Computer Programming Assignment 1

Checkpoints

1. You should do the assignment in your own. You are not allowed to share code with others and/or copy code from other resources. If you are caught, as in the syllabus, you will get a failing grade.
2. Grading will be done in the Linux environment using Java 10.
3. Program failed to compile/run will result 0.
4. Do not loop your program to repeat unless you are told so.
5. Do not change input/output format unless you are told so.
6. Write your name and student number at top of program as a comment.
7. Do not include Korean (and any other language than English) comment. In some encoding formats, Korean comments will cause compilation errors in the Linux environment, which will result in a 0 for your grade.

Submission

1. Submit your assignment on eTL.
2. Zip your file (or tar) as '<Student ID>-assign1.zip'
    a. ex.) 2017-12345-assign1.zip
3. Due of this assignment is Oct 15(Mon) .
4. No late submission is allowed.

Problem 1 Affine Cipher (40 pts)

Write a java program “Affine_cipher.java”(be case-sensitive) that takes a line of text, a char (e or d) indicating operation, and two integer type key value.

Affine cipher consists of two key values and modular operations(in this problem mod 67), and it will map ‘x’ to ‘a*x + b (mod m)’ where ‘a’ and ‘b’ should be coprime.
Since 67 is prime number, all positive number for ‘a’ will be candidates.
e.g.) Let’s assume m = 10, with {1, 2, …, 9, 0}.

- m = 10, x = 4, a = 3, b = 6 -> [4] maps to [3*4 + 6 (mod 10) = 8].
- m = 10, x = 4, a = 4, b = 6 -> invalid since ‘a’ and ‘m’ have 2 as common divisor.

You will be given a file to read from(input file), and your program should read input data line by line.

In the first line, Input text will consist of {a~z}, {A~Z}, {0~9}, and {‘ . ’, ‘ , ’ ,‘!’, ‘?’, ‘#‘}. Starting from ‘a’, all alphabets and number characters will have its value starting from 1.
{a, b, c, d, …, A, B, …, Z, 0, 1, 2, … , !, ?, ‘#‘} → {1, 2, 3, 4, …, 24, 25, …, 46, 47, … 65, 66, 0}

If input text contains a character not in bound, print error message (Leftmost first occurrence).
e.g.) aDT1$$77 yua&abb! -> “Error, input value $ is out of range”

The second line will contain two key values: ‘a’ and ‘b’.

‘a’ key values will be between 1 <= a <= 1023
‘b’ key values will be between -1024~1024

Depending on operation shown in the third line, program produces following result:
Encryption(e):
Shift an input text by given key values.
ex.) key = 5 1 : a→f, b→k

Decryption(d):
Shift an input text by inverse operations with given key value.
ex.) key = 5 1 : k→b, f→a

```
[cp00@cp ~]$ cat input.txt
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.,!?#
```

```
7 3
e
[cp00@cp ~]$ java Affine_cipher input.txt output.txt
[cp00@cp ~]$ cat output.txt
jqxELSZ6?fmtAHOV29bipwDKRY5!elszGNU18ahovCJQX4,dkryFMT07#gnuBIPW3.c
```

```
[cp00@cp ~]$ cat input2.txt
jqxELSZ6?fmtAHOV29bipwDKRY5!elszGNU18ahovCJQX4,dkryFMT07#gnuBIPW3.c
7 3
d
[cp00@cp ~]$ java Affine_cipher input2.txt output2.txt
[cp00@cp ~]$ cat output2.txt
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.,!?#
```

```
[cp00@cp ~]$ cat input3.txt
****1
13 5
e
[cp00@cp ~]$ java Affine_cipher input3.txt output3.txt
[cp00@cp ~]$ cat output3.txt
Error, input value * is out of range
```

Problem 2 Tennis (60 pts)

Write a Java code Tennis.java that implements tennis scoring system (https://en.wikipedia.org/wiki/Tennis_scoring_system).
Input and Output will be given as files.
In this task, you need to consider two major events in Tennis: (a) Australian Open (https://en.wikipedia.org/wiki/Australian_Open) and (b) US Open (https://en.wikipedia.org/wiki/US_Open_(tennis)).

- Choose the type of match and gender. For instance, (US Open/Female), (Australian/Male), etc.

- Start a game and proceed by reading inputs from the file. After each input, output the current status to the output file.

Example: (User enters the bold-faced part.)
// Do not print the comments (/* */) below: these are intended for hints!
/ * Although the following case seems quite long, TAs expect you to follow each line so that the program may calculate the correct score for every possible situation. Note that there are many possible cases. */

```
[cp00@cp ~]$ cat game_info.in
/*All input char are in one line. Consider the following line skipping is gone*/
AMLLRLLLRLRLRRLRR /*omitted*/
LLLRLLLLRLL/*omitted*/RLLRLLLLRR/*omitted*/RRL/*omitted*/R
[cp00@cp ~]$ javac Tennis.java
[cp00@cp ~]$ java Tennis game_info.in game_result.out
[cp00@cp ~]$ cat game_result.out
Australian Open/Male chosen.
Current: 0-0
Left wins
Current: 0-0(15-0)
Left wins
Current: 0-0(30-0)
Right wins
Current: 0-0(30-15)
Left wins
Current: 0-0(40-15)
Left wins
Current: 1-0
Left wins
Current: 1-0(15-0)
Right wins
Current: 1-0(15-15)
```

```
Left wins
Current: 1-0(30-15)
Right wins
Current: 1-0(30-30)
Left wins
Current: 1-0(40-30)
Right wins
Current: 1-0(40-40) /* Deuce. */
Right wins
Current: 1-0(40-40A)
Left wins
Current: 1-0(40-40)
Right wins
Current: 1-0(40-40A)
Right wins
Current: 1-1
/* Omitted. */
Current: 5-5(40-15) /* Play until 7-5 when this set becomes 5-5. */
Left wins
Current: 6-5
Left wins
Current: 6-5(15-0)
Left wins
Current: 6-5(30-0)
Right wins
Current: 6-5(30-15)
Left wins
Current: 6-5(40-15)
Left wins
Current: 7-5
Left wins
Current: 7-5 0-0(15-0)
Left wins
Current: 7-5 0-0(30-0)
Right wins
Current: 7-5 0-0(30-15)
Left wins
Current: 7-5 0-0(40-15)
Left wins
Current: 7-5 1-0
/* Omitted. */
Current: 7-5 4-6
Left wins
```

```
Current: 7-5 4-6 0-0(15-0)
Right wins
Current: 7-5 4-6 0-0(15-15)
Right wins
Current: 7-5 4-6 0-0(15-30)
Left wins
Current: 7-5 4-6 0-0(30-30)
Right wins
Current: 7-5 4-6 0-0(30-40)
Left wins
Current: 7-5 4-6 0-0(40-40)
Left wins
Current: 7-5 4-6 0-0(40A-40)
Left wins
Current: 7-5 4-6 1-0
/* Omitted. */
Current: 7-5 4-6 5-5(30-40)
Right wins
Current: 7-5 4-6 5-6
Left wins
Current: 7-5 4-6 5-6(15-0)
Left wins
Current: 7-5 4-6 5-6(30-0)
Right wins
Current: 7-5 4-6 5-6(30-15)
Left wins
Current: 7-5 4-6 5-6(40-15)
Left wins
Current: 7-5 4-6 6-6 /* Tie Break at 6. */
Left wins
Current: 7-5 4-6 6-6(1-0)
Left wins
Current: 7-5 4-6 6-6(2-0)
Right wins
Current: 7-5 4-6 6-6(2-1)
Right wins
Current: 7-5 4-6 6-6(2-2)
/* Omitted. */
Current: 7-5 4-6 6-6(3-5)
Right wins
Current: 7-5 4-6 6-6(3-6)
Right wins
Current: 7-5 4-6 6-7(3-7)
```

```
Left wins
Current: 7-5 4-6 6-7(3-7) 0-0(15-0)
/* Omitted. */
Current: 7-5 4-6 6-7(3-7) 2-5(15-40)
Right wins
Current: 7-5 4-6 6-7(3-7) 2-6
Game finished! /* Australian Open Male. */
[cp00@cp ~]$
```

@ You do not need to take care about doubles (복식).
@ You do not need to consider withdrawals (기권), order of service (서브 순서) and changing courts (코트 체인지).
@ You should output in exactly the same format as in the example. No points will be given if the output is different from what is written in the specification.
@ You may write additional methods, but everything needs to be inside a class `Tennis`. In addition, make sure you put everything into Tennis.java.
@ Graders will not enter wrong inputs.

@ Some useful resources:

- Tie-breaker criterion:

https://en.m.wikipedia.org/wiki/Tennis_scoring_system#Scoring_a_tiebreak_game .
https://en.wikipedia.org/wiki/Longest_tiebreaker_in_tennis .

- Some extreme case links (including advantage sets):

https://ausopen.com/scores/results/C13 .
https://ausopen.com/scores/results/C11 .
https://ausopen.com/scores/results/C8 .