Programming Languages - Homework 3

Due: **June 10 (Fri) 11:59 pm**

– Zip the source code (ONLY .hs files; absolutely no executable or object files) and submit it in ezhub (portal).

– The program must run on the Linux server (csedev.hanyang.ac.kr).

Write Haskell functions for the following problems.

1. Finding Primes (25pts)

Print prime numbers using given inputs (N and M). You have to find M prime numbers which are bigger than or equal to N.

- Input : initial number ‘N’ and the number of primes ‘M’
- Output : a list of primes
- Report error if input is not correct.

```
e.g. ghci> FindingPrimes 2 5
     2 3 5 7 11
     ghci> FindingPrimes 6 6
     7 11 13 17 19 23
```

2. Flip Coins (25pts)

SeulGi has a **stack** of coins. She has a strange hobby, which is to make all coins’ face same direction. Without scattering coins, she is supposed to make all coins head up by flipping the stack repeatedly. Let’s see following example, where H means Head and T means Tail.

```
flip "HTTHTHH" 5
"HTHHTHH"
flip "HTHHTHH" 4
"TTHTTHH"
```

First, flipping 5 coins changes “HTTHT” into “HTHHT” (=not “THTTH”) because the order is reversed and each coins’ face is flipped. In the same manner, flipping 4 coins makes “HTHH” become “TTHT”. Make a function that lists the indices of flipping to make all coins face same way as following described rule.

- Input : A string will consist of more than 2 coins and each coin will have a character either ‘H’ or ‘T’. The top coin on a stack appears first on the string, the bottom coin appears last.
- Output : A list of digits which are indices of flip (the index from the top, 1~n).

If all coins face **Head**, put 0 which means the flipping is end.

- Report error if input is not correct.

```
e.g. ghci> flipCoin "HT"
     [1,2,0]
     ghci> flipCoin "HTTH"
     [1,3,0]
     ghci> flipCoin "THTHTH"
     [5,3,1,2,4,0]
```

3. Minesweeper (25pts)

Have you ever played Minesweeper? It's a cute little game which comes within a certain Operating System which name we can't really remember. Well, the goal of the game is to find where are all the mines within a M × N field. To help you, the game shows a number in a square which tells you how many mines there are adjacent to that square. For instance, supose the following 4 × 4 field with 2 mines (which are represented by an '*' character):

```
*...
....
.*..
....
```

If we would represent the same field placing the hint numbers described above, we would end up with:

```
*100
2210
1*10
1110
```

As you may have already noticed, each square may have at most 8 adjacent squares.

- Input : The first two elements of input are integers n and m which stands for the number of lines and columns of the field respectively. The next string contains n*m characters to represent the field and n-1 backslashes, which mean "newline". Each safe square is represented by an '.' character (without the quotes) and each mine square is represented by an '*' character (also without the quotes).
- Output : The output should contain the field with the '.' characters replaced by the number of adjacent mines to that square.

- Report error if input is not correct.

```
e.g. ghci> minesweep 4 4 "*...\....\.*..\...." % … (1)
     "*100\2210\1*10\1110"
     ghci> minesweep 3 5 "..*..\.*.*.\..*.." % … (2)
     "12*21\1*4*1\12*21"

     (1)
     *...         *100
     ....         2210
     .*..         1*10
     ....         1110

     (2)
     ..*..        12*21
     .*.*.        1*4*1
     ..*..        12*21
```

4. Expressions (25pts)

Let X be the set of correctly built parenthesis expressions. The elements of X are strings consisting only of the characters '(' and ')'. The set X is defined as follows:

- an empty string belongs to X
- if A belongs to X, then (A) belongs to X
- if both A and B belong to X, then the concatenation AB belongs to X.

For example, the following strings are correctly built parenthesis expressions (and therefore belong to the set X): `"()(())()", "(()(()))"`

The expressions below are not correctly built parenthesis expressions (and are thus not in X):
`"(()))(()", "())(()"`

Let E be a correctly built parenthesis expression (therefore E is a string belonging to X).

The length of E is the number of single parenthesis (characters) in E.

the depth D(E) of E is defined as follows:

```
D(E) = 0, if E is empty
     | D(A) + 1, if E = (A), and A is in X
     | max(D(A), D(B)), if E = AB, and A and B are in X
```

For example, the length of '`()(())()`' is 8, and its depth is 2.

What is the number of correctly built parenthesis expressions of length n and depth d, for given positive integers n and d?

- Input : two integers - n and d.
- Output : The number of correctly built parenthesis expressions of length n and depth d.
- Report error if input is not correct.

```
 e.g. ghci> expression 6 2
      3
      ghci> expression 20 10
      1
```

There are exactly three correctly built parenthesis expressions of length 6 and depth 2: `(())()`, `()(())`, `(()())`