**CS 2318 (Assembly Language, §253 & §254, Spring 2023)** *penalty if not filled-->* Name: Samuel Wait

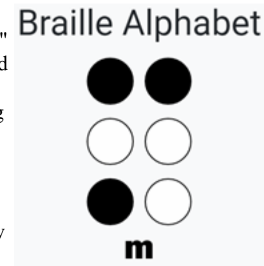
## Assignment 1 Part 1 (posted 01/26/2023, DUE as assigned on Canvas)

Score (by grader):

(Don't have to type but must be legible - **no credits for any parts I have difficulty reading/understanding**.)

1 (4 points)

In 1824, a 15-year-old French boy invented a system for representing text using combinations of flat and raised dots on paper so that the text could be read by touch. The system became very popular with visually impaired people as it provided a relatively fast and reliable way to "read" text without seeing it. Louis Braille's system has much in common with modern day digital data representation. There are only two states (raised and flat) per dot, and yet combinations of dots can be used to represent reference books and works of literature. Each character in Braille is represented with a matrix or cell of 6 dots. Each dot can either be raised or not raised. Different numbers and letters can be represented by using different patterns of raised and not raised dots.




Can you see that Braille is a representation using bits? Each dot can be in 1 of 2 different states (raised and not raised) and sequences (ordered arrangements) of these are used to represent distinct patterns. The letter m, for example, could be written as 110010, where "1" means raised dot, and "0" means not raised dot and we read orderly from left to right and then down. This is the same as how we often use 0's and 1's to show the way a typical modern day computer is representing data.

***Question 1a***: How many distinct patterns can the 6 dots of a Braille cell give? (Show how you get your answer to not risk incurring penalty.) (Don't discount the all-flat case.)

There are 6 cells in the matrix that can be in two different states, which can make $\boxed{2^6 = 64 \text{ distinct patterns}}$

Braille also illustrates why binary representation is so popular. It would be possible to have five kinds of dot: flat, ¼-raised, ½-raised, ¾-raised, and fully-raised. A skilled Braille reader could distinguish them.

***Question 1b***: With five states per dot, what (minimum) number of dots would be needed per cell to support a number of distinct patterns not less than the number involved in Braille? (Note: Important to have the latter number correctly determined in ***Question 1a***. And show how you get your answer to not risk incurring penalty.)

If there are 5 states per dot in a matrix of 6 dots, then there are $\boxed{5^6 = 15{,}625 \text{ distinct patterns}}$

Why is using five states per dot to cover the same number of distinct patterns not taken up? The trouble is that you would need more accurate devices to create the dots, and people would need to be much more acute at sensing them. If a page was squashed, even slightly, it could leave the information unreadable.

Modern day finite-state devices almost always use two states (binary) for similar reasons: computer memory (storage) devices can be made cheaper, smaller, more reliable, … if the finite-state devices they use only need to be able to distinguish between two states, commonly as high-voltage and low-voltage, rather than fine-grained distinctions between very subtle differences in voltages. Using ten states (to correspond to the ten digits used in our every day decimal counting system) would be much more challenging and costly.

2 (4 points) You are to assign each of 1,500,000 (1½ million) individuals with a distinct (unique) *k*-character pattern/string involving the digits (`'0'` through `'9'`), select uppercase letters (`'A'` through `'Z'` excluding `'I'`, `'O'`, `'S'` and `'Z'`) and special characters `'@'`, `'#'` and `'_'`, with the provision that the first (leftmost) of the *k*-character must not be one of the digits (`'0'` through `'9'`). What is the minimum *k* that you will need?

CAUTION: You will earn NO CREDITS if you simply indicate what minimum *k* is and not *clearly show working* (i.e., how you arrive at the minimum *k*).

TIP: One can use Microsoft Excel to help evaluate expressions involving multiplication and number raised to some power. For example, to find the value of 56 x $123^4$, type the formula `=56*123^4` into an Excel Worksheet cell and press Enter.

There are 10 digits, 3 special characters, and 22 alphabetic characters. The total number of options for characters is 35.
The total number of options for the leftmost character excludes digits is 35−10 = 25.

Therefore, $k=1 \Rightarrow 25 \ngeq 1{,}500{,}000$

$k=2 \Rightarrow 25(35) = 875 \ngeq 1{,}500{,}000$

$k=3 \Rightarrow 25(35)^2 = 30{,}625 \ngeq 1{,}500{,}000$

$k=4 \Rightarrow 25(35)^3 = 1{,}071{,}875 \ngeq 1{,}500{,}000$

$k=5 \Rightarrow 25(35)^4 = 37{,}515{,}625 \geq 1{,}500{,}000$

∴ The minimum k needed to assign 1,500,000 individuals a unique string of the specified characters and digits is $k=5 \Rightarrow 25(35)^4 = 37{,}515{,}625$

3 (6 points) Using *Horner's scheme*, evaluate the *decimal* values of (a) *binary unsigned whole number* `10111010` and (b) *base-9 unsigned whole number* `2318`.

CAUTION: You will earn NO CREDITS if you don't use *Horner's scheme* to more efficiently evaluate polynomial.
(In particular, you will get a **0** if you resort to evaluating the polynomial directly "brute force".)
You will earn NO CREDITS if you simply show the final result and not *clearly show working* (i.e., intermediate steps).

a) $(10111010)_2 = (((((((1\times2+0)\times2+1)\times2+1)\times2+1)\times2+0)\times2+1)\times2+0) \boxed{=(186)_{10}}$

b) $(2318)_9$

$\Rightarrow$ 2000, 300, 10, 8 $\Rightarrow$

$9\times2+3=21$

$9\times21+1=190$

$9\times190+8=\boxed{(1718)_{10}}$

4 (6 points) First use the *repeated division method* to represent the *decimal unsigned whole number* `467` in *binary*, then re-write the result obtained (*decimal unsigned whole number* `467` in *binary*) more compactly in hex.

CAUTION: For the first part, NO CREDITS if you simply show the final result and not *clearly show working* (i.e., intermediate steps).

CAUTION: For the re-writing part, *clearly show working* (i.e., how you group the bits) to not risk incurring penalty, and NO CREDITS for answer obtained from doing repeated division using another base like 16.

$(467)_{10} \Rightarrow$

2 | 467
2 | 233 R 1
2 | 116 R 1
2 | 58 R 0
2 | 29 R 0
2 | 14 R 1
2 | 7 R 0
2 | 3 R 1
2 | 1 R 1
0 R 1

$\Rightarrow \boxed{(467)_{10} = (111010011)_2}$

$(111010011)_2 = (\underbrace{0001}_{1}\ \underbrace{1101}_{D}\ \underbrace{0011}_{3})_2 \Rightarrow \boxed{(111010011)_2 = (1D3)_{16}}$

$\boxed{\therefore (467)_{10} = (111010011)_2 = (1D3)_{16}}$