Sheet 2 - Basic Data Types

Assignment 1

a)

Signed types can represent both positive and negative values, whereas unsigned types can only represent positive values (and zero).

There are three well known methods for representing negative values in binary:
Signed magnitude. This is the easiest to understand, because it works the same as we are used to when dealing with negative decimal values: The first position (bit) represents the sign (0 for positive, 1 for negative), and the other bits represent the number. Although it is easy for us to understand, it is hard for computers to work with, especially when doing arithmetic with negative numbers.
In 8-bit signed magnitude, the value 8 is represented as 0 0001000 and -8 as 1 0001000.

One's complement. In this representation, negative numbers are created from the corresponding positive number by flipping all the bits and not just the sign bit. This makes it easier to work with negative numbers for a computer, but has the complication that there are two distinct representations for +0 and -0. The flipping of all the bits makes this harder to understand for humans.
In 8-bit one's complement, the value 8 is represented as 00001000 and -8 as 11110111.

Two's complement. This is the most common representation used nowadays for negative integers because it is the easiest to work with for computers, but it is also the hardest to understand for humans. When comparing the bit patterns used for negative values between one's complement and two's complement, it can be observed that the same bit pattern in two's complement encodes for the next lower number. For example 11111111 stands for -0 in one's complement and for -1 in two's complement, and similarly for 10000000 (-127 vs -128).
In 8-bit two's complement, the value 8 is represented as 00001000 and -8 as 11111000.

b)

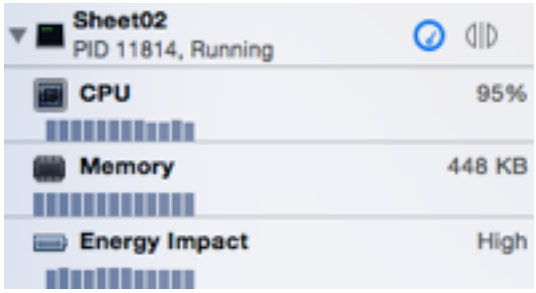
Name	Description	Size*	Range*
`char`	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
`short int (short)`	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
`int`	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`long int (long)`	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`bool`	Boolean value. It can take one of two values: true or false.	1byte	true or false
`float`	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
`double`	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`long double`	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`wchar_t`	Wide character.	2 *or* 4 bytes	1 wide character

c)

```
//////////////Assignment 1 ////////////////

Listing1:
-24

Listing2:
```



listing1 gets the result of -24, it can gets out the loop.
listing 2, the loop runs forever, never gets out.