CSCI 2271 Computer Systems
Assignment 3: Pointers
Due Friday, February 12

1. Download the file *patternMatch.c* from the course website. This file contains part of a program to do pattern matching. The function *main* looks like this:

```
int main() {
  char text[40], pattern[40], *position;
  int textlength, patternlength;

  printf("Enter text: ");
  textlength = readline(text, 40);
  printf("Enter pattern: ");
  patternlength = readline(pattern, 40);
  position = findmatch(pattern, text, patternlength, textlength);
  printmessage(position, text, patternlength, textlength);
}
```

The function *readline* should read a line of characters from the keyboard. The second argument to the function is the maximum number of characters to read. If a line has more characters than that, the function should ignore the rest of the characters on the line. The function returns the number of characters read. Note that the array should not contain the '\n' from the end of the line, nor should it include a terminator character such as '\0'. (In other words, if you happen to be familiar with how C implements strings, forget about it. The text and pattern arrays are NOT strings.)

The program calls *readline* twice. The first line its reads is called the *text*, and the second line is called the *pattern*. The program calls the function *findmatch* to determine if the pattern matches some portion of the text. If it does, then the function returns a pointer to the location in the text where the match begins. If it does not, then the function returns the pointer NULL.

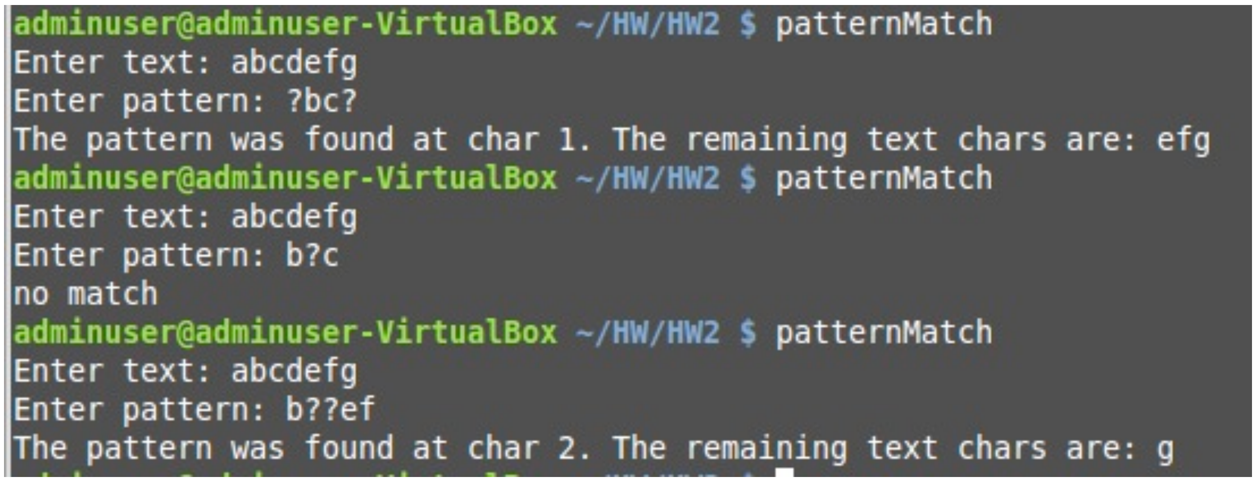
Finally, the program calls the function *printmessage* to print a message about what happened. If a match was not found, then a "no match" message is printed. If a match is found, then the function prints the position in the text where the match begins, and prints the remaining text chars that appear after the match ends. See the example below for details.

Your job is to write these three functions. You must write them exactly as described -- for example, *findmatch* must return a pointer to a char. You are also welcome to write other functions you deem appropriate. In all cases, your functions must not use any array notation -- everything must be done using pointers and pointer arithmetic.

When processing a pattern, you should treat the character '?' as a "wild card" symbol, which matches any text character. So for example, the pattern "?b??" matches the text "abcd" and "bbaa", but not "babb" or "bbb" or "????".

Use the following brute-force algorithm to find a match. First, compare the pattern against the text starting with the first char of the text. If that doesn't work, compare the pattern against the text starting with the second char of the text, and so on. If you get to the case where the end of the pattern goes beyond the end of the text, then you know that there is no match.

Here is an example of my program in action:



```
adminuser@adminuser-VirtualBox ~/HW/HW2 $ patternMatch
Enter text: abcdefg
Enter pattern: ?bc?
The pattern was found at char 1. The remaining text chars are: efg
adminuser@adminuser-VirtualBox ~/HW/HW2 $ patternMatch
Enter text: abcdefg
Enter pattern: b?c
no match
adminuser@adminuser-VirtualBox ~/HW/HW2 $ patternMatch
Enter text: abcdefg
Enter pattern: b??ef
The pattern was found at char 2. The remaining text chars are: g
```

When you are finished, submit your revised *patternMatch.c* to Canvas.