CSCI165 Computer Science II
Lab Prep

Problem Description
In a certain code language called **KenSpeak™**, individual numerals are represented by a either symbol or a letter. This process is called ***transliteration***

You will encrypt numbers by translating them to **KenSpeak™** The code mappings are shown in the following table

Numeral	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Symbol Code	*****	**B**	**E**	**A**	**@**	**F**	**K**	**%**	**R**	**M**

For example, 487692 is encoded into KenSpeak™ as @R%KME

Problem 1:

- Write a program that will ask the user to enter a sequence of base$_{10}$ digits.
- Encode the digit sequence using the table above and display the encoded String.
- You will not know how long the entered number will be so you can either
    - Treat is as a String and use a **for loop** based on the **length** of the input.
    - Leave it as a number and separate the digits using division and a while loop. You'll need to find the appropriate power of 10 to begin this operation
- Use this problem as an incremental step in solving and verifying your approach for problem 2

Problem 2:
- Write a program that reads the file ***numbers.txt*** one line at a time. It should encode the numbers into **KenSpeak™** and write the encoded values to a new file, called: ***encodedNumbers.txt***

Helpful Hints:
- API research: String, Character, Scanner, PrintWriter
- It may help to define a static helper method called:
    - **public static String transliterate(String original)**
- Define a String called codes and assign the values of the Symbol Code table above
    - **String codes = "*BEA@FK%RM";**
    - You could also define this in a primitive char array

- Now you have a **correlation between the index values of the codes String and the numerals** that are being translated. You essentially have that table built and loaded into memory. You are required to use this approach for the assignment.
- You can convert a char to an int in the following way, using the **Character** class
    - **char c = '1'; // c == ASCII 49**
    - **int a = Character.getNumericValue(c);**
    - **now "a" equals integer 1**

Requirements:

- I do not want to see explicit if statements of the following form. Code of this format will immediately receive a 50% reduction in grade.

    - ```
      if(input.charAt(x) == '1')
              encodedString += "B";
      ```

- **Rationale**: We want to write code that could easily respond to code sets of different lengths and characters. Embedding the mapping of the character and digit into our decision logic makes for code that cannot be easily updated.

- Use the mapping of character index to code values as described above

Cool Extension

- Use random numbers to generate a random 10 character transliteration table. Take a look at the ASCII table to find appropriate ordinal ranges and ask Java to give you random numbers in those ranges.
      ```