Lista zadań nr 9

Zadanie 1. (2 pkt)

Pętla **`while`** w czystym języku funkcyjnym nie ma sensu, bo wartość wyrażenia kontrolnego nigdy nie zmieni się w czasie. Ale odkąd mamy mutowalny stan, nasz język przestał być czysty. Dodaj do języka FUN_EXPLICIT lub FUN_MONAD pętlę **`while`**.

Ciekawostka: W OCamlu też jest pętla while. W tym i kolejnym zadaniu nie wolno z niej korzystać!

Zadanie 2. (2 pkt)

Rozbuduj pętlę z poprzedniego zadania o konstrukcje `break` i `continue`.

Zadanie 3. (1 pkt)

Zmodyfikuj język z wyjątkami FUN_EXC tak, by próba dzielenia przez zero kończyła się zgłoszeniem wyjątku.

Zadanie 4. (2 pkt)

W językach FUN_EXC i FUN_MONAD umieść typ obliczeń (`comp`) wewnątrz modułu z sygnaturą, tak, by stał się on abstrakcyjny. Jakie jeszcze operacje dodasz do sygnatury i modułu, by dalej można było go użyć w interpreterze?

Wskazówka: Rozwiązanie tego zadania może bardzo uprościć rozwiązania dwóch kolejnych zadań.

Zadanie 5. (2 pkt)

Połącz języki FUN_MONAD i FUN_EXC w jeden język, który ma i mutowalny stan, i wyjątki.

To zadanie można zrobić na dwa sposoby, w zależności od typu rezultatu. Może to być

```
h -> (v option, h)
```

albo

```
h -> (v, h) option
```

Te dwa typy opisują dwie możliwe wersje złożenia mutowalnego stanu i wyjątków – stan może być albo globalny (wówczas rzucenie wyjątku nie zmienia stanu) albo transakcyjny (wówczas obsługa wyjątku przywraca oryginalny stan sprzed próby obliczenia wyrażenia w bloku `try`) – czy widzisz, który typ odpowiada któremu złożeniu?

Zaimpementuj jeden z tych sposobów. Przygotuj przykład pokazujący, które złożenie zostało zaimplementowane.

Zadanie 6. (1 pkt)

Zaimplementuj też ten drugi rodzaj złożenia stanu i wyjątków.

Zadanie 7. (2 pkt)

Napisz translator wyrażeń w RPN do wyrażeń arytmetycznych w składni abstrakcyjnej.

Wskazówka: Nie po raz pierwszy taki translator ma kształt interpretera tylko z ciekawszym typem wartości. Tym razem jest to oczywiście kształt interpretera wyrażeń RPN, a typem wartości są... wyrażenia w składni abstrakcyjnej.