CS 136 Fall 2008
Mark Giesbrecht

Assignment 1
Due September 25, 2008 at 4pm.

For the Scheme questions you must use the module language. Please follow the CS 135 guide for coding and commenting.

1. The `grep` command in UNIX searches for a sequence of characters or a pattern in a text file. In this assignment you will write a variant of `grep` in Scheme.

   Write a Scheme function `grep` which consumes two string arguments: a *search word*, and a *file name*. The function produces a list of pairs of integers, where each pair contains the line number and column number of where the *search word* appears in the file.

   For example, if the file `marksfile.txt` has contents:

   ```
   All work and no play makes
   Mark work less and working less
   makes him unworkable.
   ```

   Then the expression (*grep* "work" "marksfile.txt") produces ((1 5) (2 6) (2 20) (3 13)).

   Put your function(s) for this question in a file `a1q1-grep.ss`.

2. Sometimes it is useful to match patterns instead of simply words. In UNIX, the `grep` command allows for *wild card* matching, where special characters in a pattern may match many different characters in a line. For example, the period (`.`) matches *any* single character, while the asterisk (`*`) matches zero or more of any character.

   For example:

`"a.c"`	matches the strings `"abc"`, `"aac"`, `"a c"`
`"a.c.e`	matches the string `"abcde"`
`"a.c"`	does not match the strings `"ac"`, `"acb"`
`"a*c"`	matches the strings `"ac"`, `"axc"`, `"axyc"`, `"axxyc"`.
`"a*c"`	does not match the strings `"a"`, `"aC"`

   All matches are case sensitive. That is "`a`" does not match "`A`" and vice versa.

   In this question you are to write a simple string pattern matcher. Write a function *match*, which consumes two strings as parameters, a pattern and a line. It produces #t if the line matches the pattern and #f otherwise.

   For example:

   (*match* "a.c" "aBc"); ⇒ #t
   (*match* "a.c.f" "a c f") ; ⇒ #t
   (*match* "a.c.f" "a c fX") ; ⇒ #f
   (*match* "a*f" "a n   f") ; ⇒ #t
   (*match* "a*f" "Xabf"); ⇒ #f
   (*match* "a.b*c" "aXbYYc"); ⇒ #t

(*match* "∗a.b∗" "xxxaybzzzz"); ⇒ #t
(*match* "∗∗a.b.∗" "xxxaybzzzz"); ⇒ #t

Put your function(s) for this question in a file `a1q2-match.ss`.

3. The UNIX command `grep` also allows *wild cards* in searches. In this question you are to write a pattern-matching variant of `grep`. Write a scheme function *pgrep* which consumes three string arguments, a pattern, an input file, and an output file. When you run your function, all lines in the input file which match the pattern are to be written to the output file. Read lines from the input file using *read-line*, and write lines to the output file using *fprintf*, as described in class.

   For example, if the input `ifile.txt` has contents:

```
I am drinking
I am drinking beer with yellow flowers
in underground sunlight
and you can see that I am a sensitive man
And I notice that the bartender is a
sensitive man too
so I tell him about his beer
```

   then the scheme expression (*pgrep* "∗I∗m∗i∗" "ifile.txt" "ofile.txt") causes the file `ofile.txt` to contain:

```
I am drinking
I am drinking beer with yellow flowers
and you can see that I am a sensitive man
so I tell him about his beer
```

   while the scheme expression (*pgrep* "∗a∗d.i∗" "ifile.txt" "ofile.txt") causes the file `ofile.txt` to contain:

```
I am drinking
I am drinking beer with yellow flowers
```

   You should use your function *match* from question 2.

   Put your function(s) for this question in a file `a1q3-pgrep.ss`.