Week 07, Homework 06	Weight: 3%	Due: Monday week 9, 09:00am (via `sync`)

Pre-homework Preparation:

- **Lab: Week 05, Lab 05**
- **Lectures: Weeks 01, 02, 03, 04, 05**

Homework Activities:

Obtain the homework files using **`sync`**. For each question's source file, be sure to fill in the file comment header accurately!

Question 1: Repetition, a known number of times

In **`hw06q01.c`** write a function called **`message_printer(int repeat_count)`** that will print the text **`--Message Number x`**, where x is replaced with the value 1 through to **`repeat_count`**. The program must query the user for how many times they would like the message repeated, then call **`message_printer`** based upon their request. All output text with -- is to be sent from the **`message_printer`** function.

The program must output as follows:

```
Welcome to Question 6! (Choose 0 to quit!)
How many times would you like to repeat the message? 3
--Called: "message_printer" function...
--Message Number 1
--Message Number 2
--Message Number 3
--Done! 3 messages printed.
How many times would you like to repeat the message? 2
--Called: "message_printer" function...
--Message Number 1
--Message Number 2
--Done! 2 messages printed.
How many times would you like to repeat the message? 1
--Called: "message_printer" function...
--Message Number 1
--Done! 1 message printed.
How many times would you like to repeat the message? 0
Programming finishing!!! Goodbye!
```

If the user selects 0 the program must not call **`message_printer`**, and instead the program must finish. Also note, that when the **`message_printer`** function only repeats the message once, the **`--Done!`** outputs the text **`message`**, singular, in contrast to when it is called to repeat more than once where the **`messages`** is plural.

Question 2: Nested for loops

In **hw06q02.c** the **main** function calls other functions to output the following shapes.

```
1) Square:
#####
#####
#####
#####
#####

2) Triangle:
#
##
###
####
#####

3) Inverted Triangle:
#####
####
###
##
#

4) Right-Aligned Triangle:
    #
   ##
  ###
 ####
#####

5) Inverted Right-Aligned Triangle:
#####
 ####
  ###
   ##
    #
```

To complete the output of each shape you will need to use nested for loops. You may only call **printf** via the following functions, which you must also implement:

- **print_hash()**
- **print_newline()**
- **print_single_space()**

Do not call any other functions to output to the console, you may only use the three print functions outlined above.

Question 3: Boolean Logic: NAND, NOR, XOR, XNOR

Recall the truth tables for AND, OR and NOT:

a	b	a AND b	a OR b	NOT a	NOT b
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	0
1	1	1	1	0	1

The corresponding operators in C are **`&&`**, **`||`** and **`!`**.

There are other Boolean logic operators which do not have corresponding C operators - however you can write your own implementation of them!

The truth tables for NAND, NOR, XOR and XNOR are as follows:

a	b	a NAND b	a NOR b	a XOR b	a XNOR b
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

In **`hw06q03.c`** complete the bodies of the following functions:

- **`int nand_gate(int a, int b)`** which returns an integer represented the NAND operation of a NAND b.
- **`int nor_gate(int a, int b)`** which returns an integer represented the NOR operation of a NOR b.
- **`int xor_gate(int a, int b)`** which returns an integer represented the XOR operation of a XOR b.
- **`int xnor_gate(int a, int b)`** which returns an integer represented the XNOR operation of a XNOR b.

The main function has already been written for you. It will call each of the logic gate functions, and print out the results. Once your versions of the logic gate functions are implemented, the program's output should yield the same results as the truth tables.

Question 4: Strings are null-terminated arrays of chars

In **`hw06q04.c`** write the following functions, each of which take a string parameter.

```
char get_char_at(char string[], int index)
```

This function should return the character at a particular location in the string.

```
int get_string_length(char string[])
```

This function should return the length of the string. Do not use the standard library function strlen() to implement this function! Do it yourself, remembering that in C strings are null terminated.

```
void remove_last_char(char string[])
```

This function should make the string one character shorter.

```
int is_palindrome(char string[])
```

This function should return true if and only if its argument is a palindrome, i.e. a string that reads the same forwards and backwards. For example, 'refer', 'civic', 'rotator' are all palindromes.

You can define your own **`main()`** function to help you test these functions. You should also run

```
make test
```

…and ensure that all tests pass.

Question 5: Set pixel

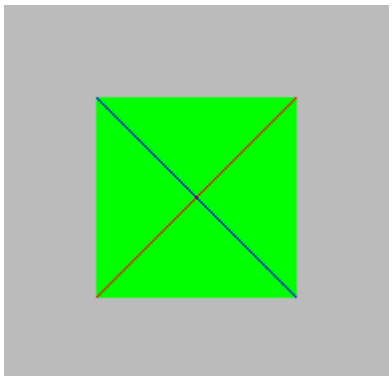
In P1 image world, images are represented using arrays of unsigned chars (bytes). However, these one-dimensional arrays (*one-dimensional* - using only one number to represent location, i.e. the offset from the start of the array) actually represent a two-dimensional rectangular area on screen. It is more natural to want to access this array using two numbers, an *x* and a *y* coordinate.

Define the following useful function to set the colour values of a pixel at the stated x and y coordinates in the image with handle target.

void set_pixel(int target, int x, int y, int a, int r, int g, int b)

To help you remember that the offset in the pixel array is given by a formula involving the x and y coordinates, the width of the world in pixels, and the number of bytes per pixel (4).

Then, noting that 0 is the handle of the current screen, produce a close approximation to the following image *without creating any additional images using the* **`create_image()`** *function*.



The square in the image is 100 x 100 pixels and should be centred on the screen. You should define additional functions to help structure your algorithm.

Question 6: Copy and paste

It is often helpful to be able to copy a rectangular area from one image and create a new image using it. Implement this in a function with the following signature:

int copy(int source, int x, int y, int width, int height)

The return value of the function should be the handle of the newly created image, or -1 if the coordinates given are outside the bounds of the source image.

The file **`meerkat.bmp`** contains an image of a cute animal, which I am pretty sure is a meerkat, although the visit to the zoo was rather a while ago. You can view it using the command **`fbi`** or write a program to load and display it yourself.

In **`hw06q06.c`** write a program that loads the image and copies a rectangle featuring just the meerkat's face. Generate an image such as the following with copies of the face placed at random:

Question 7: Image manipulation

The image **`boats.bmp`** is attractive but a greyscale (black and white) image would suit the subject better. In hw06q06.c write a program that loads boats.bmp and displays it in greyscale. As part of your program you should write a function with the following signature that will make a copy of a source image, identified by its handle, but in greyscale.

int to_greyscale(int source)

The tricky part of this question is determining what level of grey should correspond to what colour pixel.

Greys are achieved by setting the values for red, green and blue for a pixel to all the same value, varying between 0x000000 (black) and 0xffffff (white).

There are a number of different ways to map a coloured pixel to a greyscale one. Do some research online and carefully document any sources of information that you use in your question header, as well as using comments in your code to explain what your algorithm is doing.

Question 8: Find and replace

It is sometimes useful to be able to replace every instance of one colour with another in an image.

The image **`bigben.bmp`** features a daytime image of Big Ben (part of the Palace of Westminster, home of the UK parliament, in London) against a totally uniform blue sky. In hw6q08.c write a program that instead displays Big Ben at night. As part of this you should implement the following functions.

void replace(int target, int fa, int fr, int fg, int fb, int ra, int rr, int rg, int rb)

One of the challenges in this question is working out which colour should be replaced.

For full marks add randomly placed 'stars' (single white pixels are fine) to the night sky, taking care that these do not appear in front of the image of the building. There is a 'clever' way to do this…

Question 9: Advanced find and replace

The image **`rabbit.bmp`** features an evil looking bunny, who is suffering from serious redeye. This phenomenon is caused when the light of a camera's flash reflects off an eye's retina. It can be corrected to produce more normal looking pictures by replacing red pixels with black ones.

The problem is that usually the red pixels do not have identically the same colour values. So it is necessary to replace all pixels with colour values *near* to a target colour.

In **`hw06q09.c`** write a program to display a more normal looking rabbit with redeye fixed. As part of this you should implement the following function.

void close_replace(int target, int fa, int fr, int fg, int fb, int ra, int rr, int rg, int rb, float tolerance)

One way of determining 'closeness' between two pixels is to treat the values of red, green and blue of each as coordinates in 3D space, and then determine the straight-line distance between these two points.

Homework Submission:

Run the `sync` command to submit your completed homework.

Marking Criteria:

Have you completed each of the following?

Marking Criteria:	Week 07, Homework 06 Weight 3%	Maximum Possible Mark:	Mark achieved?
Q1:	Code structured as described in the question	4	
	Reliably produces the correct output for valid inputs	4	
	Code well formatted, named and commented	2	
Q2:	Code structured as described in the question	4	
	Produces the correct output	4	
	Code well formatted, named and commented	2	
Q3:	Code structured as described in the question	4	
	Produces the correct output	4	
	Code well formatted, named and commented	2	
Q4:	Code structured as described in the question	4	
	All tests pass	4	
	Code well formatted, named and commented	2	
Q5:	Code structured as described in the question	4	
	Produces the correct output	4	
	Code well formatted, named and commented	2	
Q6:	Code structured as described in the question	4	
	Produces the correct output	4	
	Code well formatted, named and commented	2	
Q7:	Code structured as described in the question	4	
	Produces a greyscale image	4	
	Code well formatted, named and algorithm clearly presented	2	
Q8:	Code structured as described in the question	4	
	Big ben displayed on black background	2	
	Image also includes randomly placed stars	2	
	Image overlayed on starry background	2	
Q9:	Code structured as described in the question	4	
	Redeye fixed with limited harm to rest of image	4	
	Code well formatted, named and algorithm clearly presented	2	
Total:		**90**	