For the problems in the first part, no submission is required.

1] Write a C program that uses the characters space and hashtag to print on the screen the first letter of your (last) name. The exact size and shape of the letter is up to you, but it should be at least 6 lines tall. For example, an A could be shown like this:

#

# #

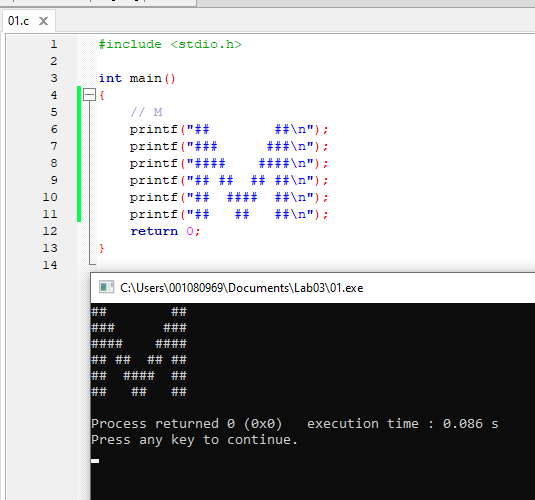
# #

########

# #

# #

* Hint: Use multiple *printf* statements.



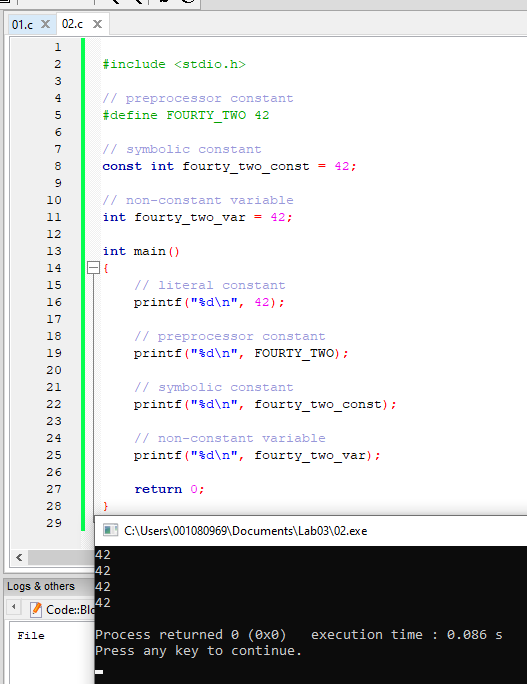
2] Write a program to print the constant integer 42 in three ways, with three separate *printf* statements:

• Using a literal constant.

• Using a symbolic preprocessor constant (with #define)

• Using a symbolic constant declared with the keyword *const*

Additionally, declare and print 42 as a (non-constant) variable.



3] In class, we noted that, for signed integers, the positive and negative ranges (Table 3.2) are not quite equal - the negative range is in reality one unit longer, i.e. all the 7s are actually 8s.

Table

Description automatically generated

Let us verify this for signed short integers. Write a C program in which you:

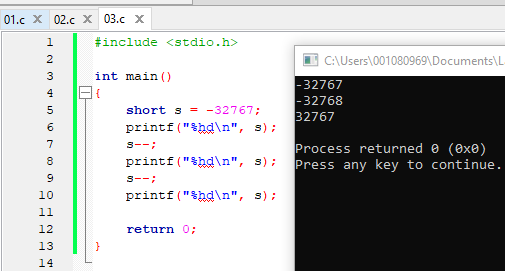
* declare one variable **s** of type **short** (remember that the keywords *signed* and *int* are optional!)
* initialize **s** with -32767
* print **s** (Hint: Use the format specifier **%hd** for short – do you remember what h stands for?)
* decrement **s** and print it again. What happened?

A: The variable s decreased to a value of -32768.

* decrement **s** again and print it. Explain what happened:

A: The variable s wrapped around to positive 32767, because it underflowed.

For amusement, check out this xkcd comic: <https://xkcd.com/571/>



4] Print the *size* of the *short* data type in three ways, using the **sizeof** operator:

* by applying sizeof to a variable **s**
* by applying sizeof t the type **short** itself
* same as above, but use the full name **signed short int**.

Now do the same for the unsigned version of the type, **unsigned short**.

Give two reasons why **sizeof** is an operator and not a function:

The sizeof operator works on types so it cannot be a function.

The parenthesis on sizeof are not required which are for a function.

Quacks like a duck and walks like a duck, but is not a duck!

**To submit for lab report**

1] Write a C program that uses the characters space and hashtag to print on the screen the first two letters of your (last) name. The exact size and shape of the letter is up to you, but it should be at least 6 lines tall. For example, AB could be shown like this:

# ##

# # # #

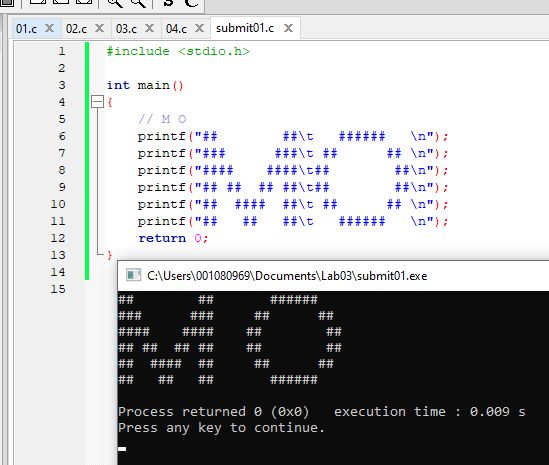
# # # #

######## # ##

# # # #

# # ###

* Hint: To align properly, you may find the *tab* escape character \t useful.



2] Make a program that will declare, and initialize the next data (select proper names):

Integer for the age of a patient initialized with 8.

Constant integer for number of students, with value 12.

The same, but using pre-processor directive. Use another name. The value must be 14

Real number (floating point) with double precision, for the distance between two cities. Initialize with 15.15.

Display all numbers using printf. Remember you can use %d (for integers) and %lf (for doubles) in the format.

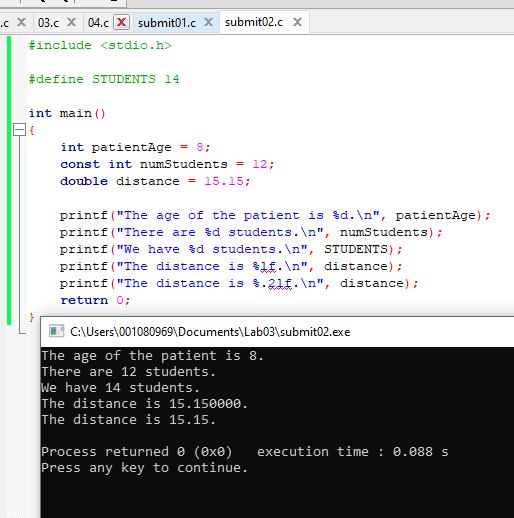
Expected output:

**The age of the patient is 8.**

**There are 12 students.**

**We have 14 students.**

**The distance is 15.15 (don't worry if more digits appear)**

  
I added the extra line to round to 2 decimal places.