Egzamin z programowania
część I

Zadanie 1. (12p) Zamień poniższe programy na ich odpowiedniki, w których nie ma instrucji `goto`, a jedynymi strukturami sterującymi są pętla `do while` i instrukcja `if`.

a)
```
L1: C1;
    if (b1)  goto L1;
    C2;
    if (b2)  goto L2;
    C3;
L2: C4;
```

b)
```
    do {
      C1;
      if (b1)  goto L1;
      C2;
    } while (b);
L1: C3;
```

c)
```
L1: C1;
    if (b)
    { C2;
      if (b1) then goto  L1;
      C3;
    }
```

Zadanie 2. (12p) Listy możemy sortować za pomocą metody zwanej **sortowaniem przez scalanie**. W metodzie tej dzielimy listę na dwie w przybliżeniu równe części, sortujemy je rekurencyjnie i następnie scalamy tworząc wynikową, posortowaną listę. Napisz funkcje:

a) `dow* merge(dow *L1, dow *L2)`, która zwraca w wyniku wskaźnik do listy zawierającej elementy z list `L1` oraz `L2` w porządku rosnącym. Zakładamy, że `L1` oraz `L2` są posortowane.

b) `void divide(dow *L, dow* &L1, dow* &L2)`, która dzieli `L` na dwie w przybliżeniu równe części i zwraca w `L1` wskaźnik do pierwszej z nich, a w `L2` do drugiej.

c) `dow* mergesort(dow *L)`, która sortuje listę `L` przez scalanie i zwraca do niej wskaźnik.

Dowiązanie jest zdefiniowane jako `struct dow { int key; dow* next }`. Fukcja `merge` powinna zdefiniowana rekurencyjnie.

Zadanie 3. Pierwsze przybliżenie procedury sortującej przez wstawianie mogłoby wyglądać następująco:

```
i=0;
while(i<N) (
   wstaw a[i] w odpowiednie miejsce w tablicy a[] wśród pozycji 0,1,...,i
      być może przesuwając pewne elementy
   i=i+1;
)
```

a) **(2p)** Jak wyglądałby niezmiennik tej pętli użyty w dowodzie stwierdzenia mówiącego, że po zakończeniu działania tablica jest posortowana?

b) **(3p)** Napisz ostateczną wersję tego fragmentu programu.

c) **(4p)** Napisz funkcję `bool ZawieraPodzbiór(int a[], int N, int K, int P)` sprawdzającą, czy w tablicy `a` o wielkości `N` istnieje `K` elementowy podzbiór elementów, których suma jest mniesza niż `P`.

Zadanie 4. (10p) Definujemy składnię dwóch wersji programów licznikowych:

- Wersja (**A**), półstrukturalna:

```
instrukcja =  [ etykieta ] instrukcja
            |  ident "="  ident ( ("+" "1") | ("-" "1") )
            |  ident "=" "0"
            | "if" ident ("=" | "!=") "0" "then" instrukcja { instrukcja } "fi"
            | "goto" etykieta
```

- Wersja (**B**), standardowa, nieznacznie wzbogacona:

```
instrukcja =  [ etykieta ] instrukcja
            |  ident "=" ident ( ("+" "1") | ("-" "1") )
            |  ident "=" "0"
            | "if" ident ("=" | "!=") "0" "then" "goto" etykieta
            | "goto" etykieta
```

W zadaniu tym masz napisać procedurę `void instrukcja()` tłumaczącą instrukcję z wersji **A**, na ciąg instrukcji napisanych w wersji **B**. Procedura `instrukcja` powinna korzystać z globalnej zmiennej `token` typu `string`, wskazującej na symbol, który aktualnie powinien być zanalizowany. Możesz korzystać z:

- funkcji `getToken()` czytającej z wejścia do zmiennej globalnej `token` oraz z funkcji `putToken()` wypisującej na wyjście globalną zmienną `token`,
- funkcji `copyToken(int n) { for (int i=0; i<n; i++) { putToken(); getToken();}}`
- funkcji `isIdent(string)` sprawdzającej, czy argument jest identyfikatorem (nazwą zmiennej) oraz funkcji `isLabel(string)` sprawdzającej, czy argument jest etykietą.
- funkcji `string newLabel()`, generującej za każdym wywołaniem unikalną, nie występującą nigdzie wcześniej etykietę.

Program powinien działać poprawnie przy założeniu, że kompilowany program nie zawiera błędów składniowych.

Zadanie 5. Gramatyki G_1, G_2, G_3 nad alfabetem $\{a, b\}$, określone są, odpowiednio, przez zbiory produkcji P_1, P_2, P_3, gdzie

$$P_1 = \{S \to aSbS,\ S \to aS\ S \to \varepsilon\}$$
$$P_2 = \{S \to aS,\ S \to SS,\ S \to \varepsilon,\ S \to aSb,\ S \to bSa\}\}$$
$$P_3 = \{S \to aS,\ S \to \varepsilon,\ S \to aSb,\ S \to bSa\}$$

Ponadto niech $R = (a(\varepsilon + b + ab))^*$. Pokaż, że

a) **(4p)** Jeżeli $w \in L(G_1)$ oraz $w = w_1w_2$ to waga$(w_1) \geq 0$, gdzie waga$(w) = |w|_a - |w|_b$.

b) **(3p)** $L(G_2) \neq L(G_3)$

c) **(4p)** $R \subseteq L(G_2)$

Zadanie 6. Poniższa gramatyka w notacji BNF opisuje fragment składni języka C.

```
<deklaracja> ::= <typ> <deklarator> ;
       <typ> ::= int | char ;
<deklarator> ::= *<deklarator> | <deklarator>[<liczba>]
               | <deklarator> ( <typ> ) | ( <deklarator> )
               | <identyfikator>
```

a) **(2p)** Pokaż, że powyższa gramatyka nie jest jednoznaczna.

b) **(4p)** Podaj jednoznaczną gramatykę generującą ten sam język.