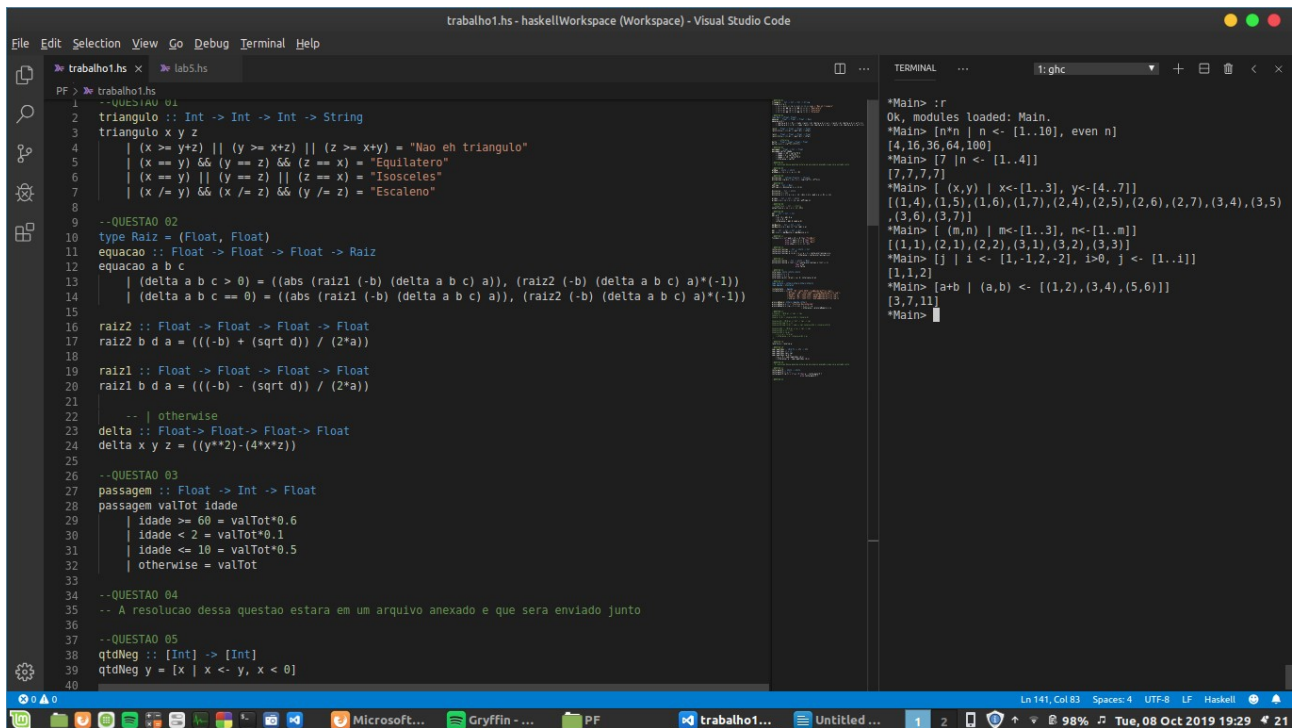


Anexo ao trabalho 1 de Programação Funcional 2019/2

QUESTÃO 04

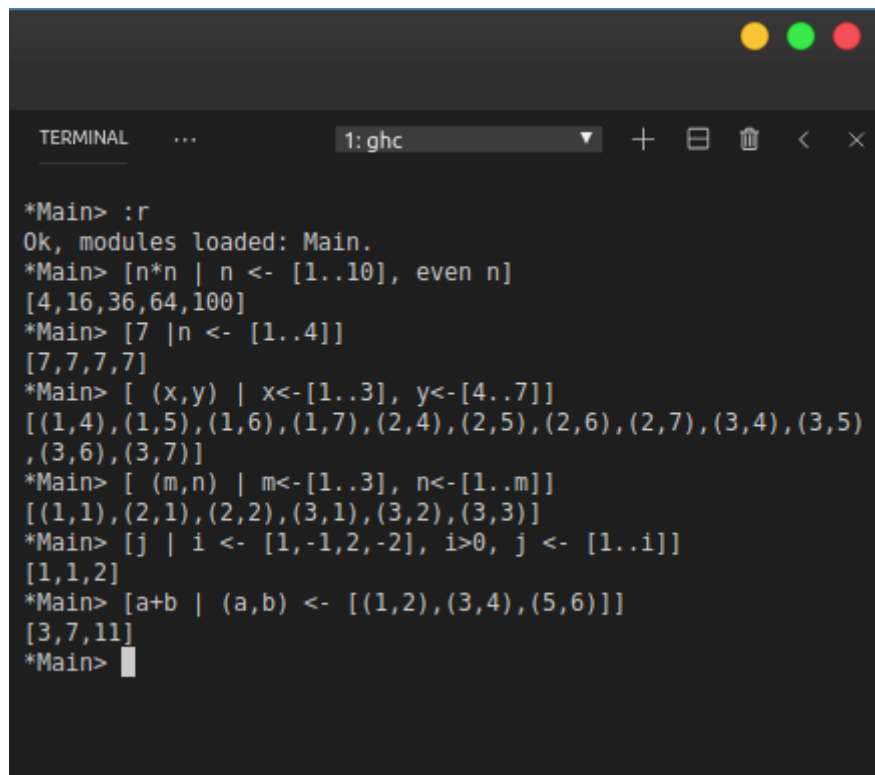


The image shows a Visual Studio Code editor window with a Haskell workspace. The editor displays the code for 'Questão 04', which includes functions for checking if a triangle is equilateral, isosceles, or scalene, and calculating the area of a triangle using Heron's formula. The code is written in Haskell and uses the `do` notation for monadic actions. The terminal window on the right shows the execution of the code using the `ghc` compiler. The output of the program is displayed in the terminal, showing the results of the calculations for various inputs.

```
trabalho1.hs
lab5.hs

PF > trabalho1.hs
1 --QUESTAO 01
2 triangulo :: Int -> Int -> Int -> String
3 triangulo x y z
4   | (x >= y+z) || (y >= x+z) || (z >= x+y) = "Nao eh triangulo"
5   | (x == y) && (y == z) && (z == x) = "Equilatero"
6   | (x == y) || (y == z) || (z == x) = "Isosceles"
7   | (x /= y) && (x /= z) && (y /= z) = "Escaleno"
8
9 --QUESTAO 02
10 type Raiz = (Float, Float)
11 equacao :: Float -> Float -> Float -> Raiz
12 equacao a b c
13   | (delta a b c > 0) = ((abs (raiz1 (-b) (delta a b c) a)), (raiz2 (-b) (delta a b c) a)*(-1))
14   | (delta a b c == 0) = ((abs (raiz1 (-b) (delta a b c) a)), (raiz2 (-b) (delta a b c) a)*(-1))
15
16 raiz2 :: Float -> Float -> Float -> Float
17 raiz2 b d a = (((-b) + (sqrt d)) / (2*a))
18
19 raiz1 :: Float -> Float -> Float -> Float
20 raiz1 b d a = (((-b) - (sqrt d)) / (2*a))
21
22 -- | otherwise
23 delta :: Float -> Float -> Float -> Float
24 delta x y z = ((y**2)-(4*x*z))
25
26 --QUESTAO 03
27 passagem :: Float -> Int -> Float
28 passagem valTot idade
29   | idade >= 60 = valTot*0.6
30   | idade < 2 = valTot*0.1
31   | idade <= 10 = valTot*0.5
32   | otherwise = valTot
33
34 --QUESTAO 04
35 -- A resolucao dessa questao estara em um arquivo anexado e que sera enviado junto
36
37 --QUESTAO 05
38 qtdNeg :: [Int] -> [Int]
39 qtdNeg y = [x | x <- y, x < 0]
```

```
*Main> :r
Ok, modules loaded: Main.
*Main> [n*n | n <- [1..10], even n]
[4,16,36,64,100]
*Main> [7 | n <- [1..4]]
[7,7,7,7]
*Main> [ (x,y) | x<-[1..3], y<-[4..7]]
[(1,4),(1,5),(1,6),(1,7),(2,4),(2,5),(2,6),(2,7),(3,4),(3,5),
(3,6),(3,7)]
*Main> [ (m,n) | m<-[1..3], n<-[1..m]]
[(1,1),(2,1),(2,2),(3,1),(3,2),(3,3)]
*Main> [j | i <- [1,-1,2,-2], i>0, j <- [1..i]]
[1,1,2]
*Main> [a+b | (a,b) <- [(1,2),(3,4),(5,6)]]
[3,7,11]
*Main>
```

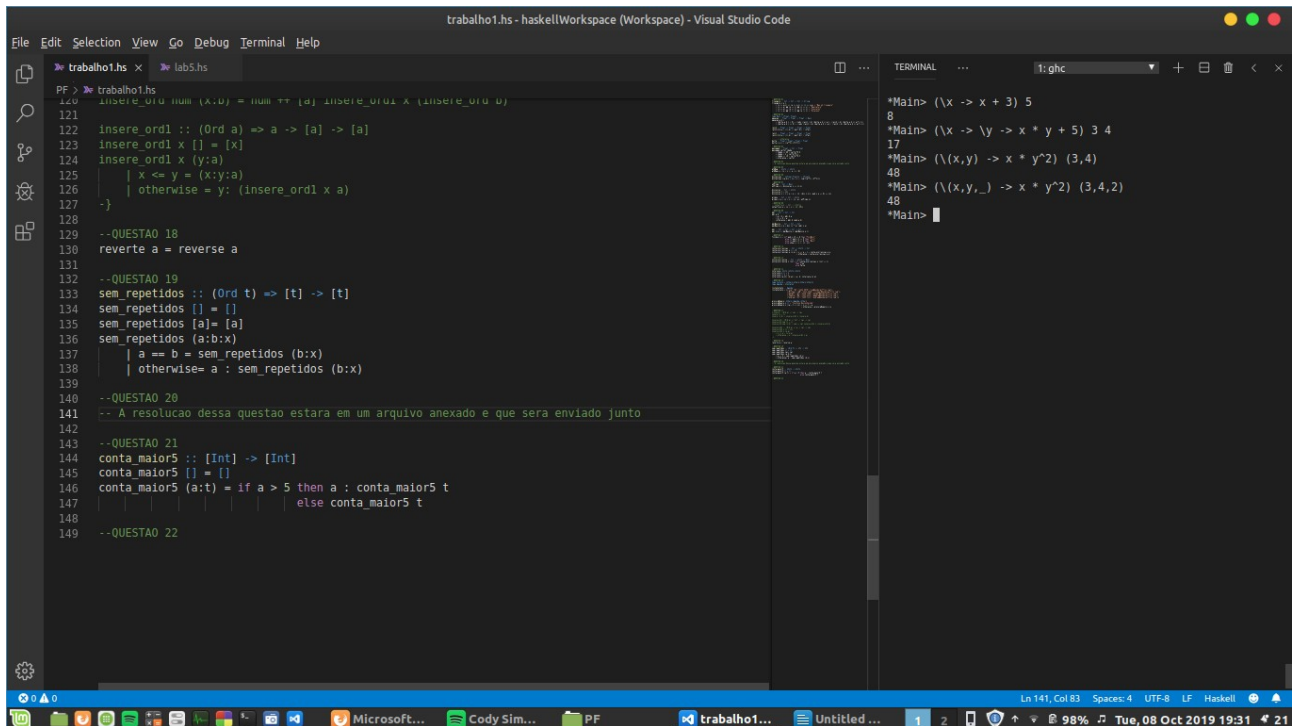


This is a close-up view of the terminal window from the previous image. It shows the same sequence of Haskell code being executed and the corresponding output. The terminal window has a title bar with standard window controls and a dropdown menu showing '1: ghc'.

```
TERMINAL 1: ghc

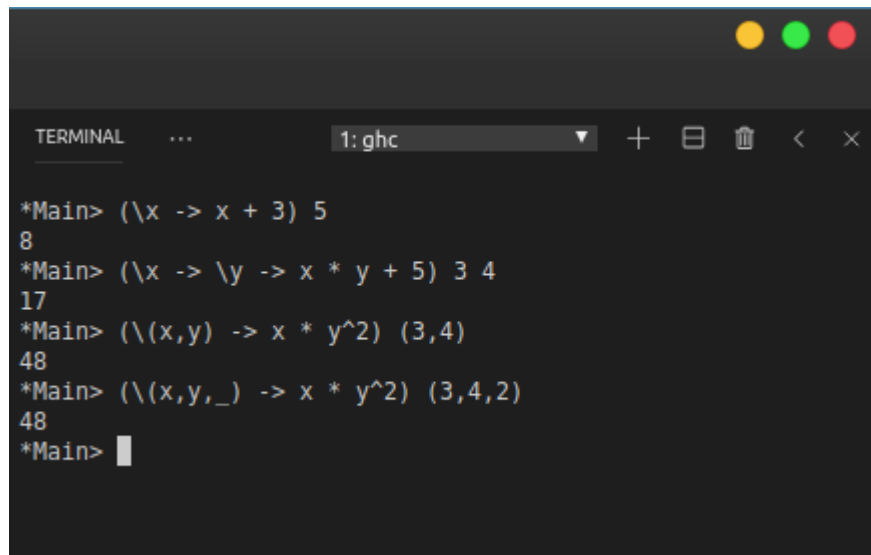
*Main> :r
Ok, modules loaded: Main.
*Main> [n*n | n <- [1..10], even n]
[4,16,36,64,100]
*Main> [7 | n <- [1..4]]
[7,7,7,7]
*Main> [ (x,y) | x<-[1..3], y<-[4..7]]
[(1,4),(1,5),(1,6),(1,7),(2,4),(2,5),(2,6),(2,7),(3,4),(3,5),
(3,6),(3,7)]
*Main> [ (m,n) | m<-[1..3], n<-[1..m]]
[(1,1),(2,1),(2,2),(3,1),(3,2),(3,3)]
*Main> [j | i <- [1,-1,2,-2], i>0, j <- [1..i]]
[1,1,2]
*Main> [a+b | (a,b) <- [(1,2),(3,4),(5,6)]]
[3,7,11]
*Main>
```

QUESTÃO 20



The screenshot shows the Visual Studio Code editor with a Haskell file named `trabalho1.hs`. The code defines several functions, including `insere_ord` and `sem_repetidos`. A comment indicates that the solution for 'QUESTÃO 20' is in an attached file. The terminal window on the right shows the execution of the program using `ghc`, with the following output:

```
*Main> (\x -> x + 3) 5
8
*Main> (\x -> \y -> x * y + 5) 3 4
17
*Main> (\(x,y) -> x * y^2) (3,4)
48
*Main> (\(x,y,_) -> x * y^2) (3,4,2)
48
*Main>
```



A close-up view of the terminal window from the previous image, showing the same Haskell execution results:

```
TERMINAL 1: ghc

*Main> (\x -> x + 3) 5
8
*Main> (\x -> \y -> x * y + 5) 3 4
17
*Main> (\(x,y) -> x * y^2) (3,4)
48
*Main> (\(x,y,_) -> x * y^2) (3,4,2)
48
*Main>
```