



Модул 11. Функционално програмиране

Тема 1. Въведение

Задача 1.1. Първо и фамилно име

Дефинирайте функция, която чете от конзолата първо име, след това фамилно име и принтира двете имена на един рег.

Вход	Изход
Danail Iliev	Danail Iliev
Kristiyan Petrov	Kristiyan Petrov

Подсказки

1. Използвайте do-block
 - а. Четете стойностите с функцията ``getline`` и ги съхранявайте в променлива
 - б. Принтирайте конкатенирания символен низ на конзолата с функцията ``putStrLn``

Задача 1.2. Умножение на числа

Дефинирайте функция, която чете от конзолата последователно две числа и принтира резултата от умножението им.

Вход	Изход
10 20	200
50 50	2500
50 0	0



Подсказки

1. Използвайте функцията ``read``, за да преобразувате текста, подаден от конзолата, в числен тип и да направите пресмятанията
2. Използвайте функцията ``show`` преди да принтирате, за да преобразувате числения тип в символен низ

Задача 1.3 Лице на кръг

Дефинирайте функция, която чете от конзолата число - радиус на окръжност и намира лицето на кръга, около който е описана окръжността.

Вход	Изход
10.5	346.3606
0	0.0
9.98	312.90387

Подсказки

1. Използвайте функцията ``read``, за да преобразувате текста, подаден от конзолата, в числен тип и да направите пресмятанията
2. Използвайте функцията ``pi``, която връща числото π

Тема 2. Функции и стойности

Задача 2.1. Създайте функция

Дефинирайте функция, която удвоява стойност.

Вход	Изход
5	10

Подсказки

Дефинирайте функция `doubleVal`, приемаща 1 параметър `val` и връща като резултат `val + val`

Задача 2.2. Проверка за четно число

Дефинирайте функция, която проверява дали дадено цяло число е четно или нечетно.

Вход	Изход
5	False



10	True
----	------

Подсказки

1. Дефинирайте функцията `isEven`, приемаща 1 параметър `val` и връщаща като резултат `True`, ако числото е четно и `False`, ако не е
2. Разгледайте как работи вградената в Haskell функция `mod` и я използвайте в тялото на вашата функция

Задача 2.3. Най-голямото от три числа

Дефинирайте функция, която приема като параметри 3 числа и връща най-голямото от тях.

Вход	Изход
5 10 15	15
15 24 11	24

Подсказки

1. Дефинирайте функция `biggestOf3` приемаща 3 параметъра и връщаща най-големия от тях като резултат
2. Проверете дали първият параметър `a` е по-голям от вторият параметър `b`
 - a. Ако е проверете дали първият параметър `a` е по-голям и от третият параметър `c`
 - i. Ако е върнете `a`
 - ii. Ако не е върнет е `c`
 - b. Ако `a` не е по-голям от `b` проверете дали `b` е по-голям от `c`
 - i. Ако е върнете `b`
 - ii. Ако не е върнете `c`

Задача 2.4. Функция, изпълняваща друга функция

Дефинирайте две функции, едната от които да приема като параметър число и да прибавя 1 към него. Нека другата приема като параметър функция, както и число, с което да се извика получената функция.

Вход	Изход
<code>add1 5</code>	6
<code>remove1 7</code>	6



Подсказки

1. Дефинирайте функции, които главната функция ще извиква
 - a. Например функция `add1` приемаща 1 параметър - число и връщаща като резултат числото + 1
 - b. Функция `remove1` приемаща 1 параметър - число и връщаща като резултат числото - 1
2. Дефинирайте функция `execute` приемаща като параметри функция както и число, с което да извика приетата функция
 - a. Като резултат върнете приетата функция с подаден параметър полученото число

Задача 2.5. Факториел

Дефинирайте функция, която приема като параметър число n и връща като резултат n факториел

Вход	Изход
5	120
10	3628800

Подсказки

1. Дефинирайте функция `factorial` приемаща 1 параметър n
2. При $n = 1$ или $n = 0$ върнете като резултат 1 (дъно на рекурсията)
3. В противен случай върнете n умножено по-резултата получен при извикване на същата функция за $n - 1$

Задача 2.6. Редицата на Фибоначи

Дефинирайте функция, която приема като параметър число n и връща като резултат n -тото число от редицата на Фибоначи. Редицата на Фибоначи започва от 1 и всяко следващо число е равно на сбора от предишните две - например второто число от редицата е равно на $1 + 0$ (за нулево число от редицата се приема 0), третото число от редицата е равно на второто + първото ($1 + 1$) = 2 и тн.

Вход	Изход
10	55



21	10946
----	-------

Подсказки

1. Дефинирайте функция `fibonacci` приемаща един параметър `n` и връщаща `n`-тото число от редицата като резултат
2. При `n = 1` или `n = 2` върнете като резултат 1 (гъно на рекурсията)
3. В противен случай за `n` върнете като полученият резултат от извикването на същата функция за `n - 1` + резултат от извикването на същата функция за `n - 2` (двете предишни числа от редицата на фибоначи)

Тема 3. Рекурсия

Задача 3.1. Логаритъм втори от `n`

Дефинирайте функция, която приема един параметър - число и връща като резултат логаритъм втори от подаденото число (закръглено до целочислен тип)

Вход	Изход
15	3
10	3
10000	13

Подсказки

1. Дефинирайте функцията като за 1 нека връща резултат 0
2. При `n > 1` върнете резултат `1 + резултата рекурсивното извикване на същата функция за $n / 2$`
 - а. По този начин се постига цикличен ефект (подобно на `for-loop`), като началото е `n`, условието е `n > 1`, а на всяка итерация `n` намалява двойно

Задача 3.2. Факториел (опашкова рекурсия)

Дефинирайте функция, която връща като резултат `n`-факториел. Използвайте алгоритъм с опашкова рекурсия

Вход	Изход
5	120
10	3628800



Подсказки

1. Дефинирайте функция `findFactorial`, която приема 3 параметъра
 - a. `n` - желания факториел
 - b. `initialValue` - началната стойност (факториела започва от 1)
 - c. `Index` - индекс точно, какъвто би се използвал в нормален `for`-цикъл
2. Проверете дали индекса надвишава желаното число `n`
 - a. Ако да - върнете `initialValue`
 - b. Ако не - рекурсивно извикайте `findFactorial` като подгетем същия `n`, промените `initialValue` на стойност равна на сегашната умножено по индекса, увеличете индекса с единица
3. Дефинирайте втора функция `factorial`, която да служи за помощна и да приема само 1 параметър - `n` - желания факториел
4. Нека `factorial` извиква функцията `findFactorial` като задава за стойности на нейните параметри `n` за търсеното число, 1 за стартова стойност и 1 като начален индекс

Задача 3.3. Фибоначи (опашкова рекурсия)

Дефинирайте функция, която връща като резултат `n`-тото число от редицата на Фибоначи. Използвайте алгоритъм с опашкова рекурсия.

Вход	Изход
10	55
21	10946

Подсказки

1. Дефинирайте функцията `findFibonacci`. Нека тя приема 4 параметъра - `n` - желаното по ред число от редицата на Фибоначи, `initialValue` - стойността, от която редицата започва, `prevValue` - стойността на предишното по ред число от редицата, `index` - индексатор, който следи до кое число от редицата сме стигнали
2. Проверете дали индексаторът не е надвишил или е равен на желаното по ред число - `n`
 - a. Ако да - нека функцията върне началната стойност - `initialValue`



- б. Ако не - нека функцията се извика рекурсивно като стойността на n се запазва, началната стойност вече е равна началната стойност плюс предишното по ред число. За стойност на предишното число вече ни е нужно `initialValue`, а индекса трябва да се увеличи с единица, за да напредне рекурсията към дъното си
3. Дефинирайте помощна функция `fibonacci`, която приема един параметър - n и извиква функцията `findFibonacci` като задава стойности за n - n , за начална стойност - единица, за предишно по ред число - 0 и за индекс - 1

Задача 3.4. Обърнат триъгълник

Дефинирайте функция, която връща приема като параметър число - n и принтира на конзолата обрнат триъгълник от "*", като на се започне на първи ред с n звездички и на всеки следващ ред принтира с една по-малко. При вход 0 да не се принтира нищо на конзолата

Вход	Изход
5	***** **** *** ** *
1	*
4	**** *** ** *

Подсказки

1. Дефинирайте помощна функция `asterixStringRow`, която приема един параметър - броя на символи, които трябва да се повтарят и връща като резултат символен низ с дължина n , съставен от звездички ("*")



- a. Разгледайте как работи вградения метод `replace`
2. Дефинирайте функцията `printTriangle`, която приема `n`
 - a. За ``printTriangle 0`` функцията не трябва да връща нищо (Разгледайте `void` типа в Haskell - `()` и го използвайте като резултат от функцията)
 - b. За всяко друго `n` нека функцията принтира резултата от `asterixStringRow` за `n`, след което рекурсивно извиква себе си за `n - 1`

Тема 4. Списъци

Задача 4.1. Обръщане на списък

Дефинирайте функция, която приема списък и връща като резултат списъкът в обратен ред.

Вход	Изход
[1,2,3,4,5]	[5,4,3,2,1]
[1]	[1]
[]	[]

Подсказки

1. Проучете как работи `(x : xs)` pattern matching-а в Haskell
2. Използвайте рекурсивно ``++`` оператора между опашката и главата на списъка

Задача 4.2. Дължина на списък

Дефинирайте собствена функция, която приема списък и връща като резултат броят на елементите в списъка.

Вход	Изход
[123,456]	2
[1]	1
[]	0
"Hello, world!"	13



Подсказки

1. Обхождатے рекурсивно опашката на списък като за всяко извикване връщайте 1 + резултата от рекурсията

Задача 4.3. Дублиране на елементи от списък

Дефинирайте функция, която приема списък и връща като резултат списъкът с дублирани елементи.

Вход	Изход
"abc"	"aabbcc"
[1,2,3]	[1,1,2,2,3,3]
[1]	[1,1]
[]	[]

Задача 4.4. Премахване на всеки n-ти елемент

Дефинирайте функция, която приема списък и число n и връща като резултат списък с премахнат всеки n-ти елемент от началния списък

Вход	Изход
[1,2,3,4,5,6,7,8,9] 3	[1,2,4,5,7,8]
[1,2,3,4,5,6,7,8,9] 1	[]
[1,2,3] 3	[1,2]
[] 3	[]

Подсказки

1. Проучете как работят вградените в Haskell функции `take` и `drop`

Задача 4.5. Фибоначи (списък)

Дефинирайте функция, която приема число - n връща като резултат списък с елементите от редицата на Фибоначи от 1 до n.

Вход	Изход
10	[1,1,2,3,5,8,13,21,34,55]
15	[1,1,2,3,5,8,13,21,34,55,89,144,233,377,610]
1	[1]



0	[]
-1	[]

Задача 4.6. Факториел (списък)

Дефинирайте функция, която приема число - n връща като резултат списък с елементи съответно факториел на числата от 1 до n.

Вход	Изход
10	[1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]
5	[1, 1, 2, 6, 24]
0	[]
1	[1]

Тема 5. Функции от по висок ред

Задача 5.1. Генериране на математически израз

Дефинирайте функция, която приема списък от числа и генерира математически израз в следния формат:

$((a + b) + c) + d$

, където a,b,c,d са елементите на подадения списък ([a,b,c,d])

Вход	Изход
[1, 2, 3, 4, 5]	"(((1+2)+3)+4)+5"
[1]	"1"
[1, 10]	"(1+10)"
[]	""

Подсказки

1. Изполвайте `fold` функция, за да преминете през всички елементи от списъка
2. Дефинирайте помощна функция, която да приема 2 аргумента и да връща като резултат форматиран символен низ от тип
 - a. "(a+b)"
 - b. "б" - при a - празен символен низ



Задача 5.2. Генериране на математически израз

Дефинирайте функция, която приема списък от числа и генерира математически израз в следния формат:

$$(a + (b + (c + d)))$$

, където a,b,c,d са елементите на подадения списък ([a,b,c,d])

Вход	Изход
[1,2,3,4,5]	"(1+(2+(3+(4+5))))"
[1]	"1"
[1,10]	"(1+10)"
[]	""

Подсказки

- Изполвайте `fold` функция, за да преминете през всички елементи от списъка
- Дефинирайте помощна функция, която да приема 2 аргумента и да връща като резултат форматиран символен низ от тип
 - "(a+b)"
 - "б" - при а - празен символен низ

Задача 5.3. Компресиране на списък

Дефинирайте функция, която приема списък и го компресира, като премахва повтарящите се последователни елементи:

Вход	Изход
[1,1,1,1,1,1,1,1,1,1,1,1,2,3,4,5,5,7,8]	[1,2,3,4,5,7,8]
[1]	[1]
[1,10]	[1,10]
[]	[]

Подсказки

- Използвайте `fold` функция за да обработите списъка



Задача 5.4. Дупликация на списъчни елементи

Дефинирайте функция, която приема списък и връща нов списък като дублира всеки елемент от него

Вход	Изход
[1,2,3,4]	[1,1,2,2,3,3,4,4]
[1,2,3,4,4]	[1,1,2,2,3,3,4,4,4,4]
[1]	[1,1]
[]	[]

Задача 5.5. Репликация на списъчни елементи

Дефинирайте функция, която приема списък и число - n и връща нов списък като репликира всеки елемент от него n на брой пъти

Вход	Изход
[1,2,3,4,5] 2	[1,1,2,2,3,3,4,4,5,5]
[1,2] 5	[1,1,1,1,1,2,2,2,2,2]
[1,2,3] 0	[]
[] 10	[]

Задача 5.6. Отрязване на списък

Дефинирайте функция, която приема списък, начален индекс и краен индекс и връща като резултат нов списък - елементите от началния до крайния индекс от първоначалния списък

Бележка: Ако крайният индекс надвишава дължината на списъка, функцията да връща всички елементи до края.

Вход	Изход
[1,2,3,4,5] 1 2	[2,3]
[1,2,3,4,5] 0 4	[1,2,3,4,5]
[1,2,3,4,5] 1 0	[]
[] 5 5	[]
[1,2,3,4] 0 10	[1,2,3,4]



Тема 6. Затваряне на състояние във функция

Задача 6.1. Умножение на числа

Дефинирайте функция, която да приема 2 числа и да връща резултат умножението им. Използвайте функция с вътрешно състояние, като единият аргумент трябва да идва извън рамките на анонимна функция

Вход	Изход
5 10	50

Задача 6.2. Най-голямо от три числа

Дефинирайте функция, която приема като параметри 3 числа и връща най-голямото от тях. Използвайте функция с вътрешно състояние. Нека 2 от аргументите функцията да приема отвън.

Вход	Изход
5 10 20	20
5 10 (-20)	10

Задача 6.3. Подаване на функция като аргумент на функция

Дефинирайте функция с вътрешно състояние, която да приема друга функция и параметър и да подава параметъра на функцията. Нека параметъра да е свободната променлива.

Вход	Изход
5 add1	6
5 remove1	4

Задача 6.4. Генериране на математически израз

Дефинирайте функция, която приема списък от числа и генерира математически израз в следния формат:

$((a + b) + c) + d$

, където a,b,c,d са елементите на подадения списък ([a,b,c,d]). Използвайте функция с вътрешно състояние за помощната функция, която форматира числата в символен низ

Вход	Изход
[1,2,3,4,5]	"(((1+2)+3)+4)+5"
[1]	"1"
[1,10]	"(1+10)"
[]	" "



Съдържание

Модул 11. Функционално програмиране.....	1
Тема 1. Въведение	1
Задача 1.1. Първо и фамилно име.....	1
Задача 1.2. Умножение на числа.....	1
Задача 1.3 Лице на кръг.....	2
Тема 2. Функции и стойности.....	2
Задача 2.1. Създайте функция.....	2
Задача 2.2. Проверка за четно число.....	2
Задача 2.3. Най-голямото от три числа.....	3
Задача 2.4. Функция, изпълняваща друга функция.....	3
Задача 2.5. Факториел.....	4
Задача 2.6. Редицата на Фибоначи.....	4
Тема 3. Рекурсия.....	5
Задача 3.1. Логаритъм втори от n.....	5
Задача 3.2. Факториел (опашкова рекурсия).....	5
Задача 3.3. Фибоначи (опашкова рекурсия).....	6
Задача 3.4. Обърнат триъгълник.....	7
Тема 4. Списъци.....	8
Задача 4.1. Обръщане на списък.....	8
Задача 4.2. Дължина на списък.....	8
Задача 4.3. Дубликиране на елементи от списък.....	9
Задача 4.4. Премахване на всеки n-ти елемент.....	9
Задача 4.5. Фибоначи (списък).....	9
Задача 4.6. Факториел (списък).....	10
Тема 5. Функции от по висок ред.....	10
Задача 5.1. Генериране на математически израз.....	10
Задача 5.2. Генериране на математически израз.....	11
Задача 5.3. Компресиране на списък.....	11
Задача 5.4. Дубликация на списъчни елементи.....	12
Задача 5.5. Репликация на списъчни елементи.....	12
Задача 5.6. Отрязване на списък.....	12
Тема 6. Затваряне на състояние във функция.....	13



Задача 6.1. Умножение на числа.....	13
Задача 6.2. Най-голямо от три числа.....	13
Задача 6.3. Подаване на функция като аргумент на функция	13
Задача 6.4. Генериране на математически израз.....	13