Zadania: zestaw 4.

W niniejszym zadaniu będziemy kontynuowali naszą przygodę z OCamlem. Zajmiemy się mianowicie rachunkiem λ i kodami de Bruijn'a. Zarówno o jednym jak i o drugim możecie przeczytać w notatkach z wykładu (z tego lub z zeszłego roku), albo po prostu zajrzeć do Wikipedii:

http://en.wikipedia.org/wiki/De_Bruijn_notation

Głównym celem Waszego zadania będzie konwertowanie λ termów do notacji de Bruijn'a i z powrotem oraz wykonywanie pewnych prostych operacji na λ termach zapisanych w odpowiedniej postaci.

1. λ termy

λ termy są reprezentowane przez następujący typ danych:

```
type lambdaterm =
TmVar of string
| TmAbs of string * lambdaterm
| TmApp of lambdaterm * lambdaterm
```

2. Notacja de Bruijn'a

Termy w notacji de Bruijn'a są zdefiniowane przez następujący typ danych:

```
type debruijnterm =
DbVar of int
| DbAbs of debruijnterm
| DbApp of debruijnterm * debruijnterm
```

Kontekst nazewnictwa jest to po prostu lista łańcuchów taka, że ostatni element listy ma zawsze indeks 0, przedostatni indeks 1 itd. Na przykład lista *["v"; "w"; "x"]* ma kontekst nazewnictwa taki, że *"v"* ma indeks 2, *"w"* ma indeks 1, a *"x"* ma indeks 0.

3. Zadanie domowe

(1) Napisać funkcję w OCamlu *fv: lambdaterm -> string* która wypisuje zbiór wszystkich zmiennych wolnych danego termu.
(2) Napisać funkcję w OCamlu *tm2db: lambdaterm -> string list -> debruijnterm* która przy danym λ termie i kontekście nazewnictwa zwraca odpowiadający im term w notacji de Bruijn'a.
(3) Napisać funkcję w OCamlu *isClosedDB: debruijnterm -> bool* która zwraca wartość *true* wtedy i tylko wtedy, gdy dany λ term w notacji de Bruijn'a jest domknięty.
(4) Napisać funkcję w OCamlu *alphaEquivalent: lambdaterm -> lambdaterm -> bool* która zwraca wartość *true* wtedy i tylko wtedy, gdy dane dwa λ termy są α-równoważne.
(5) Napisać funkcję w OCamlu *subst: int -> debruijnterm -> debruijnterm* taką, że *subst i s t* zwraca term otrzymany przez zastąpienie wszystkich wolnych wystąpień i w t przez s.

Rozwiązane zadanie zapiszcie w pliku *lab4.ml* i prześlijcie do mnie do **1 czerwca** (lub do **15 czerwca**, w zależności od grupy ćwiczeniowej). Proszę pamiętać o umieszczeniu tagu [agh2rlog] w polu subject!

4. Uwagi i pomoce

Zapisywanie λ termów w postaci struktur danych OCamla potrafi być uciążliwe. Sugeruję nieśmiało, aby przy testowaniu funkcji posłużyć się bardziej czytelną formą zapisu λ termów. W tym celu możecie posłużyć się funkcją *str2tm* zdefiniowaną w dołączonym parserze *(lambdaparser.ml)*. W szczególności

funkcja ta rozpoznaje znak \ jako λ. Przypomnijmy, że w OCamlu (czy raczej w ML-u) znak \ rozpoznawany jest jako symbol ucieczki, a zatem dla zapisania pojedyńczego znaku \ w łańcuchu powinniście napisać \\.

Typ danych *lambdaterm* jest zdefiniowany w *lambdaparser.ml*. Z kolei parser *lambdaparser.ml* używa skanera *lambdalexer.ml*, zaś obydwa są generowane ze specyfikacji zawartych w *lambdaparser.mly* i *lambdalexer.mll*. Możecie wygenerować stosowne pliki *.ml* przy użyciu *ocamlyacc* i *ocamllex*. Wszystkie pliki do ściągnięcia są tutaj:

http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/lambdaparser.ml
http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/lambdaparser.mly
http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/lambdalexer.ml
http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/lambdalexer.mll.