CMPT125, Spring 2023
Lab exam

Wednesday, March 22, 2023, 2:30pm-3:20pm

You need to implement the functions in ***labexam.c.***
Submit only the ***.c*** file to Coursys
Coursys Assignment - **Lab Exam D205-D206 Wed 2:30pm**

You have 50 minutes to solve all 3 problems.
The maximal score is 20 points.

The exam will be graded both **automatically** and by **reading your code**.
You can run your code using

```
>> make
>> ./run_test
```

Correctness: Make sure that your code compiles without warnings/errors,
and works as expected.

Readability: Your code should be readable. Add comments wherever necessary.
If needed, write helper functions to break the code into small, readable chunks.

Compilation: Your code MUST compile in CSIL with the Makefile provided.
If the code does not compile in CSIL, the grade on the assignment is 0 (zero).
Even if you can't solve a problem, make sure it compiles.

Helper functions: If necessary, you may add helper functions to the .c file.

main() function: do not add main(). Adding main() will cause compilation errors, as the main() function is already in the test file.

Using printf()/scanf(): Your function should not have any unnecessary printf() statements.
They may interfere with the automatic graders.

Warnings: Warnings during compilation will reduce points.
More importantly, they indicate that something is probably wrong with the code.

Testing: An example of a test file is included.
Your code will be tested using the provided tests as well as additional tests.
You are *strongly encouraged to write more tests* to check your solution is correct, but you don't need to submit them.

Good luck!

Question 1 [7 points]

Write a function that gets a string, and returns the most frequent char in the string, and the number of times it appears in the string. In case of a tie, you may return any of the most frequent chars. Use the struct char_int defined in labexam.h to return the values.

```
// gets a string, and finds the most frequent char in the string
// returns the most frequent char and the number of times it appears
// in case of a tie, any of the most common chars will be accepted
char_int most_frequent(const char* str);
```

Question 2 [6 points]

Write a function that gets an array of ints of length n>0, and a predicate pred. It applies pred on each element. If pred(ar[i]) is true, it remains unchanged, and if pred(ar[i]) is false, then it is changed to zero. The function returns the number of changed entries. For example:

- *Suppose we have ar = [1,2,0,3,8,7], and the predicate is* `is_even(int x)`
- *When applying filter_to_zero(ar, 6, is_even), ar becomes [0,2,0,0,8,0], and the function returns 3.*

```
// the function gets an array of ints of length n, and a predicate
// for each entry a[i]: if pred(ar[i]) is true, it remains unchanged
// and if pred(ar[i]) is false, then it is changed to zero
// the function returns the number of changed entries
int filter_to_zero(int* ar, int n, bool(*pred)(int));
```

Question 3 [7 points]

Write a function that gets ar1 and ar2, two arrays of ints of length n, and returns a new array SUM, where SUM[i] = ar1[i] + ar2[i].
For example:

- *On input [1,2,3,4,5] and [0,2,-3,4,10] the function returns [1,4,0,8,15].*

```
// gets ar1 and ar2 of length n
// returns a new array SUM of length n, where SUM[i] = ar1[i] + ar2[i]
int* sum_arrays(const int* ar1, const int* ar2, int n);
```