**CSC 3020 – Java Programming**

**Homework 3 – Kenneth Gajefski**

**25 points – Due February 23, 10am**

**Late deadline is February 24, 11:59pm, but 20% off**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit this document and your .java file(s) to the Blackboard item where you downloaded this document. Do not submit a zip file but individually attach your files.

**1) [10 points]** You've been hired by *Sort Scorchers* to write a Java console application that sorts an **array list** of random real numbers. Use a validation loop to prompt for and get from the user the number of real numbers to sort in the range 10-10,000. Create and store in the array list that many random real numbers. Randomly generate the real numbers in the range 0-1,000. Print the array list size and use formatted output to print the first ten values of the array list in two columns:

● The first column is the zero-based index of the value.

● The second column is the right-justified value.

Use the insertion sort method to sort the array list. Monitor the number of cycles and swaps during the sort, and the elapsed time in milliseconds to complete the sort. Use formatted output to print the results (cycles, swaps, total cycles and swaps, and elapsed time) in two columns:

● The first column is a left-justified label. Include units if needed.

● The second column is a right-justified value.

Print the array list size and first ten values of the array list again. Format all numbers with a comma. Format all real numbers to one decimal place. Run the program three times with the following inputs. Enter the elapsed times for each run:

|  |  |  |
| --- | --- | --- |
| Run | Array list size | Elapsed time (ms) |
| 1 | 10 |  |
| 2 | 5,000 |  |
| 3 | 10,000 |  |

*[your program code here]\**

**If possible, format your code like this:**

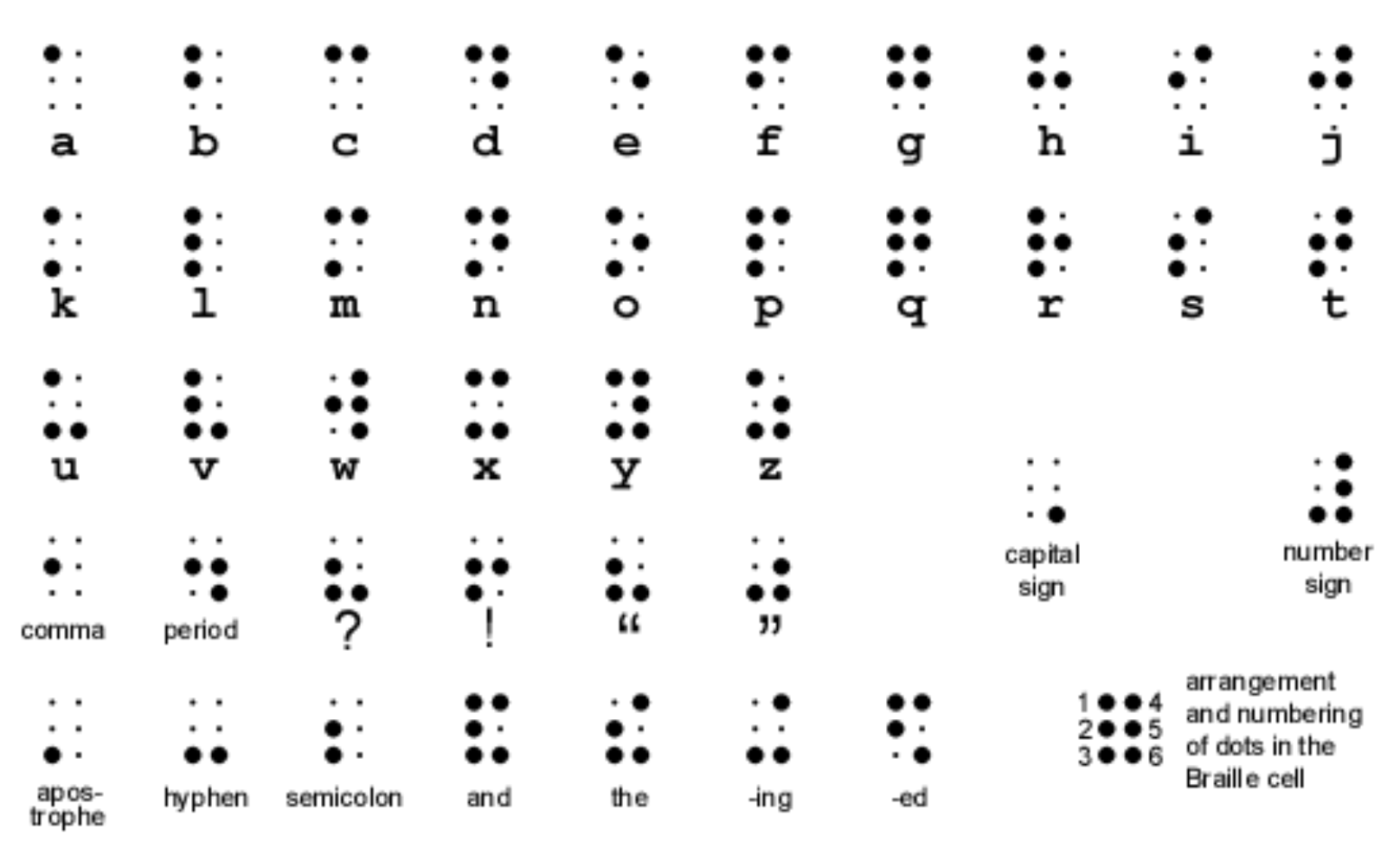
**Font “Courier New”**

**Font size “9”**

**Bold**

*[your program output here – one screenshot per run]\*\**

**2) [15 points]** You've been hired by *Braille Buddies* to write a Java console application that prints the equivalent Braille of the text entered by the user. Braille is a special language used by the blind to read text. It uses a 3x2 grid of dots to represent different characters. Here are the grid definitions for Grade 1 Braille:



To specify an upper case letter, the "capital sign" grid defined above precedes a lower case letter. Use the following three-dimensional array declaration to represent the Braille grid definitions. The first dimension is the letter, the second dimension is the row, and the third dimension is the column of a spot in a grid. A one (1) indicates a big spot while a zero (0) indicates a small spot:

final int[][][] BRAILLE\_GRIDS =

{

{{ 1, 0}, { 0, 0}, { 0, 0}}, // 'a' - 0

{{ 1, 0}, { 1, 0}, { 0, 0}}, // 'b' - 1

{{ 1, 1}, { 0, 0}, { 0, 0}}, // 'c' - 2

{{ 1, 1}, { 0, 1}, { 0, 0}}, // 'd' - 3

{{ 1, 0}, { 0, 1}, { 0, 0}}, // 'e' - 4

{{ 1, 1}, { 1, 0}, { 0, 0}}, // 'f' - 5

{{ 1, 1}, { 1, 1}, { 0, 0}}, // 'g' - 6

{{ 1, 0}, { 1, 1}, { 0, 0}}, // 'h' - 7

{{ 0, 1}, { 1, 0}, { 0, 0}}, // 'i' - 8

{{ 0, 1}, { 1, 1}, { 0, 0}}, // 'j' - 9

{{ 1, 0}, { 0, 0}, { 1, 0}}, // 'k' - 10

{{ 1, 0}, { 1, 0}, { 1, 0}}, // 'l' - 11

{{ 1, 1}, { 0, 0}, { 1, 0}}, // 'm' - 12

{{ 1, 1}, { 0, 1}, { 1, 0}}, // 'n' - 13

{{ 1, 0}, { 0, 1}, { 1, 0}}, // 'o' - 14

{{ 1, 1}, { 1, 0}, { 1, 0}}, // 'p' - 15

{{ 1, 1}, { 1, 1}, { 1, 0}}, // 'q' - 16

{{ 1, 0}, { 1, 1}, { 1, 0}}, // 'r' - 17

{{ 0, 1}, { 1, 0}, { 1, 0}}, // 's' - 18

{{ 0, 1}, { 1, 1}, { 1, 0}}, // 't' - 19

{{ 1, 0}, { 0, 0}, { 1, 1}}, // 'u' - 20

{{ 1, 0}, { 1, 0}, { 1, 1}}, // 'v' - 21

{{ 0, 1}, { 1, 1}, { 0, 1}}, // 'w' - 22

{{ 1, 1}, { 0, 0}, { 1, 1}}, // 'x' - 23

{{ 1, 1}, { 0, 1}, { 1, 1}}, // 'y' - 24

{{ 1, 0}, { 0, 1}, { 1, 1}}, // 'z' - 25

{{ 0, 0}, { 0, 0}, { 0, 1}} // upper case follows - 26

};

Use a validation loop to prompt for and get from the user a string that contains only upper and lower case letters and spaces. If an invalid string is entered, print an error message. If a valid string is entered, convert and display it in Braille. Print the following four lines for each string:

|  |  |
| --- | --- |
| Line | Output |
| Character | ● For an lower case letter, print the letter.  ● For an upper case letter, print "UP" and the corresponding lower case letter.  ● For the space character, print a space. |
| Rows 1-3 | ● For lower case letters, print its grid.  ● For upper case letters, prints two grids: the "capital sign" grid and the corresponding lower case letter grid.  ● For the space character, print spaces. |

To access the correct grid in array BRAILLE\_GRIDS, convert a lower case letter to value between 0 and 25 (the first dimension) using the fact that each character is assigned a unique integer. Here is a run with a sample string:

Enter a string (letters and spaces only) to convert to Braille (q to exit): Hello World

Character: UP h e l l o UP w o r l d

Row 1: 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 1 0 1 0 1 1

Row 2: 0 0 1 1 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1

Row 3: 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 0 1 0 0 0

Enter a string (letters and spaces only) to convert to Braille (q to exit):

Continue to prompt the user for strings until they enter “q”. Use these strings for the last three inputs:

|  |  |
| --- | --- |
| Run | Text |
| 1 | The quick Fox |
| 2 | Jumps over |
| 3 | the Lazy Dog |

*[your program code here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

*[your program output here – one screenshot per run]\*\**

\* **Copying-and-pasting application code to a Word document**

1) From the program editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

\*\* **Copying-and-pasting application output to a Word document**

1) From the Eclipse main screen, maximize the Console window.

2) From the Console window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.