University of Arizona, Department of Computer Science

CSc 372 — Assignment 2 — Due noon, Fri Sep 23 — 7%

Christian Collberg
September 12, 2005

1 Introduction

The purpose of this assignment is to get improve our skills writing Haskell functions over lists. For the purposes of this assignment, don't use any of the higher-order built-in functions such as `map`, `foldr`, etc. — I want you to write all functions "from scratch"!

Whenever possible, use Haskell's *pattern syntax*, or *guard* syntax.

You may freely introduce auxiliary functions if that makes your program cleaner. Also, feel free to introduce local definitions (`where`-clauses) to make your code easier to read.

You will be graded primarily on correctness and style, not on the execution efficiency of your code.

Functions written for one problem may be freely used in subsequent problems. In fact, this is encouraged!

All functions must be commented.

2 Grading Problems

1. The final grade assignment for a student in this class with a total score of t and final exam score of f is given by the formula [10 points]

$$\text{grade} = \begin{cases} E & \text{if } f < 50 \\ \begin{cases} A & \text{if } t \in [90, 100] \\ B & \text{if } t \in [80, 89] \\ C & \text{if } t \in [70, 79] \\ D & \text{if } t \in [60, 69] \\ E & \text{if } t < 60 \end{cases} & \text{otherwise} \end{cases}$$

   Code this formula in Haskell:

```
grade :: Float -> Float -> Char
grade f t = ...

> grade 45.0 85.0
'E'
> grade 90.0 85.0
'B'
> grade 90.0 105.0
program error: illegal score
```

```
> grade (-5.0) 99.0

Program error: illegal score
```

2. Write a function `member x ys` which returns `True` if `x` is an element of `ys`, `False` otherwise. [10 points]

```
member :: Float -> [Float]-> Bool
member x xs = ...

> member 4 [1,2,3]
False
> member 4 [1,2,3,4.0,5]
True
```

3. Write a function `maxl xs` that generates an error `"empty list"` if `xs==[]` and otherwise returns the largest element of `xs`: [10 points]

```
maxl :: [Float] -> Float
maxl xs = ...

> maxl [2,3,4,5,1,2]
5.0
> maxl []

Program error: empty list
```

4. Write a function `mull xs m` which returns a new list containing the elements of `xs` multiplied by `m`: [10 points]

```
mull ::[Float] -> Float -> [Float]
mull xs x = ...

> mull [1,2,3,4,5] 0.0
[0.0,0.0,0.0,0.0,0.0]
> mull [1,2,3,4,5] 2
[2.0,4.0,6.0,8.0,10.0]
```

5. Write a function `setsub xs ys` that takes two integer lists `xs` and `ys` as input, both lists representing sets. In other words, `xs` and `ys` are unsorted lists that contain no duplicate elements. `setsub xs ys` returns `xs-ys`, i.e. the list containing the elements in `xs` that are not in `ys`: [10 points]

```
setsub :: [Float] -> [Float] -> [Float]
setsub xs ys = ...

> setsub [1,2,3] []
[1.0,2.0,3.0]
> setsub [1,2,3] [3]
[1.0,2.0]
> setsub [1,2,3] [1,2,3]
[]
```

```
> setsub [] [1,2,3]
[]
> setsub [] []
[]
```

6. In computing the grades for this class, each individual score (final, midterm, quizzes, assignments) will be curved by multiplying each score by 100.0/*2nd_highest_class_score*. [15 points]

   This can be computed by the formula

   $$\min(100, (100.0/\max(x - \max(x)))s)$$

   where `-` is set-subtraction, `x` is the set of grades, and `s` is the score of a particular student.

   Use `maxl`, `mull`, and `setsub` from the problems above to write a function `curve xs` that scales the scores in `xs`.

   For example, assume the following final exam scores:

   $$34 \quad 45 \quad 66 \quad 88 \quad 98$$

   After the curve has been applied, the scores will be

   $$38.6 \quad 51.1 \quad 75 \quad 100 \quad 100$$

   Note that no score can be higher than 100.0.

```
curve :: [Float] -> [Float]
curve xs = ...

> curve [34,45,66,88,98]
[38.63636,51.13636,75.0,100.0,100.0]
```

7. Define a recursive function `histogram L` which takes a list of grades as input (pairs of `(final,total)`) and returns a string which is a histogram of the grades. Example: [15 points]

```
> putStr (histogram [(60,90),(50,38),(100,100),(100,100),(34,80),(92,92)])
'A' *** (3)
'B' * (1)
'C'  (0)
'D'  (0)
'E' ** (2)
```

   `putStr` prints out a string on the console. You may need to use the built-in `show n` function, which converts a number/character/etc. to a string:

```
> show 6
"6"
> show '6'
"'6'"
> show 6.6
"6.6"
```

   You will find that for a more complicated function like `histogram` you must break it up into several smaller functions. For example, you could organize your program like this:

```
-- Return the number of students who got a particular
-- grade. For example,
--   > grades  [(60,90),(50,38),(100,100),(100,100),(34,80),(92,92)] 'A'
--     3
grades :: [(Float,Float)] -> Char -> Int
grades ((f,t):xs) g = ...

-- Helper function for histogram. The first argument is
-- the list of grades that we are going to print out
-- the histogram for.
hist :: [Char] -> [(Float,Float)] -> String
hist (g:gs) ft = ...

-- The main function to print out a histogram.
histogram :: [(Float,Float)] -> String
histogram ft = hist ['A','B','C','D','E'] ft

-- Generate a string consisting of n copies of c
stringDup Int -> String -> String
stringDup n c = ..
```

3 Computing Primes

Compute lists of prime numbers, using a Eratosthenes' sieve. [20 points]

The method is as follows:

1. start with a list of numbers beginning with 2, for example [2,3,4,5,6,7,8,9,10].
2. The first number in the list is prime. Remove all its multiples. In this case we get [3,5,7,9].
3. Repeat the previous step: 3 is prime, and after 'sieving' out the multiples we are left with [5,7].
4. Repeat the step again: 5 is prime, and sieving leaves [7].
5. Do it again: 7 is prime, and sieving leaves [].
6. When no numbers remain, we've found all the primes in the given range.

Define a function `sieve xs` which implements this algorithm, such that sieve `[2..10] = [2,3,5,7]`.

Use the function

```
relprime p n = n `mod` p > 0
```

to test if one number is a multiple of another.

You may find it helpful to first implement a function `siever n xs` that returns the elements of `xs` that are not multiples of `n`.

Examples:

```
> siever 2 [3..10]
[3,5,7,9]
> sieve [2..10]
[2,3,5,7]
> sieve [2..20]
[2,3,5,7,11,13,17,19]
> sieve [2..]
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,
71,73,79,83,89,97,101,103,107,109,113,127,131,137,139,
149,151,157,163,167,173,179,181,191,193^C{Interrupted!}
```

4 Submission and Assessment

The deadline for this assignment is noon, Fri Sep 23. It is worth 7% of your final grade.

You should submit the assignment electronically using the `Unix` command

```
turnin cs372.2 ass2.hs
```
.

Don't show your code to anyone, don't read anyone else's code, don't discuss the details of your code with anyone. If you need help with the assignment see the instructor or the TA.