Wrocław, dn. 02.11.2017

Paradygmaty programowania - studia stacjonarne, lista 4, 02.11.2017, godzina 13:15
Czas na rozwiązanie: 70 min.

WAŻNE1: Do każdego zadania przygotuj 3-5 testów sprawdzających poprawność działania!
WAŻNE2: Nie wolno stosować funkcji bibliotecznych i konstrukcji innych niż te przedstawione na wykładzie 3.

1. Napisz funkcję dekodującą listę par - liczba wystąpień danego elementu, element - do listy tych elementów.
Zwróć szczególną uwagę na optymalność złożoności obliczeniowej i pamięciowej.

Przykład:
wywołanie: dekoduj [(3,'a');(4,'b');(1,'c')]
wynik: ['a','a','a','b','b','b','b','c']

Punkty: 4 (język Ocaml)

2. Napisz funkcję kodującą listę do listy par - liczba wystąpień danego elementu, element.
Zwróć szczególną uwagę na optymalność złożoności obliczeniowej i pamięciowej.

Przykład:
wywołanie: koduj ['a','a','a','b','b','b','b','c']
wynik: [(3,'a');(4,'b');(1,'c')]

Punkty: 7 (język Scala)

3. Zdefiniuj funkcje sortowania przez wstawianie z zachowaniem stabilności i złożoności $O(n^2)$. Pierwszy argument jest funkcją, sprawdzającą porządek.
Przygotuj przykłady wywołania, które:

- sortują rosnąco listę punktów 2-wymiarowych według ich odległość od punktu (0,0),
- Sortują malejąco listę punktów 2-wymiarowych według ich odległość od punktu (0,0).

Zwróć szczególną uwagę na optymalność złożoności obliczeniowej i pamięciowej.

UWAGA! Skomentuj kroki algorytmu sortowania - brak komentarzy oznacza 0 punktów

Przykład:
wywołanie: `insertionsort distance [(1,2);(2,1);(1,1);(0,0)];;`
wynik: [(0,0);(1,1);(1,2);(2,1)]

Punkty: 9 (język Ocaml)