Домашна работа № 3 по Функционално програмиране
специалност „Компютърни науки“, II курс, II поток
2022/2023 учебна година

Решенията трябва да са готови за автоматично тестване. Важно е програмният код да бъде добре форматиран и да съдържа коментари на ключовите места. Предайте решенията си в един архив с наименование *hw3_<FN>.zip*, където *<FN>* е Вашият факултетен номер, който включва

- файл с наименование *hw3_<FN>.hs*, съдържащ решенията на двете задачи;
- файловете с тестови случаи *sample.txt* и *input.txt*.

Домашните работи се предават като изпълнение на съответното задание в курса по ФП в Moodle (https://learn.fmi.uni-sofia.bg/course/view.php?id=8504) най-късно до 23:55 ч. на 04.01.2023 г. (сряда).

Приятна работа и успех!

Задача 1.

Нека са дадени два сортирани във възходящ ред списъка от цели числа **`as`** и **`bs`**. Да се дефинира функция **`generate :: [Int] -> [Int] -> [[Int]]`**, която връща като резултат списък от всички възможни списъци от числа във възходящ ред, за които първият елемент е от **`as`**, вторият елемент е от **`bs`** и т.н. Задължително последният елемент на всеки от тези списъци е число от списъка **`bs`**.

Пример:

```
generate [10, 15, 25] [1, 5, 20, 30] → [[10, 20], [10, 20, 25, 30],
[10, 30], [15, 20], [15, 20, 25, 30], [15, 30], [25, 30]]
```

Задача 2.

Докато подготвяли шейната за тазгодишния полет, елфите на Дядо Коледа забелязали, че четирицифрените седемсегментни дисплеи не работят добре; вероятно са се повредили по време на едногодишния престой. Те Ви молят за помощ - без тези дисплеи Дядо Коледа няма да може да разбере дали се движи навреме и дали не са се появили други проблеми с шейната по време на полета.

Всяка цифра на седемсегментен дисплей се изобразява чрез включване или изключване на кой да е от седемте сегмента от `a` до `g`:

```
  0:          1:          2:          3:          4:
 aaaa        ....        aaaa        aaaa        ....
b    c      .    c      .    c      .    c      b    c
b    c      .    c      .    c      .    c      b    c
 ....        ....        dddd        dddd        dddd
e    f      .    f      e    .      .    f      .    f
e    f      .    f      e    .      .    f      .    f
 gggg        ....        gggg        gggg        ....

  5:          6:          7:          8:          9:
 aaaa        aaaa        aaaa        aaaa        aaaa
b    .      b    .      .    c      b    c      b    c
b    .      b    .      .    c      b    c      b    c
 dddd        dddd        ....        dddd        dddd
.    f      e    f      .    f      e    f      .    f
.    f      e    f      .    f      e    f      .    f
 gggg        gggg        ....        gggg        gggg
```

Следователно, за да се изобрази `1`, ще бъдат включени само сегменти `c` и `f`; останалото ще бъде изключено. За изобразяване на `7` ще бъдат включени само сегменти `a`, `c` и `f`.

Проблемът е, че сигналите, които управляват сегментите, са се разместили за всеки дисплей. Главният компютър на шейната все още се опитва да показва числа, като произвежда изходни сигнали на сигнални проводници от `a` до `g`, но тези проводници са свързани към сегменти **произволно**. Още по-лошо: проводниците/сегментните връзки се смесват отделно за всеки четирицифрен дисплей! (Всички цифри **в конкретен дисплей** обаче използват едни и същи връзки.)

С други думи, от това, че само сигналните проводници `b` и `g` са включени, не следва, че **сегментите** `b` и `g` са включени. Единствената цифра, която използва два сегмента, е `1`, следователно това, че само сигналните проводници `b` и `g` са включени, означава, че сегменти `c` и `f` са включени. Обаче само с тази информация все още не е ясно кой проводник (`b`/`g`) към кой сегмент (`c`/`f`) отива. За целта ще трябва да съберете повече информация.

За всеки дисплей наблюдавате променящите се сигнали за известно време, отбелязвате **всичките десет уникални модела на сигнала**, които виждате, и след това записвате една единствена **четирицифрена изходна стойност**. Използвайки моделите на сигнала, трябва да можете да определите кой модел на коя цифра съответства.

Например, ето какво може да видите в един запис в бележките си:

```
acedgfb cdfbe gcdfa fbcad dab cefabd cdfgeb eafb cagedb ab | cdfeb
fcadb cdfeb cdbaf
```

(Записът тук е разположен на два реда; във вашите бележки той ще бъде на един ред.)

Всеки запис се състои от десет **уникални модела на сигнал**, разделител (`|`) и накрая **четирицифрената изходна стойност**. В рамките на един запис се използват едни и същи проводници/сегментни връзки (но не знаете как те са свързани). Моделите на сигнала съответстват на десетте различни начина, по които компютърът на шейната се опитва да изобрази цифра, използвайки текущите кабелни/сегментни връзки. Тъй като `7` е единствената цифра, която използва три сегмента, **`dab`** в горния пример означава, че за изобразяване на `7` сигналните линии `d`, `a` и `b` са включени. Тъй като `4` е единствената цифра, която използва четири сегмента, **`eafb`** означава, че за изобразяване на `4` сигналните линии `e`, `a`, `f` и `b` са включени.

Използвайки тази информация, трябва да можете да определите коя комбинация от сигнални проводници съответства на всяка от десетте цифри. След това можете да декодирате четирицифрената изходна стойност. За съжаление в горния пример всички цифри в изходната стойност (`cdfeb`, `fcadb`, `cdfeb`, `cdbaf`) използват пет сегмента и са по-трудни за извеждане.

Засега решавате да се **фокусирате върху лесните цифри**. Разглеждате тази част записи:

```
be cfbegad cbdgef fgaecd cgeb fdcge agebfd fecdb fabcd edb |
fdgacbe cefdb cefbgd gcbe
edbfga begcd cbg gc gcadebf fbgde acbgfd abcde gfcbed gfec |
fcgedb cgb dgebacf gc
fgaebd cg bdaec gdafb agbcfd gdcbef bgcad gfac gcb cdgabef |
cg cg fdcagb cbg
fbegcd cbd adcefb dageb afcb bc aefdc ecdab fgdeca fcdbega |
efabcd cedba gadfec cb
aecbfdg fbg gf bafeg dbefa fcge gcbea fcaegb dgceab fcbdga |
gecf egdcabf bgf bfgea
fgeab ca afcebg bdacfeg cfaedg gcfdb baec bfadeg bafgc acf |
gebdcfa ecba ca fadegcb
dbcfg fgd bdegcaf fgec aegbdf ecdfab fbedc dacgb gdcebf gf |
cefg dcbef fcge gbcadfe
bdfegc cbegaf gecbf dfcage bdacg ed bedf ced adcbefg gebcd |
ed bcgafe cdgba cbgef
egadfb cdbfeg cegd fecab cgb gbdefca cg fgcdab egfdb bfceg |
gbdfcae bgc cg cgb
gcafb gcf dcaebfg ecagb gf abcdeg gaef cafbge fdbac fegbdc |
fgae cfgab fg bagce
```

Тъй като всяка от цифрите `1`, `4`, `7` и `8` използва уникален брой сегменти, може да намерите кои комбинации от сигнали съответстват на тези цифри. Преброявайки **само цифрите в изходните стойности** (частта след `|` на всеки ред), в горния пример има `26` екземпляра на цифри, които използват уникален брой сегменти (маркирани по-горе).

Да се дефинира функция **`countUniques :: String -> Int`**, която приема последователност от записи и извежда броя на появяванията на цифрите `1`, `4`, `7` или `8` в изходните стойности.

Примери:

```
sample <- readFile "sample.txt"
input <- readFile "input.txt"
countUniques sample → 26
countUniques input  → 245
```