// Lab 1, CS 335J, Winter 2008

// Alphabet Class

// An alphabet is a non-empty, finite set of "symbols" and can

// be used to generate a language (a set of strings). If S is

// an alphabet, S\* denotes the language containing all strings

// of length 0 or more made up from the symbols in alphabet S.

// Since S\* is infinite, it cannot be represented in its entirety

// on a computer. The most interesting and useful method of the

// alphabet class is "getLanguage(int n)" which returns a Vector

// of StringBuffer objects corresponding to all elements of S\*

// with length <= n.

import java.util.\*; // need the Vector class

public class Alphabet {

// \*\* DATA FIELDS \*\*

protected int size; // number of symbols in alphabet

protected char[] symbol; // character symbols

// \*\* CONSTRUCTORS \*\*

public Alphabet(String inString)

// fully-specified constructor

// pre: inString is a non-empty string with no repeated characters

// (characters should be listed in the desired lexicographic order)

// post: creates an alphabet whose symbols are the characters in inString

{

??? // store size

??? // allocate array for symbols

??? // store symbols

}

public Alphabet()

// no-arg contructor

// post: creates alphabet {a, b}

{

??? // use “this” to call working constructor

}

// \*\* ACCESSOR METHODS \*\*

public int getSize()

// returns number of symbols in this alphabet

???

public char[] getChars()

// returns a COPY of the alphabet symbols in a char array

???

// \*\* OTHER METHODS \*\*

public void printAlphabet()

// post: does pretty printing with braces and commas

???

public static void printLanguage(Vector v)

// pre: v contains StringBuffer objects, where the first object

// represents the empty string

// post: v is pretty-printed, one string per line, from head to tail

// print “epsilon” for the empty string (instead of blank)

???

public java.util.Vector getLanguage(int n)

// post: returns a vector (extensible array) holding language S\*

// (in lexicographic order) up through length n; language strings

// are stored in the Vector as StringBuffer objects.

{

java.util.Vector v = new java.util.Vector();

v.add(new StringBuffer("")); // first is the empty string

int finger = 0; // pointer for generating more strings

StringBuffer newString; // holds new string

while(((StringBuffer) v.get(finger)).length() < n)

{

// use v.get(finger) to generate more strings in language

for(int i=0; i<size; i++) // for each alphabet symbol...

{

??? // append symbol to end of v.get(finger) and add to v

}

// advance finger

finger++;

}

return v;

}

// \*\* TEST METHOD \*\*

public static void main(String[] args) {

// create an alphabet S1 = {x}

???

// create an alphabet S2 = {a, b}

???

// create an alphabet S3 = {0, 1, 2}

???

// print S1, S2, S3

System.out.print("S1 = ");

S1.printAlphabet();

System.out.println();

System.out.print("S2 = ");

S2.printAlphabet();

System.out.println();

System.out.print("S3 = ");

S3.printAlphabet();

System.out.println();

System.out.println("Strings of length <= 8 in S1\* are:");

??? // use PrintLanguage appropriately

System.out.println("Strings of length <= 2 in S2\* are:");

??? // use PrintLanguage appropriately

System.out.println("Strings of length <= 3 in S3\* are:");

??? // use PrintLanguage appropriately }

}