Convention used: In the remainder, if a decimal-looking number has no subscript to its right it will mean that the number is indeed a decimal number, i.e. in base-10 or also noted radix-10.

Problem 1.
(a) How would you write 1024 bytes, or 2048 bytes in correct kilobyte format?
(b) What does 1MB mean ? What does 1MB if it refers to main memory mean? What does 1MB if it refers to disk-drive space mean? How about 1Mb ?
(c) How many bytes in 3KiB?

Problem 2.
Write down in binary, hexadecimal and octal the unsigned integer (natural number) 31 and 32.

Problem 3.
Write `120 , 63` in 8-bit unsigned binary (notation). Write `120 , 63` in 16-bit unsigned binary (notation).

Problem 4.
What is the octal and hexadecimal representation of `120, 63` ?

Problem 5.
What is the decimal representation of `10001111` if represented as an unsigned integer.

Problem 6.
What is the octal representation of unsigned binary integer `10001111`. What is its hexadecimal representation?

Problem 7.
What is the difference between `clear` and `clc` in MATLAB?

Problem 8.
What is the default data type of a number in `MATLAB`? How many bytes in it?

Problem 9.
For the following piece of MATLAB code, what is the data type of x, y, z. How many bytes?

```
x= int8(10);
y= 20;
z= uint32(10);
```

Solutions

Problem 1.

(a) We could write `1KiB` to denote 1024 bytes. Thus `1KiB` can only be used based on what we learnt in class. One should avoid writing `1kB` because the little-kay in `1kB` might get confused with a kilo and `1 kilo` is `1000`, not `1024`. For 2048 bytes we could write `2KiB` (or `2KKB` noting the earlier remark). (If you read computer articles however, `1kB` implies 1024B more often than not, and it's unlikely to denot 1000B.)

(b) 1MB. If it refers to disk-drive capacity it means 1,000,000B. If however it refers to main memory it means $2^{20} = 1,048,576$B. (One should have written then `1MiB`.) Luckily few (or no) machines nowadays use "megabytes" of (RAM) memory. The `1Mb` does not denote anything related to "bit". It means nothing. Avoid using a `b`; some people might misinterpret it for a `bit` or a `byte`. Use `bit` if you mean "bit".

Note. Go to `https://calmail.berkeley.edu/docs/quota-1gib.html` for an interesting use of `1GiB` i.e. 2^{30}B.

(c) `3KiB` is 3×1024 bytes, i.e. 3072 bytes.

Problem 2.

31 i.e. 31_{10} (in base 10) is in binary `11111`. Thus in octal, if we group the bits in triplets starting from the right `11111` becomes `11'111`, and this is `37` in octal since `111` in binary is 7 in octal (and also decimal), and `11` is 3. Thus $31_{10} = 37_8$.

For hexadecimal, we group in quadruplets, i.e. `11111` becomes `1'1111` and thus tt 1F, since `1111` is 15 (decimal) or `F` (hexadecimal). Thus $31_{10} = 37_8 = 1F_{16}$. A formal way to rewrite the latter is by saying $31_{10} = 0o37 = 0x1F$. The `0o` prefix signify that the number that immediately follows is in octal; similarly `0x` is the hexadecimal prefix.

In summary: $31_{10} = 11111_2 = 0o37 = 0x1F$.

For 32 we have

Representation of 32: $32_{10} = 100000_2 = 0o40 = 0x20$.

Problem 3.

Unsigned integer 120 in 8-bit binary is $\texttt{01111000}_2$. Similarly 63 is $\texttt{00111111}_2$.

In 16-bit notation we get some leading zeroes added respectively. Thus 120 is $\texttt{0000000001111000}_2$ and 63 is $\texttt{0000000000111111}_2$. But we prefer to write instead $\texttt{0000 0000 0111 1000}_2$ and $\texttt{0000 0000 0011 1111}_2$, by adding a space after every four bits.

Problem 4.

For the octal representation of `120, 63` we use the results of the previous problem.

Given that 120 is $\texttt{01111000}_2$, we can group it right to left into groups of three as follows: `01'111'000` or by adding a leading zero to the leftmost group we get `001'111'000` which is in octal $\texttt{170}_8$ or `0o170`. Similarly 63 is $\texttt{00111111}_2$ or `00'111'111` or `000'111'111` i.e. $\texttt{077}_8$, which can be written $\texttt{77}_8$, or `0o77` since the leading zero of `077` can be omitted. (Note that sometimes before the introduction of `0o` we could write down an octal number by preceding it with a zero. Thus `077` denoted an octal number; this is deprecated and replaced by the `0o77`.)

For the hexadecimal representation of `120, 63` we similarly get the following.

For 120, we get by groupping `0111'1000` that is $\texttt{78}_{16}$ i.e. `0x78`. For 63, i.e. `0011 1111` we get $\texttt{3f}_{16}$ i.e. `0x3f`.

Problem 5.

The unsigned integer `1000 1111` is $2^7 + 2^3 + 2^2 + 2^1 + 2^0$ which is `143`.

Problem 6.
Binary `10001111` in hexadecimal is `1000'1111 = 8F = 0x8F`. Binary `10001111` in octal is `10'001'111 = 217 = 0o217`.

Problem 7.
clear clears the variables in the workspace of the user. **clc** justs clears the screen and moves the cursor to the top of the command window.

Problem 8.
The default data type is **double** and there are 8 bytes in a `double`.

Problem 9.
The data type of `x` is `int8` and it is 1B.
The data type of `y` is `double` and it is 8B.
The data type of `z` is `uint32` and it is 4B.