CMSC113 – COMPUTER SCIENCE 1
Assignment#5 – Due on Wednesday, April 16, 2025.

In this assignment, you will write a Java program to generate a memorizable, yet secure password. Such passwords are passphrases made up of several words from a common dictionary randomly strung together. Look at the file:

~dkumar/CMSC113/Assignments/Assignment5/eff_wordlist.txt

It contains a curated list of words for use in passphrases (from the Electronic Frontier Foundation - eff.org). Here are the first 5 lines from this file:

```
7776
11111 abacus
11112 abdomen
11113 abdominal
11114 abide
...
```

In the file there are 7776 such words (first line of file indicates this). Each word also has a 5-digit code next to it. To create a passphrase, first you decide how many words it is going to be. Let us say, we will create a two-word passphrase. Then, you will take five 6-sided dice and roll them. When you juxtapose all the five rolls together, you get a code. For example, if you rolled, 4, 6, 5, 1, 6 to get the code: 46516. Next, you look up the passphrase words list to locate the word with the code 46516. It is **rebel**. That becomes your first word in the passphrase. Repeat the process for the second word. Say you rolled, 6, 6, 6, 6, 1 (code: 66661) to get the word **zone**. Then your passphrase is **rebel zone**.

See, it is easily remembered. But how safe is this passphrase?

If you pick a passphrase of length 1 out of the list, any hacker can crack your password by trying out all 7776 words. That would be very easy. For passphrases of two words, the hacker would need to try 7776*7776=60,466,176 different passphrases to crack your password. We may be getting somewhere! 60 million tries is a small number for any computer hacker to try, so you would typically need to have a passphrase that is 5 or more words long. For a five-word passphrase it would require 15,000,000,000,000,000,000 (or 15 quintillion) tries! We'll be well protected.

In this exercise you will write a Java program that uses the list of words provided in the file **eff_wordlist.txt** to generate a passphrase of a given length. For example,

```
$ java-introcs PassPhrase 4 <
~dkumar/CMSC113/Assignments/Assignment5/eff_wordlist.txt
Your passphrase is: enviable swirl aqua onion
```

Here are some other example passphrases:

```
marathon strut angles spoils
oval strained vanity rug
stunning prize myth strut
refund trouble impromptu rebirth
```

If you can get a set of five dice (or just one will do, use it five times), try to generate some passphrases of your own by rolling the dice, creating a 5-digit code, and then looking up the word. One quick way to find the word in Linux is to use the command **grep**:

```
$ grep 66661 ~dkumar/CMSC113/Assignments/Assignment5/eff_wordlist.txt
66661 zone
$ grep 56622 ~dkumar/CMSC113/Assignments/Assignment5/eff_wordlist.txt
56622 stuck
```

So, the two-word pass phrase is: **`zone stuck`**

Design your program in steps with the following **requirements**:

- You will read and store the codes and words in two arrays: **an integer array, `codes[]`** and a parallel **string array `words[]`**.
- Write the function: **`readData(codes, words)`** to read and store the entire contents of the file **`eff_wordlist.txt`** (from standard input).
- Write the function: **`getWord(codes, words)`** to return a random word chosen by rolling five dice to produce a code.
- The function **`getWord()`** should use the function**: `randInt(a, b)`** which returns a random number in the range [a, b].
- The function **`getPhrase(n, codes, words)`** to return a passphrase string containing **`n`** randomly chosen words (using **`getWord()`**).

While trying to generate the passphrases by hand (as described above), think about the overall flow of the program considering the functions described above. Work incrementally to implement the program in steps: first read the data, then write code to get a random word, and then write code to generate the n-word passphrase. Test at each step! Several times.

And, start early!

What to hand in:

- A printout of the program file.
- Attach five sample outputs from your program with n=4 in English. Also see the files wordlist_fi.txt (Finnish), wordlist_it.txt (Italian), and wordlist_se.txt (Swedish). Use your program to generate 1 4-word passphrase from each language (Finnish, Italian, and Swedish).
- Write and print out a short 1/2-paragraph reflection on your thoughts on this lab (how this assignment went for you, what went wrong, well, etc.).