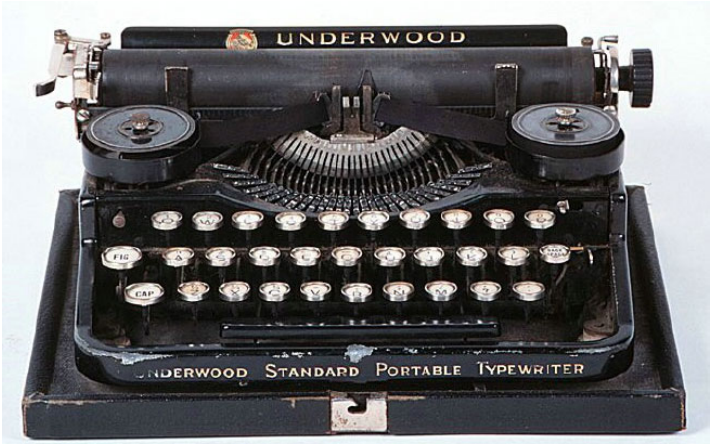
Challenge VI: Keyboard Layout
Is Dvorak really better than QWERTY?

COSC 3327 Algorithms and Data Structures
Due: See Canvas

Background

The old school way of getting text on paper involved the typewriter:

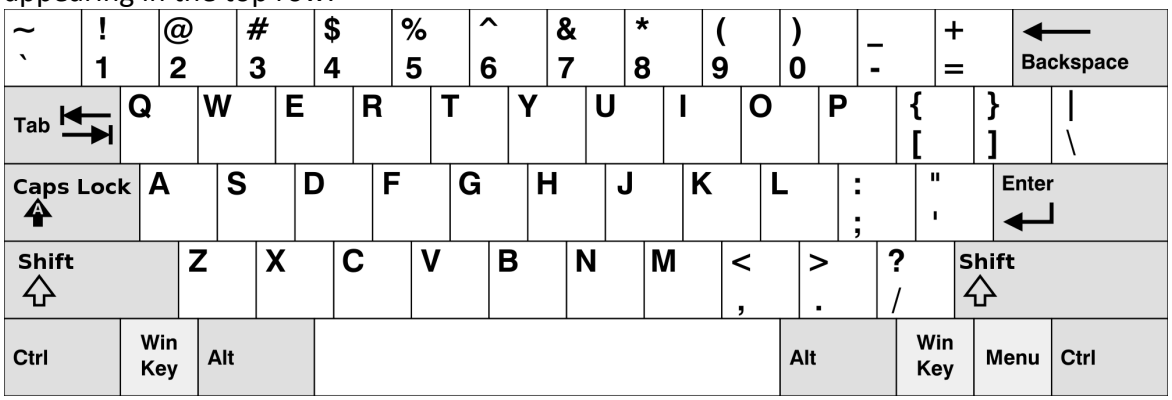


In order to type, an individual sheet of paper would first be inserted into the roller above. Then, as each white key was depressed by the typist's finger, the corresponding "typebar" would rise up to meet the black ribbon, push the ribbon into the paper, and make a black character on the paper. Then the paper would move over one character width.

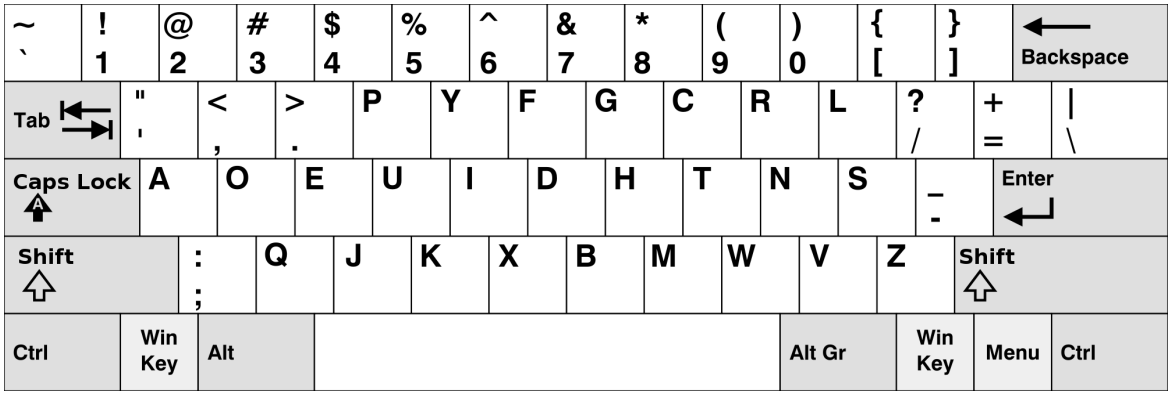
One frequently occurring problem, especially when typing two close keys rapidly in succession, was a typebar jam:

In the 1860's, the inventor of the first mass-produced typewriter, Christopher Sholes, was concerned with mitigating the typebar jam problem. He commissioned a study to determine the most common letters and letter combinations used in English. Then, Sholes used the results to try to "scatter" words across his keyboard. Whether Sholes

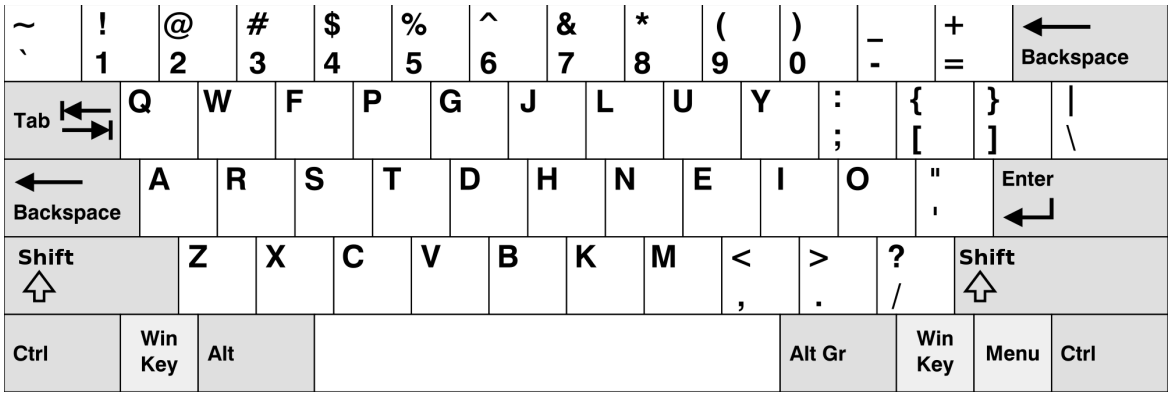
was trying to slow typists down is a matter of debate, but it is not hard to imagine that decreased typebar jams are correlated with decreased speed.
The Remington company purchased his typewriter design, and slightly modified his keyboard layout to produce what is now known as QWERTY because of first six letters appearing in the top row:



August Dvorak was an educational psychologist who, around 1914, became interested in typewriter keyboard layouts. His objective was to scientifically investigate whether there was a keyboard layout that would decrease errors and increase speed over QWERTY. Dvorak's brother-in-law Dealey joined him in this research. They studied the most frequently used letters and letter combinations in the English language, the science of motion, and the physiology of the human hand. In 1932, the Dvorak Simplified Keyboard was introduced:



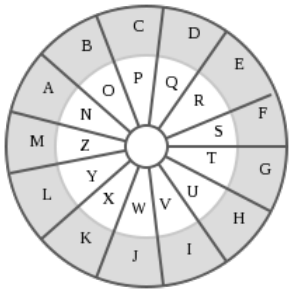
Obviously, there are a great number of possible keyboard layouts. However, a recent trend is for keyboards to drop the Caps Lock key. In 2006, a "Million Dollar Keyboard" competition was held for the best keyboard design to do away with this key. The competition was won by Shai Coleman's "Colemak" keyboard layout:



Here's the Apple Numeric keyboard that I have in my office. This assignment assumes the following geometric layout of the keys while allowing the possibility of remapping the alphabet (e.g. when the 'F' key is pressed a 'T' gets transmitted from the keyboard). So, we could turn the standard QWERTY layout below into a Dvorak layout. In fact, these remapping capabilities are often built into the operating system. However, there is one modification that I want you to incorporate: I've divided the spacebar up into 5 smaller keys (see the Apple Numeric MB110LL Keyboard graph below).

ROT_13

ROT_13 is a permutation of the alphabet in which each letter gets mapped to the letter that is 13 letters "downstream." So,



A gets sent to N,
B gets sent to O,
C gets sent to P,
.
.
.
M gets sent to Z,
N gets sent to A,
O gets sent to B, etc.

Encoding Scheme

You may choose the encoding scheme.

Typing Model

We will need a model of typing in order to calculate finger travel. We will use the venerable "Hunt and Peck" model. Furthermore, we will assume that the typist only has two fingers – one on each hand. The finger on the left hand is always on the SHIFT_1 key, while the finger on the right hand is on the key that corresponds to the last character that was typed. Before beginning to type any string, the finger on the right hand starts on the **physical** key labeled 'J' below. Note that this **physical** key labeled 'J' produces a 'j' or a 'J' in QWERTY, an 'h' or an 'H' in Dvorak, and an 'n' or an 'N' in Colemak.

Finger Travel Model

We will use a simple model to determine distance traveled. The distance traveled when moving the right hand finger from key_start to key_end is defined to be the length of the smallest path key_start, key_1, key_2, key_3, …, key_end, where key_start and key_1, key_1 and key_2, etc. are neighbors. QWERTY key neighbors are defined in the graph on the last page. In particular, notice that the neighbors of 'A' are 'W', 'S', 'Z', 'Q', and 'SHIFT_1'. The neighbors for other layouts are defined in the analogous way.

Assignment

Assume that we are dealing with the Apple Numeric MB110LL keyboard (shown above). Your assignment is to write software that can help answer the question: Which is the best key layout: QWERTY, Dvorak, Colemak, or ROT_13?

You will write code to calculate finger travel when given a key layout (See KeyLayout below) and a String. (See the method KeyboardMetrics.getDistance(String str) below). So, your code will be able to answer the question: Which keyboard layout results in the least finger travel for someone typing the English translation of Tolstoy's *War and Peace*.

Tips

- Do not "reinvent the wheel." Once you have KeyLayout implemented for QWERTY, use these values to figure out all distances for Dvorak, Colemak, and ROT_13.
- `new ArrayList<Key>(keyToNeighborsMap.keySet())` yields a list of all the keys in the Map keyToNeighborsMap.

Deliverables

Upload to Canvas (before 11:59 PM):

- AppleNumericMB110LLKeyboardMetricsImp_*LastName*.java (in package 'keyboard')
- Any other supporting classes, but not including the following interfaces:
    - Key.java
    - KeyLayout.java
    - KeyboardMetrics.java

Constraints

- Ensure that I will be able to compile and run your code
- Ensure that your Impl class and any supporting classes that you create are named with the LastName.java suffix and are in the package named 'keyboard'. In particular, you will have a class named AppleNumericMB110LLKeyboardMetricsImp_*LastName*.java
- Ensure that you use the interfaces provided above without changing the interface name, package, or method signatures. Do not add any additional methods to any of interfaces.

Questions to ask yourself

- The finger travel model we use always produces distances that are integers. Why does the KeyboardMetrics interface deal with doubles?
- What other concepts can I invent to make my programming easier?
- What should the 'input' data structure be to help me get that neighbor graph right?

Code snippets

See next few pages.

```
package keyboard;

public enum Key {
      A('a', 'A'), B('b', 'B'), C('c', 'C'), D('d', 'D'),
      E('e', 'E'), F('f', 'F'), G('g', 'G'), H('h', 'H'),
      I('i', 'I'), J('j', 'J'), K('k', 'K'), L('l', 'L'),
      M('m', 'M'), N('n', 'N'), O('o', 'O'), P('p', 'P'),
      Q('q', 'Q'), R('r', 'R'), S('s', 'S'), T('t', 'T'),
      U('u', 'U'), V('v', 'V'), W('w', 'W'), X('x', 'X'),
      Y('y', 'Y'), Z('z', 'Z'),
      ONE('1', '!'), TWO('2', '@'), THREE('3', '#'),
      FOUR('4', '$'), FIVE('5', '%'), SIX('6', '^'),
      SEVEN('7', '&'), EIGHT('8', '*'), NINE('9', '('),
      ZERO('0', ')'),
      BACKTICK('`', '~'), MINUS('-', '_'), EQUALS('=', '+'),
      TAB('\t', '\t'), LEFT_BRACKET('[', '{'),
      RIGHT_BRACKET(']', '}'), BACKSLASH('\\', '|'),
      SEMICOLON(';', ':'), TICK('\'', '\"'), RETURN('\n', '\n'),
      COMMA(',', '<'), PERIOD('.', '>'), FORESLASH('/', '?'),
      SPACEBAR_1(' ', ' '), SPACEBAR_2(' ', ' '),
      SPACEBAR_3(' ', ' '), SPACEBAR_4(' ', ' '),
      SPACEBAR_5(' ', ' '),
      SHIFT_1(null, null), SHIFT_2(null, null);

      private Character normalCharacter;
      private Character shiftModifiedCharacter;
      Key(Character normalCharacter,
                      Character shiftModifiedCharacter) {
         this.normalCharacter = normalCharacter;
         this.shiftModifiedCharacter = shiftModifiedCharacter;
     }

      public Character getNormalCharacter()
      {
            return normalCharacter;
      }

      public Character getShiftModifiedCharacter()
      {
            return shiftModifiedCharacter;
      }
}
```

```
package keyboard;

public enum KeyLayout {
      QWERTY, DVORAK, COLEMAK, ROTATION_13;
}
```

```
package keyboard;

public interface KeyboardMetrics {
      public double getDistance(Key key1, Key key2);
      public double getDistance(String str);
}
```

```
package keyboard;

public class AppleNumericMB110LLKeyboardMetricsImpl
                                          implements KeyboardMetrics
{
      public AppleNumericMB110LLKeyboardMetricsImpl(KeyLayout
                                                        keyLayout)
      .
      .
      .
}
```

Apple Numeric Keyboard MB110LL QWERTY