, issue the following command to package your files for submission. Replace the bracketed portion with your UW NetID.

```
tar -czf [your-uw-netid].tar.gz .
```

Notice that this command packages all files in the current directory; do not include any top-level directories. Upload the file to CollectIt.

Grading

Correct results	30
Follow submitting instructions	10
Clarity, elegance, and readability of code	25
Program efficiency	20
Write-up	15