CSC 152 programming assignment: contains

Programming Assignment

Write a function `contains` that takes two buffers as input and returns a non-zero int if and only if the second buffer is contained in the first. In other words, non-zero should be returned only if there is a contiguous sequence of bytes in the first buffer that matches exactly the entire second buffer.

What to do about an empty second buffer? Similar to the fact that an empty set is a subset of every set, the empty buffer is contained in every buffer.

The file you submit should be called `contains.c` and should have a function with the following signature inside.

```
// a points to an alen byte buffer, b points to a blen byte buffer,
// Returns non-zero iff b's buffer is found contiguously inside a's.
int contains(void *a, int alen, void *b, int blen) {

}
```

Your file must not have a `main` function and must not emit any warnings when compiled with `-Wall`.

Design Advice

Your instinct might be to use nested for-loops, but nested for-loops often are confusing. To avoid this, I usually have my outer for-loop call a function and have that function do a well-defined task (which may include a for-loop). By separating the outer loop from what the inner loop is doing, it's easier for me to keep things straight. In pseudocode, here's what I mean.

```
int contains(a, alen, b, blen)
    for i = each byte index in a where b could begin
        if is_equal(&a[i], b, blen)
            return 1
    return 0
```

That's a lot clearer than a nested for-loop would be.

Testing and Grading

Programs in this class are tested against test cases. If your program behaves as the test case expects (ie, according to spec) then credit for the test case is given. Otherwise no credit is given for the test case. This credit/no-credit grading means that it is very important that you test your code well before submission.

You may assume that the inputs are error free. This means neither alen nor blen will be negative and that a and b point to correctly allocated buffers of length alen and blen bytes.

Testing is a little like a game. You should try to think of all sorts of weird inputs that are legal because they don't violate the specification and verify that your code does the right thing for them. Any ambiguities in the specification should be asked about well before the due date. For example, nowhere in the spec does it say that alen ≥ blen, so make sure that your function works correctly when alen < blen. Also, does alen and/or blen equal to zero violate the spec? If not, test that those cases work as expected. Throughout this class you will be expected to put some thought into test cases, ask questions, and test your code thoroughly.

The easiest test setup for this assignment is something like this.

```c
#include <stdio.h>

int contains(void *a, int alen, void *b, int blen) {

}

int main(int argc, char *argv[]) {

}
```

Your main can then contain your test code. Once you're satisfied and ready to submit, either delete or comment-out the main function.

Submission

1. The file you turn in should have a small comment block at the top containing your name, the date, anyone you worked with, and the URL of any online source that provided significant help to you. It is an important habit to give credit to important sources of help. For example:

```c
// contains.c by Ted Krovetz. Submitted for CSC 152 July 4, 1776.
// Pair-programmed with Ellen Watermellon
// Used the "reverse" code from https://www.techiedelight.com/reverse-a-c-string
```

2. I will compile your code with gcc/clang flags `-Wall -Werror -fsanitize=address`. Wall "enables all the warnings about constructions that some users consider questionable, and that are easy to avoid". Werror turns all warnings into errors which prevent successful compilation. Fsanitize=address causes runtime checks to detect some (but maybe not all) illegal memory accesses. You should run some tests with these settings as well to ensure maximum credit. (On campus, if fsanitize does not wrk with gcc, try clang instead.)
3. Follow the directions at https://krovetz.net/152/fileinbox.html very carefully to submit your file. Remember your file should not have a main because I will use my main to test.

Collaboration

You may collaborate with *one or two* other students on this homework if you wish, or work alone. Collaboration must be true collaboration however, which means that the work put into each problem should be roughly equal and all parties should come away understanding the solution. Here are some suggested ways of collaborating on the programming part.

- Pair programming. Two or three of you look at the same screen and only one of you operate the keyboard. The one at the keyboard is the "driver" and the other is the "navigator". The driver explains everything they are doing as they do it and the navigator asks questions and makes suggestions. If the navigator knows how to solve the problem and the driver does not, the navigator should not dictate solutions to the driver but instead should tutor the driver to help them understand. The driver and navigator should switch roles every 10-15 minutes. Problems solved this way can then be individually submitted.
- Code review. The members of the collaborative each try to solve the problem independently. They then take turns analyzing each other's code, asking questions trying to understand each other's algorithms and suggesting improvements. After the code reviews, each of you can then fix your code using what you learned from the reviews. Do not copy code. If the result of code review is that your code needs changes to be more like your partner's, do not copy it. Instead recreate your own variant without looking at your partner's.

The goal is to learn enough from one another so that you each can do similar problems independently in an exam situation.

If you want a collaborator but don't know people in the class, you can ask on Discord and/or use the group-finding post on Piazza.