Exercise 1: Study Arithmetic Codec

On the course web-site, you find an implementation of an arithmetic encoder and decoder in C++:

bitstream.h	classes for input and output bitstreams (header only)
arithCoding.h	header for arithmetic encoder and decoder classes
arithCoding.cpp	implementation of arithmetic encoder and decoder classes
main.cpp	toy example for usage of arithmetic encoder and decoder

1. Study the arithmetic encoder and decoder in detail (compare it with the lecture slides)
2. Play around with the implementation, try different value for U.
   Try other example messages, pmfs (including values of V), and alphabets.
3. If you don't want to use C++ for the following exercises, rewrite the implementation using a programming language of your choice

Note: We don't need to modify the bitstream classes and the actual implementation of the arithmetic codec in the following exercises.

We only need to modify the Pmf class (in main.cpp) and the usage of the arithmetic codec.

Exercise 2: Arithmetic Coding of 8-bit Audio Data

In the following we want to efficiently code the 8-bit audio file "audioData.raw" from the course web site.

1. Write a first encoder and decoder that use a fixed pmf. The encoder should do the following:
   - Count the number of samples (bytes) in the input file and write it as 32-bit integer at the beginning of the bitstream (use function OBitstream::addFixed(.)).
   - Estimate the marginal pmf, quantize it to V-bits of precision (choose suitable V, check validity).
   - Write all 256 probability masses $p_V(x)$ to the bitstream (each using V-bits).
   - Encode all samples of the input file using arithmetic coding with the estimated pmf.
2. Think about how you can estimate the pmf during encoding and decoding.
   Implement a pmf class that estimates the pmf during encoding and decoding
   (there is already a pure virtual function IPmf::update(.) in the interface class IPmf).
   In principle, you have to count symbols during encoding and decoding.
3. Once you have a working adaptive pmf, try to improve the codec by using conditional coding. That means, use 256 different adaptive pmfs in your encoder and decoder. The pmf for a current sample has to be selected based on the value of the previous sample.