Задачи за подготовка за първо контролно по ФП,
специалност „Компютърни науки“

Задача 1. Да се дефинира функция **(sum-numbers a b)**, приемаща два аргумента, която намира сумата на числата в интервала **[a,b]**, чиито цифри са в низходящ (>=) ред.
Примери:
```
(sum-numbers 1 9) → 45
(sum-numbers 199 203) → 200
(sum-numbers 219 225) → 663
```

Задача 2. Да се дефинира функция **(num-bigger-elements lst)**, която за даден списък от числа **lst** връща като резултат списък с елементи от вида **(lst$_i$ n$_i$)**, където **lst$_i$** е **i**-тият елемент на **lst**, а **n$_i$** е броят на елементите на **lst**, които са по-големи от **lst$_i$**.
Примери:
```
(num-bigger-elements '(5 6 3 4)) → '((5 1) (6 0) (3 3) (4 2))
(num-bigger-elements '(1 1 1)) → '((1 0) (1 0) (1 0))
```

Задача 3. Ако **f** и **g** са числови функции и **n** е естествено число, да се дефинира функция от по-висок ред **(switchsum f g n)**, която връща като резултат функция, чиято стойност в дадена точка **x** е равна на **f(x)+g(f(x))+f(g(f(x)))+…** (сумата включва **n** събираеми).
Примери:
```
((switchsum (lambda (x) (+ x 1))
            (lambda (x) (* x 2)) 1) 2) → 3
((switchsum (lambda (x) (+ x 1))
            (lambda (x) (* x 2)) 2) 2) → 9
((switchsum (lambda (x) (+ x 1))
            (lambda (x) (* x 2)) 3) 2) → 16
((switchsum (lambda (x) (+ x 1))
            (lambda (x) (* x 2)) 4) 2) → 30
```

Задача 4. Да се дефинира функция **(repeater str)**, която получава като аргумент символен низ и връща анонимна функция на два аргумента - **count** и **glue** (число и низ). Оценката на обръщението към върнатата функция е низ, който се получава чрез **count**-кратно повтаряне на низа **str**, при което между всеки две съседни повторения на **str** стои низът **glue**.
Примери:
```
> ((repeater "I love Racket") 3 " ")
"I love Racket I love Racket I love Racket"
> ((repeater "Quack") 5 "!")
"Quack!Quack!Quack!Quack!Quack"
```
Помощна информация. За да съедините няколко низа, може да използвате вградената функцията **string-append**:
```
> (string-append "I" "Love" "Racket")
"ILoveRacket"
```
Функцията **string-append** приема произволен брой агрументи и връща низ, който представлява тяхната конкатенация.

Задача 5. Да се дефинира функция **(sum-sum-digit a b k)**, която намира сумата на естествените числа от **a** до **b** (0<a≤b), сумата от цифрите на които е кратна на **k**.

Задача 6. Да се дефинира функция **(max-ordered-sublist lst)**, която намира най-дългия възходящо сортиран подсписък на списъка от числа **lst**.
Пример:
```
(max-ordered-sublist '(1 5 2 4 6 8 3 4 1)) → '(2 4 6 8)
```

Задача 7. Да се дефинира функция **`(where list-elements list-predicates`***)*, която връща списък от всички елементи на **`list-elements`**, за които са изпълнени всички предикати в **`list-predicates`**.
Примери:
`(where '(3 4 5 6 7 8 9 10) (list even? (lambda (x) (> x 5))))` → **`(6 8 10`**) (списък от всички елементи на дадения, които са четни числа, по-големи от 5)
`(where '(3 4 5 7) (list even? (lambda (x) (> x 5))))` → () (в списъка няма четни числа, по-големи от 5)

Задача 8. Да се дефинира функция **`(set-union xs ys)`**, която връща обединението на множествата от числа **`xs`** и **`ys`**, представени като списъци, наредени във възходящ ред. Елементите на резултантното множество също трябва да са наредени във възходящ ред.
Примери:
`(set-union '(1 3 5 7) '(5 7 13))` → **`'(1 3 5 7 13)`**
`(set-union '(5 7 13) '(1 3 5 7))` → **`'(1 3 5 7 13)`**