Programowanie
Druga część egzaminu
Wersja biała

Za cały egzamin będzie można dostać 100 punktów (nie licząc punktów bonusowych). Na tę część przypada 60 punktów (+3 bonusowe). Dla całego egzaminu progi są następujące: 40 punktów daje ocenę dostateczną, 52 dostateczną z plusem, 64 dobrą, 76 dobrą z plusem, 88 bardzo dobrą.

Zatem orientacyjnie, aby zaliczyć tę część trzeba zdobyć 16 punktów.

Zadanie 1. (**8p**) Mamy następujące definicje:

```
fun nlen [] = 0 |
    nlen (_::xs) = 1+ nlen xs;
fun (f o g) x = f(g x);
fun [] @ xs = xs |
    (x::xs)@ys = x::(xs@ys);
```

Udowodnij twierdzenia
a) `nlen(xs@ys) = nlen(xs)+nlen(ys)` (**3p**)
b) `nlen o rev = nlen` (**5p**)

Zadanie 2. (**5p**) To zadanie dotyczy sml-a Napisz w sml-u funkcję, która ma typ:
a) Napisz funkcję, która ma typ: $\beta \to \alpha \to \alpha$
b) Napisz funkcję, która ma typ: $\alpha\text{list} * \beta \to \beta\text{list} * \alpha$
c) Podaj typ dla funkcji: `fun f x y = (y::[[x]],x::y)`.
d) Podaj typ dla funkcji: `fun f x y = x (y=())`.
e) Podaj typ dla funkcji: `fun f x y = x = y = x`.
Po dwa punkty za każdy podpunkt. W punktach c), d) i e) możliwa jest odpowiedź: nie ma typu.

Zadanie 3. (**5p**) W poniższym zadaniu nie musisz uzasadniać odpowiedzi. Na każde pytanie powinieneś odpowiedzieć, wybierając odpowiedź ze zbioru: `tak, nie, ?`. Sprawdzane są tylko odbowiedzi `tak` oraz `nie`. Trafiona daje 1 punkt, chybiona to `-1` punkt. Ujemna liczba punktów za całe zadanie zaokrąglana jest do zera.

Mamy następującą definicję:

```
class P : public B { ... }
```

Czy prawdziwe są następujące zdania:
a) Destruktor wywoływany jest jedynie podczas wykonywania operacji `delete`.
b) Podczas destrukcji obiektu klasy `P` wykonywane są również polecenia destruktora klasy `B`.
c) Za pomocą konstrukcji `template` możemy uzyskać szybszy kod wynikowy.
d) Bezpośrednia zmiana pola klasy jest zalecaną techniką w programowaniu obiektowym.
e) Przypisanie do zmiennej typu B obiektu typu `P` jest niemożliwe.

Zadanie 4. (**12p**)

Będziemy rozważać gramatyki bezkontekstowe w której prawe strony produkcji są albo parą symboli nieterminalnych (zapisywaną jako term `A -> (B,C)`), albo pojedynczym sybolem terminalnym, zapisywanym za pomocą termu `A->term(T)`.

Jakie odpowiedzi (wymień wszystkie) da Prolog na pytanie (**1p**):

```
?- member(A->(B,C), [ zdanie->(gp,go), go->(prz,czas), czas->term(robi) ]).
```

gdzie predykat `member` zdefiniowany jest

```
member(X,[X|_]).
member(X,[_|Xs]):- member(X,Xs).
```

Napisz predykat `len(L,N)`, prawdziwy, gdy `N` jest długością listy `L`, tak by go można było użyć do obliczania długości listy (**2p**).

Uzupełnij poniższy program tak, by powstał predykat `gram(S,P,W,Dlug)`, prawdziwy, gdy `W` jest słowem o długości mniejszej bądź równej `Dlug` wygenerowanej z `S` za pomocą produkcji zapisanych w `P`. Za każdy poprawny wpis (**1p**), dodatkowe (**2p**) za poprawny program.

```
gram(S,P,[X],Dlug) :- Dlug>0, member( (1) ).
gram(S,P,W,Dlug) :-
   D1 is Dlug-1,
   member( S->(A,B), P),
   gram( (2), W1, D1 ),
   length( W1, DW1),
   D2 is (3),
   generuj( (4), W2, D2),
   append(W1,W2,W).
```

Załóżmy, że zapytanie `?-polski(S,P)` unifikuje `S` z symbolem startowym, a `P` ze zbiorem produkcji gramatyki języka poskiego, w postaci występującej w tym zadaniu. Polskie słowa, reprezentowane przez prologowe atomy, są symbolami terminalnymi, zdania to listy takich symboli. Przykładowe zdanie: `[ala, ma, kota]`. Podaj zapytanie, które wygeneruje wszystkie polskie zdania nie dłuższe niż 10 słów, w których ponadto:

a) występuje słowo `programowanie`,
b) pewne słowo się powtarza,
c) występuje ciąg: `X,i,Y`, gdzie `X` oraz `Y` są nazwami zwierząt (mamy predykat `zwierze/1` prawdziwy dla atomów, będących nazwami zwierząt w języku polskim).

Niektóre zdania mogą być generowane więcej niż raz. Każde poprawne zapytanie to (**1p**).

Zadanie 5. Napisz funkcję `iteruj`, o nagłówku `fun iteruj n m f x`, która daje w wyniku listę postaci (**3p**):

```
[ f(n,x), f(n+1,x), ..., f(m-1,x),f(m,x) ]
```

Napisz funkcję `ins (n,(x,l))`, która wstawia na n-tą pozycję w liście `l` element `x` (**3p**). Przykładowo:

```
ins(0, ([1,2,3],7))    =    [7,1,2,3]
ins(1, ([1,2,3],7))    =    [1,7,2,3]
ins(3, ([1,2,3],7))    =    [1,2,3,7]
```

Podaj typy obu tych funkcji(**2p**).

Przy założeniu, że funkcje `iteruj` i `ins` zostały napisane zgodnie ze specyfikacją napisz funkcję `allperm`, która dla listy `l` daje listę wszystkich permutacji listy `l`. Funkcja ta powinna[1] wykorzystywać obie wspomniane wyżej funkcje (**4p**).

Możesz korzystać z funkcji `nlen` liczącej długość listy. Możesz również użyć następującej funkcji:

```
fun map_ap _ []      = [] |
    map_ap f (x::xs) = (f x) @ (map_ap f xs);
```

[1]Można dostać również maksymalną liczbę punktów za to zadanie nie korzystając z wspomnianych funkcji, ale tylko wówczas, gdy zaproponowane rozwiązanie będzie niewiększe od wykorzystującego `ins` oraz `iteruj` rozwiązania wzorcowego