Kurs języka Haskell 2024/25

LISTA NR 11 (TERMIN: 3.02.2025, godz 5:00)

Uwaga: Wszystkie rozwiązania należy umieścić w jednym module o nazwie `Main` zachowując sygnatury zgodne z szablonem rozwiązań zamieszczonym w SKOS-ie.

Zadanie 1. Rozważ multizbiór zawierający ciągi. Można go reprezentować jako drzewo prefiksowe (*trie*) z etykietami mówiącymi, ile wystąpień danego ciągu jest w multizbiorze:

```
data Trie c = PNode Int [(c, Trie c)]
```

Zaimplementuj operacje wstawiania do drzewa i sprawdzania liczby wystąpień danego słowa:

```
trieEmpty  :: Trie c
trieInsert :: (Eq c) => [c] -> Trie c -> Trie c
trieFreq   :: (Eq c) => [c] -> Trie c -> Int
```

Na przykład:

```
ghci> foldr trieInsert trieEmpty ["ab", "abc", "xx", "ab"]
 PNode 0 [
   ('x', PNode 0 [('x', PNode 1 [])]),
   ('a', PNode 0 [('b', PNode 2 [('c', PNode 1 [])])])]
ghci> trieFreq "ab" it
2
```

Zadanie 2. Napisz program który:

1. Wczytuje z argumentu wywołania nazwę pliku,
2. Wczytuje plik, dzieli go na wyrazy i normalizuje je (np. przekształca wszystkie do wielkich znaków),
3. Tworzy z tych wyrazów drzewo prefiksowe
4. Pyta w pętli użytkownika o słowa, a następnie wypisuje liczbę wystąpień słowa w zadanym pliku.

Mogą przydać się funkcje **System**.Environment.**getArgs**, Data.**Char**.**toUpper**, **words**.

Zadanie 3. Spróbuj poprawić wydajność programu przez lepszą implementację typu `Trie`. Sprawdź (benchmarkując przy użyciu jakiejś dedykowanej do tego biblioteki, narzędzia, lub chociażby poleceniem `time`), czy można poprawić wydajność np. dodając odpowiednie anotacje `!`, pragmy `UNPACK`, oraz używając sprytniejszej struktury danych niż lista asocjacyjna. Przygotuj odpowiednio duże dane testowe (można wkleić w komentarzu zawartość skryptu, który utworzy odpowiednie dane testowe przy użyciu mieszanki poleceń `curl` i `cat`[1]). Zapisz wyniki i wnioski w komentarzu w kodzie.

Zadanie 4. A co gdybyśmy zamiast drzewa prefiksowego użyli Data.Map.Map **String Int**? Porównaj wydajność. Zapisz wyniki i wnioski w komentarzu w kodzie.

Zadanie 5. Zdefiniuj funkcję

```
tryFreqSub :: [c] -> Trie c -> Int
```

która zdradza sumaryczną liczbę wystąpień danego podciągu. Na przykład:

```
ghci> foldr trieInsert trieEmpty ["abxab", "xab", "xab"]
...
ghci> trieFreqSub "ab" it
4
```

[1] Dobrym źródłem długich tekstów w formacie tekstowym dostępnych przez stabilne URL-e jest https://www.gutenberg.org, np. `curl https://www.gutenberg.org/cache/epub/4300/pg4300.txt > ulysses.txt`