Programowanie IX.2003

Za egzamin można dostać 70 punktów. Progi są następujące: 23 punkty daje ocenę dostateczną, 32 dostateczną z plusem, 41 dobrą, 50 dobrą z plusem, 59 bardzo dobrą.

Zadanie 1. (**16p**) Niech G_1 będzie gramatyką generującą język L_1 nad alfabetem $\Sigma = \{a, b\}$ z symbolem startowym S i zbiorem produkcji P równym

$$\{S \to aSb,\ S \to bSa,\ S \to \varepsilon\}$$

Odpowiedz na pytania:

a) Czy język L_1 zawiera wszystkie słowa, w których liczba znaków a jest równa liczbie znaków b (odpowiedź uzasadnij) (**3p**).
b) Czy L_1 jest zbiorem regularnym? (**2p**)
c) Czy gramatyka G_1 jest jednoznaczna? (odpowiedź udowodnij (**6p**))
d) Zdefiniuj gramatykę G_2 generującą język $L_1 \cap \mathcal{L}(a^*b^*a^*b^*)$ (**5p**)

Zadanie 2. (**20p**) W zadaniu tym powinieneś przekształcić fragmenty programów w języku C w ten sposób, by dawały ten sam efekt. Zakładamy, że sprawdzenie warunku nie powoduje efektów ubocznych. Jeżeli potrzeba, to możesz wprowadzać dodatkowe zmienne. Każdy podpunkt to (**5p**)

a) Przekształć program:

```
switch(c) {
   case 1: C1;
   case 2: C2; break;
   case 3: C3; break;
}
```

na równoważny, w którym jedyną konstrukcją sterującą jest `if` bez `else`.

b) Przekształć program:

```
for (i=N; i--; k++) {
   C1;
   if (b) break;
   C2
}
```

na równoważny, w którym jedynymi konstrukcjami sterującymi są instrukcja `goto` oraz instrukcja `if` z opcjonalnym `else`.

c) Przekształć program:

```
do {
   C1;
   if (b1) continue;
   C2;
} while(b2);
```

na równoważny, w którym jedynymi konstrukcjami sterującymi są instrukcja `while` oraz instrukcja `if` z opcjonalnym `else`.

d) Przekształć program:

```
    C1;
E1: C2;
    if (b1) goto E2;
    C3;
    if (b2) goto E1;
E2: C4;
```

na równoważny, w którym jedynymi konstrukcjami sterującymi są instrukcja `do-while`, `break` oraz instrukcja `if` z opcjonalnym `else`.

Zadanie 3. (**21p**) Mamy następujące definicje:

```
fun flatten [] = [] |
    flatten (x::xs) = x @ (flatten xs);
 fun [] @ xs = xs |
    (x::xs)@ys = x::(xs@ys);
```

a) Napisz funkcję `flat`, która liczy to samo co `flatten`, ale nie używa `@`. Podaj jej typ. (**6p**)
b) Udowodnij twierdzenie `a@(b@c) = (a@b)@c` (**5p**)
c) Udowodnij twierdzenie: (**8p**) `flatten(xs@ys)=(flatten xs)@(flatten ys)`

Dlaczego w SML-u (w przeciwieństwie do Prologa) nie można napisac funkcji `flatten` tak, by `flatten [ [], [1,2,[3,4]],[[5]]]` dało listę `[1,2,3,4,5]`. (**2p**).

Zadanie 4. (**14p**) Sortowanie bąbelkowe polega na iteracyjnym wybieraniu dwóch sąsiednich elementów listy i zamianie ich miejscami, jeżeli pierwszy jest większy od drugiego. Proces kończy się, gdy nie ma takiej pary elementów.
a) Napisz predykat `swap(L1,L2)` prawdziwy, gdy lista liczb `L2` może powstać z listy `L1` przez zamianę dwóch elementów, w sposób przedstawiony wyżej. (**5p**)
b) Jaką wartość przyjmie `Lista` po wykonaniu następującego zapytania:(**2p**)

```
?- findall(x,swap([1,2,3,4,3,2,1],L),Lista).
```

c) Wykorzystaj predykat `swap` do napisania predukatu `bubblesort(L,L2)`, który sortuje listę liczb `L` i unifikuje listę `L2` z wynikiem. Predykat powinien realizowac algorytm sortowania bąbelkowego. (**7p**)

W przypadku b) sprawdzający zakłada, że `swap` jest napisane zgodnie ze specyfikacją.