CSCI 2271 Computer Systems

Assignment 3: Pointers
Due Friday, February 8

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

```
int main() {
  int text[40], pattern[40], *position;
  int textlen, patternlen;

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

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, forget about how C implements 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.

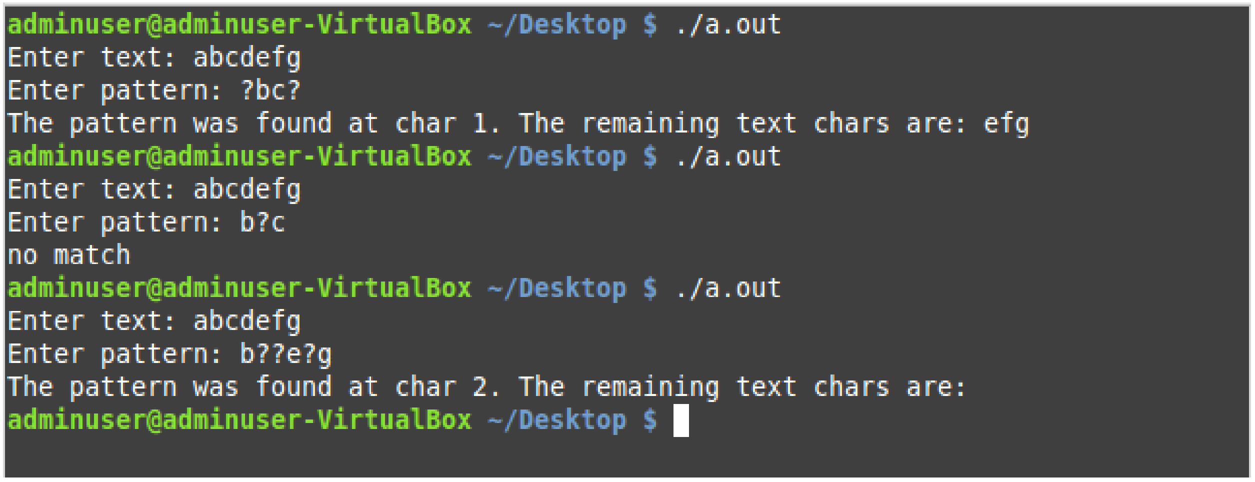
Finally, the program calls the function *printmessage* to print a message about what happened. If a match was not found, then a "no match" message is printed. If a match is found, then the function prints the position in the text where the match begins, and prints the remaining text characters that appear 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 an int. 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.

You are welcome to write additional functions if you feel that they will be of use. 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 character of the text. If that doesn't work, compare the pattern against the text starting with the second character 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 in action:



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

As with HW 3, none of the code in your functions should use array notation. **Use pointer notation exclusively!**

Hint: for this assignment you need to use getchar() and putchar():

- The function *getchar* returns the ASCII code of the next input character. This value will be positive, unless EOF is encountered, in which case -1 is returned.
- The function *putchar* takes a single argument, which is the ASCII code of some character. The function writes that character to output.
- These functions are not built into C. Instead, they are part of the *stdio* library.

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