Programowanie
Pierwsza część egzaminu

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 24 punkty.

Zadanie 1. (**15p**) Gramatyka G_1 nad alfabetem $\{a, b\}$, określona jest przez przez zbióry produkcji P

$$P_1 = \{S \to aSbS,\ S \to bSaS,\ S \to \varepsilon\}$$

Niech język L_1 będzie zbiorem tych słów nad alfabetem $\{a, b\}$, w których liczba wystąpień literki a równa jest liczbie wystąpień literki b.

a) Czy G jest jednoznaczna? (Odpowiedź uzasadnij, (**2p**))
b) $L(G) \subseteq L_1$. Udowodnij lub obal to twierdzenie (**5p**).
c) $L(G) \supseteq L_1$. Udowodnij lub obal to twierdzenie (**5p**).
d) Podaj gramatykę bezkontekstową definiującą język (**3p**) $L_1 \cap \mathcal{L}(a^*b^* + b^*a^* + (ab)^*)$.

Zadanie 2. (**12p**) Zamień poniższe programy na ich odpowiedniki, w których nie ma instrukcji `goto`, a jedynymi strukturami sterującymi są pętla `do-while` i instrukcja `if`. Możesz wprowadzać nowe zmienne, przypisywać im wartości i sprawdzać je.

Występujące w programach warunki nie wywołują efektów ubocznych, podobnie można założyć że funkcje `g` oraz `h` z podpunktu c) nie mają żadnych efektów ubocznych.

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

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

c)
```
 int p(int x) {
    if (x==0) return 1;
    int y=p(x-1);
    return g(y)+h(y)
 }
```

Zadanie 3. (**13p**) W tym zadaniu powinieneś używać języka Python.

a) Napisz funkcję `mult` przekształcającą listę liczb naturalnych, na listę wartości funkcji `f` dla tych liczb. Funkcja `f` zamienia liczbę `k` na listę `[k,k,...,k]` o długości k. (**4p**)
b) Będziemy zapisywać w Pythonie wyrażenia arytmetyczne za pomocą list (tak jak w Lispie). Przykładowo `1+2+3*4` zapiszemy jako `['+',1,2,['*',3,4]]`. Napisz funkcję `zgrupuj`, która przekształca tak zapisane wyrażenie na równoważne, w którym czynniki i składniki zostały zgrupowane, z wykorzystaniem praw łączności mnożenia i dodawania (**9p**).
   Przykładowo `zgrupuj(['+',2,3,['+', 4,5,['*',2,['*',3,5]]]]` powinno dać `['+',2,3,4,5,['*',2,3,4]]`. (**8p**)

Dla osób, które chcą sobie przypomnieć składnię Pythona zapis poniższej sesji:

```
>>> x=[1,2,3,4,5]
>>> x[0]
1
>>> x[1:]
[2, 3, 4, 5]
>>> x[1:4]
[2,3,4]
>>> y=[]
>>> for v in x:
```

```
...     y.append(v)
...     y.extend([0,0])
>>> y
[1, 0, 0, 2, 0, 0, 3, 0, 0, 4, 0, 0, 5, 0, 0]
>>> type(1)==int
1
>>> type(y)==list
1
>>> type('abc') != str
0
 >>> def funkcja(x):
...     rv=[]
...     rv.extend([x,x])
...     return rv
>>> funkcja(5)
[5, 5]
```

Zadanie 4. (**11p**) Uzupełnij poniższy program tak, by realizował on w miejscu procedurę Podział z szybkiego algorytmu sortowania (czyli przestawiał elementy tablicy tak, by mniejsze od `r` znalazły się przed większymi od `r`.

```
i=1; j=n;
r=a[(m+n)/2];
while ( (1) ) {
   while (a[i]<r) (2);
   while (a[j]>j) (3);
   if (i <= j) {
      (4)
   }
}
```

Fragmenty `(1)`, `(2)`, `(3)` warte są 1 punkt, fragment `(4)` 2 punkty. Dodatkowe 2 punkty otrzymuje się za prawidłowe wypełnienie wszystkich fragmentów.

Podaj warunek poprawności częściowej tego programu (**2p**) i niezmiennik głównej pętli (**2p**) [1].

Zadanie 5. 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, w każdym podpunkcie można zdobyć 2 punkty za wskazanie błędu i 2 za przedstawienie poprawnego programu):

1. Program liczący wartość $n!$ dla nieujemnego n

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

2. Kopiowanie łańcucha znaków z `dest` do `src`.

```
void strcpy(char *dest, char *src) {
    int i;

    for (i=0; srec[i] != '\0'; i++)
       dest[i] = src[i];
}
```

3. Wyszukiwanie binarne w posortowanej tablicy `a`, zawierającej N liczb, pierwszy jej element znajduje się na pozycji `0`. Program powinien się zawsze kończyć i po zakończeniu `a[k]` powinno być równe `x` wtedy i tylko wtedy, gdy `x` znajduje się w tablicy `a`.

```
i=0; j=N-1;
do {
   k= (i+j) / 2;
   if (x<a[k]) j=k-1;
   if (a[k]<x) i=k+1;
} while (i<j)
```

[1] W sformułowaniu obu warunków możesz wspomóc się językiem naturalnym, ważne jest w takiej sytuacji prezyzyjne jego użycie.