Programowanie – wrzesień 2004
Wersja a

Za cały egzamin będzie można dostać 100 punktów. Progi są następujące: 33 punkty daje ocenę dostateczną, 45 dostateczną z plusem, 57 dobrą, 69 dobrą z plusem, 81 bardzo dobrą.

Zadanie 1. (**14p**) Zadanie to dotyczy języków nad alfabetem $\{a, b\}$.

a) Zdefiniuj gramatykę, która generuje język tych słów, w których liczba liter b nie przekracza liczby liter a. (**5p**) Uzasadnij poprawność swej konstrukcji.(**4p**)

b) Podaj wyrażenie regularne generujące język słów nie zawierających wzorca aa. (**3p**)

c) Podaj wyrażenie regularne generujące język słów, w których liczba wystąpień literki a jest parzysta. (**2p**)

Zadanie 2. (**15p**) Zamień poniższe programy na ich odpowiedniki, w których nie ma instrukcji `goto`, a jedynymi strukturami sterującymi są pętla `while` i instrukcja `if`. Możesz wprowadzać nowe zmienne (również tablicowe), przypisywać im wartości i sprawdzać je.

Występujące w programach warunki nie wywołują efektów ubocznych.

a) (**5p**)

```
if (b1) goto L1;
C2;
while (b2) {
    C3;
L1: if (b3) break;
    C4;
L2: if (b4) goto C4;
}
```

b) (**5p**)

```
int p(int x) {
   int r;
   if (x<=0) return 1;
   while (x) {
      x /= 2;
      s += p(x);
   }
}
```

c) (**5p**) (**UWAGA:** w tym punkcie jedyną strukturą sterującą, z jakiej możesz korzystać jest pętla `while`. W szczególności zabronione jest `if`)

```
if (b1) C1;
else C2;
```

Zadanie 3. (**15p**) Poniższe fragmenty programów w języku C nie robią tego, co mają robić. Dla każdego z nich powiedz, gdzie jest problem i napisz program poprawnie (zwróć również uwagę na styl):

a) Program znajduje drugi największy element w *posortowanej* tablicy `T` zawierającej `N` liczb nieujemnych całkowitych. W tablicy są przynajmniej dwie różne wartości. Dla tablicy {`1,1,2,2,3,3`} program powinien zwrócić `2`.

```
int Prev= -1;
int Max;
for (i=0;i<N,i++) {
    if (T[i] >= Prev) {
       Nax = T[i];
       Prev = Max
    }
}
DrugiNajwiekszy = Prev;
```

b) Program liczący wartość $n!$ dla nieujemnego n

```
int factorial(int n) {
    int fac = 1;
    while (n--)
       fac *= n;
    return fac;
```

c) Program przeglądający napis `s` i zmieniający małe `litery` na duże `LITERY`.

```
#define is_lower(c) ( (c) >= 'a' && (c) <= 'z' )

while ( *s ) {
   if (is_lower(*s++))
      *(s-1) = *(s-1) + 'a' - 'A';
}
```

Zadanie 4. (**14p**) Zdefiniuj w Haskellu lub sml-u funkcję łączącą listy (**2p**). Wykorzystaj ją do zdefiniowania funkcji `nrev`(**2p**), która zmienia na przeciwny porządek elementów na liście (`nrev [1,2,3] = [3,2,1]`). Udowodnij następującą własność (**10p**):

```
nrev (nrev xs) = xs
```

Zadanie 5. 12 Rozważamy wyrażenia arytmetyczne zbudowane z nawiasów, liczb oraz znaków `'-'` (minus) Będziemy przekształcać takie wyrażenie na listę liczb, taką że:

- W liście tej występują te same liczby (z dokładnością do znaku) i w tej samej kolejności co w wyrażeniu.
- Suma elementów listy jest równa wartości wyrażenia.

Napisz w Haskellu, sml-u lub Prologu program, który wykona te przekształcenia. W tym celu:

a) Zdefiniuj typ do reprezentacji wyrażenia (tylko SML i Haskell, w Prologu wyrażenie powinno być reprezentowane jako normalny term z minusami).
b) Napisz funkcję `minus` (sml, Haskell), która zmienia na przeciwne wszystkie swoje elementy lub predykat (Prolog) służący do tego samego celu. (**3p**)
c) Napisz funkcję (predykat) `toList`, która dla danego wyrażenia zwróci listę zgodnie z podanymi wyżej zasadami. (**9p**)

Zadanie 6. (**10p**) Napisz w Haskellu lub sml-u funkcję, która ma typ:

a) `a -> b -> a`
b) `([a],b) -> (b->a) -> (a,[b])`

Podaj typ dla funkcji

A) Podaj typ dla funkcji: `f x y = (y:[[x]],x:y)`.
B) Podaj typ dla funkcji: `f x y = x (y=())`.
C) Podaj typ dla funkcji: `f x y = x = y = x`.

Po dwa punkty za każdy podpunkt. W punktach A), B) i C) możliwa jest odpowiedź: nie ma typu.

Zadanie 7. Odpowiedz na poniższe pytania. Proszę o odpowiedź na każde z pytań, przy czym odpowiedź należy wybrać ze zbioru: `T,N,?`. Odpowiedź `?` warta jest zawsze 0 punktów, odpowiedi `T` oraz `N` są warte -2 lub 2, w zależności od tego, czy są poprawne.

a) Rozstrzygalny jest następujący problem: Czy program P w języku D^* bez operacji wejścia i wyjścia zawiera instrukcję, która wykona się więcej niż 1000 razy.
b) Rozstrzygalny jest następujący problem: Czy program P w języku D^* bez operacji wejścia i wyjścia zawiera instrukcję, która wykona się mniej niż 1000 razy.
c) Czy jeżeli usuniemy z języka D^* operacje wejścia i wyjścia, natomiast typ `int` ograniczymy do 16 bitów, to problem stopu dla programów w tym ograniczonym języku stanie się rozstrzygalny?
d) Gdyby w języku D^* (bez instrukcji wejścia i wyjścia) zamienić pętle `while` na pętlę `for (i=0;i<N;i++) ...` oraz zabronić modyfikacji zmiennej sterującej w ciele pętli, wówczas każde pytanie typu : „czy program wykona kiedykolwiek pewną akcję" stałoby się rozstrzygalne.
e) Jeżeli prologowy program dla pewnego zapytania nie generuje w skończonym czasie odpowiedzi, to sytuacji tej nie da się naprawić zmieniając kolejność atomów w ciele klauzuli.
f) Ponieważ w Pythonie błędy typu zprawdzane są podczas wykonania, fragment programu `x=5; x='alala'` jest niepoprawny.
g) `[1,2,[3,4]]` jest poprawna lista w Pythonie, ale nie w Haskellu.
h) Odcięcie na końcu ostatniej klauzuli w predykacie nigdy nie zmienia semantyki programu w Prologu
i) Podczas destrukcji w C++ najpierw wywoływany jest destruktor klasy bazowej, następnie pochodnej.
j) W Pythonie i Javie, tak jak w C++ instrukcja `a=b` oznacza skopiowanie wszystkich pól obektu `b` do obiektu `a`.