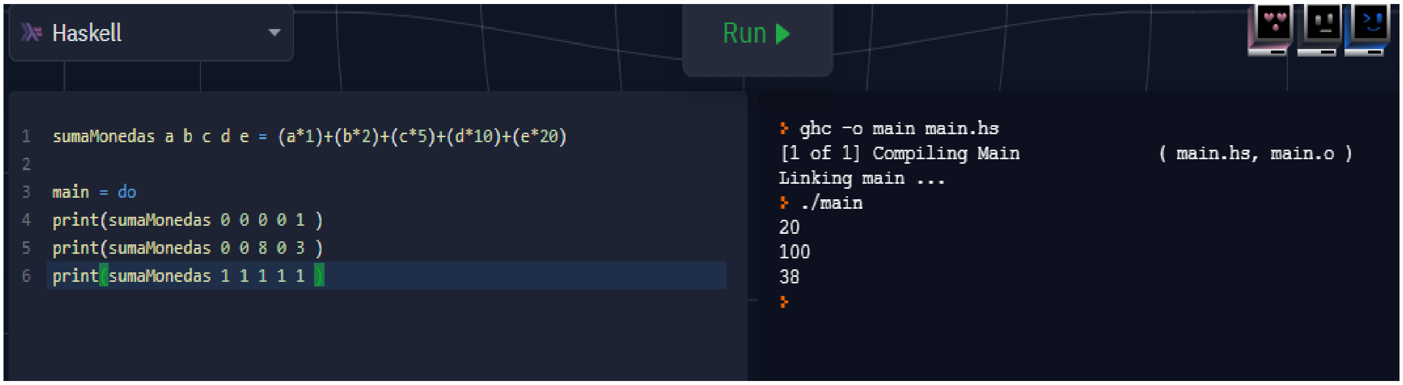
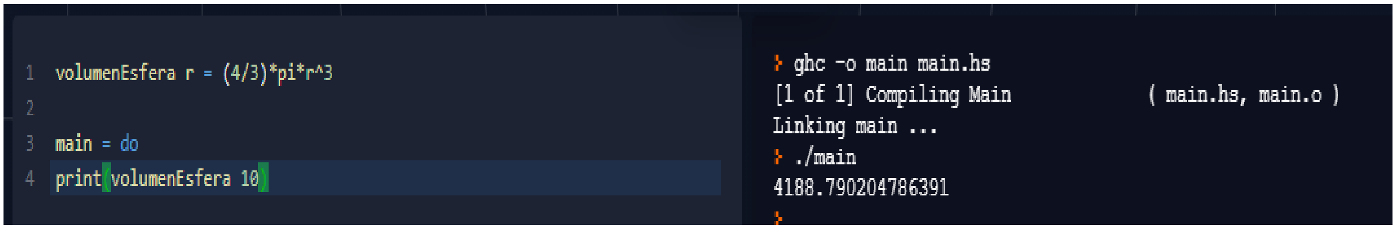
Ejercicio A



```haskell
sumaMonedas a b c d e = (a*1)+(b*2)+(c*5)+(d*10)+(e*20)

main = do
print(sumaMonedas 0 0 0 0 1 )
print(sumaMonedas 0 0 8 0 3 )
print(sumaMonedas 1 1 1 1 1 )
```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
› ./main
20
100
38
›
```

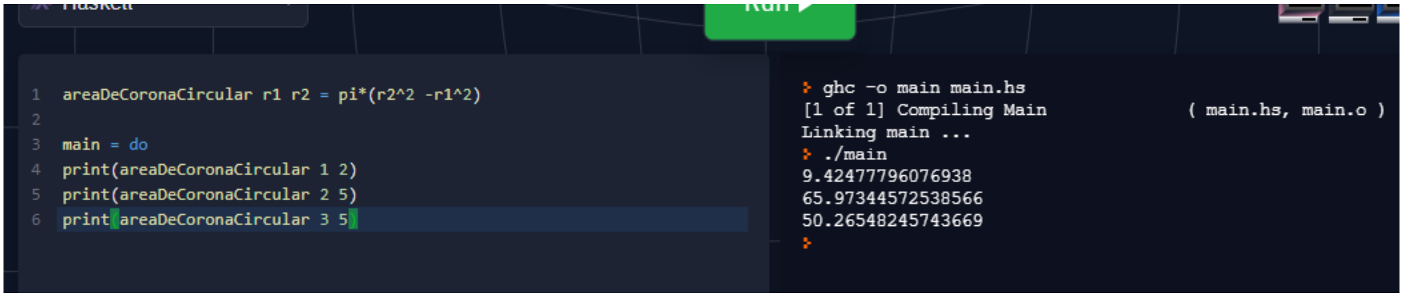
Ejercicio B



```haskell
volumenEsfera r = (4/3)*pi*r^3

main = do
print(volumenEsfera 10)
```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
› ./main
4188.790204786391
›
```

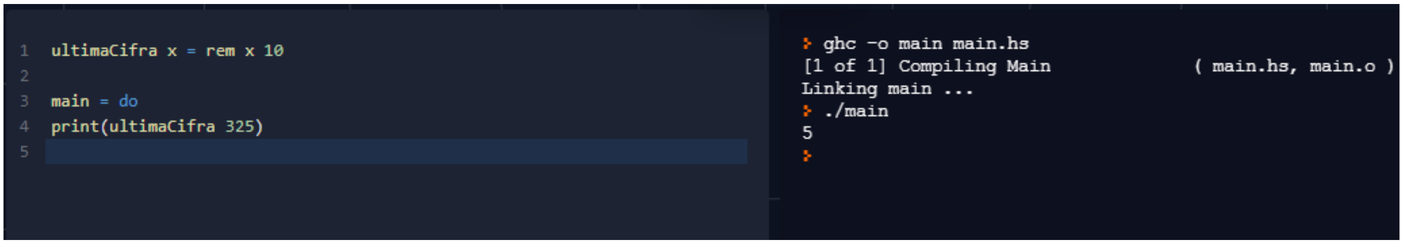
Ejercicio C



```haskell
areaDeCoronaCircular r1 r2 = pi*(r2^2 -r1^2)

main = do
print(areaDeCoronaCircular 1 2)
print(areaDeCoronaCircular 2 5)
print(areaDeCoronaCircular 3 5)
```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
› ./main
9.42477796076938
65.97344572538566
50.26548245743669
›
```

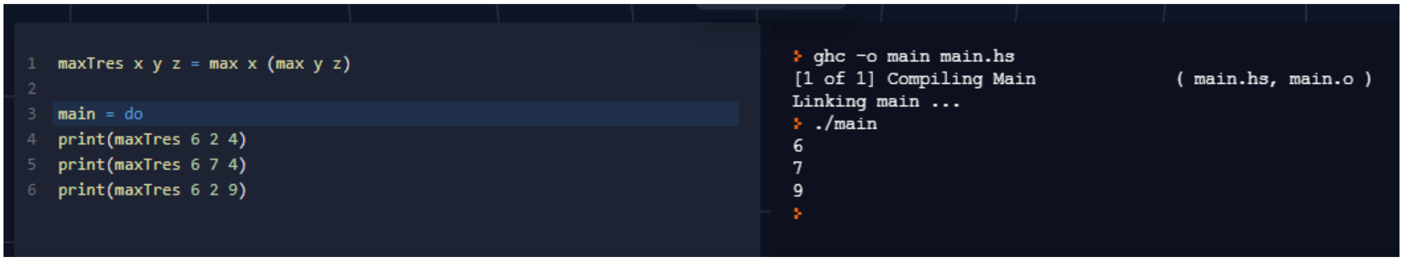
Ejercicio D



```haskell
ultimaCifra x = rem x 10

main = do
print(ultimaCifra 325)

```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
› ./main
5
›
```

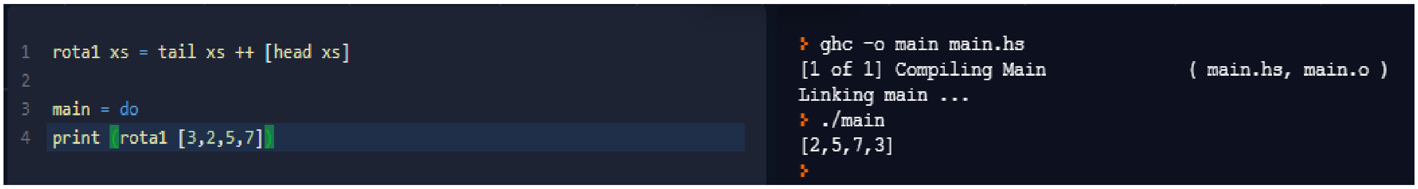
Ejercicio E



```haskell
maxTres x y z = max x (max y z)

main = do
print(maxTres 6 2 4)
print(maxTres 6 7 4)
print(maxTres 6 2 9)
```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
› ./main
6
7
9
›
```

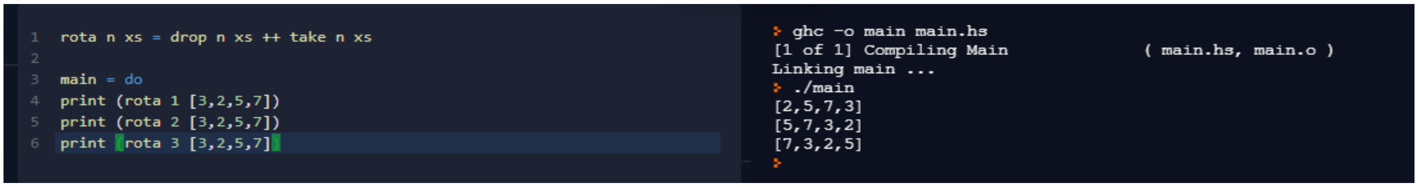
Ejercicio F



```haskell
rota1 xs = tail xs ++ [head xs]

main = do
print (rota1 [3,2,5,7])
```

```
> ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
> ./main
[2,5,7,3]
>
```

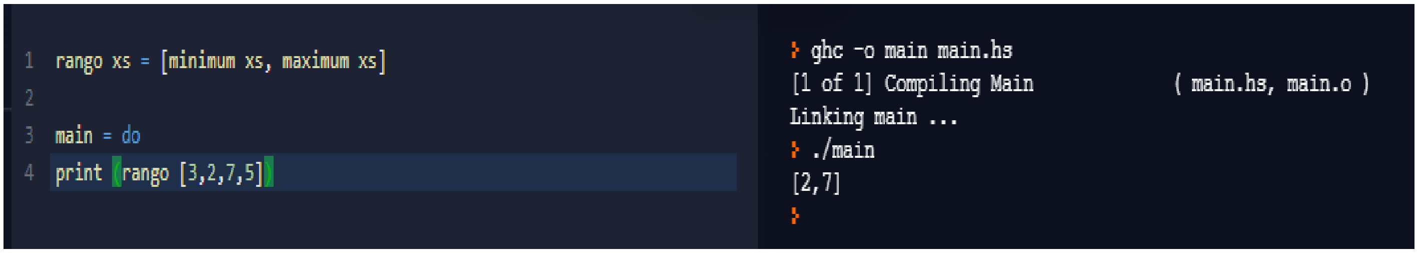
Ejercicio G



```haskell
rota n xs = drop n xs ++ take n xs

main = do
print (rota 1 [3,2,5,7])
print (rota 2 [3,2,5,7])
print (rota 3 [3,2,5,7])
```

```
> ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
> ./main
[2,5,7,3]
[5,7,3,2]
[7,3,2,5]
>
```

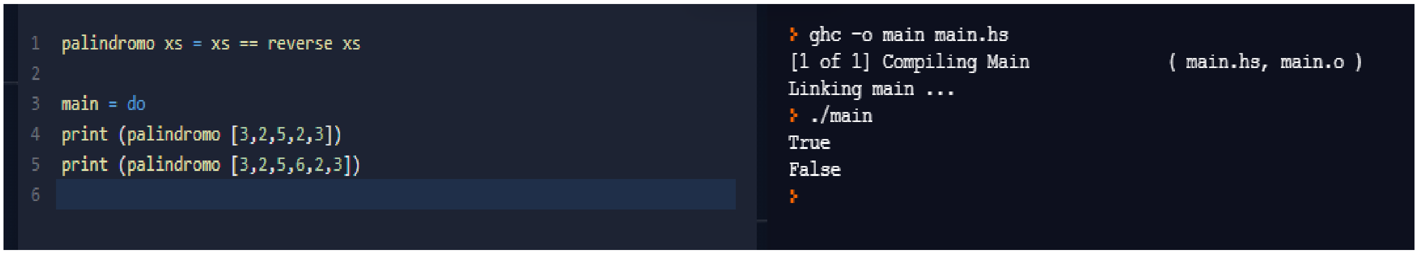
Ejercicio H



```haskell
rango xs = [minimum xs, maximum xs]

main = do
print (rango [3,2,7,5])
```

```
> ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
> ./main
[2,7]
>
```

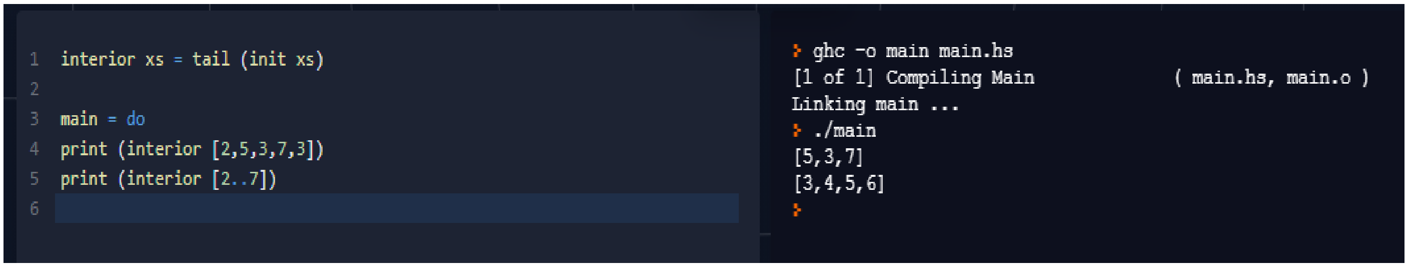
Ejercicio I



```haskell
palindromo xs = xs == reverse xs

main = do
print (palindromo [3,2,5,2,3])
print (palindromo [3,2,5,6,2,3])

```

```
> ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
> ./main
True
False
>
```

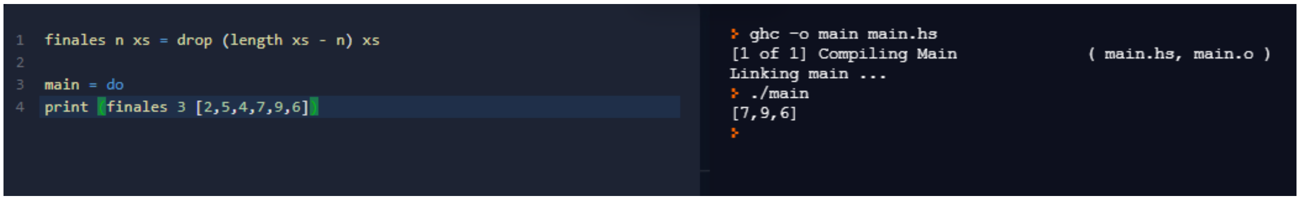
Ejercicio J



```haskell
interior xs = tail (init xs)

main = do
print (interior [2,5,3,7,3])
print (interior [2..7])

```

```
> ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
> ./main
[5,3,7]
[3,4,5,6]
>
```

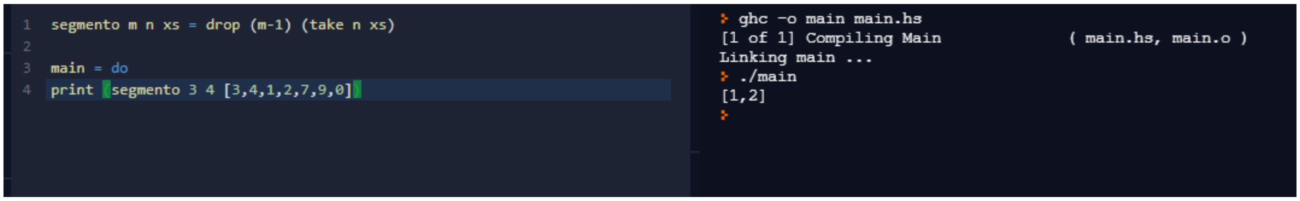
Ejercicio K



```haskell
finales n xs = drop (length xs - n) xs

main = do
print (finales 3 [2,5,4,7,9,6])
```

```
> ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
> ./main
[7,9,6]
>
```

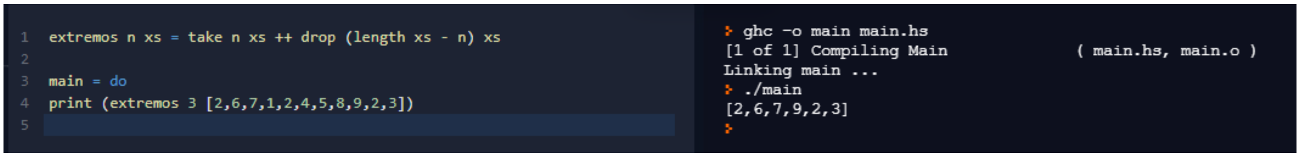
Ejercicio L



```haskell
segmento m n xs = drop (m-1) (take n xs)

main = do
print (segmento 3 4 [3,4,1,2,7,9,0])
```

```
❯ ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
❯ ./main
[1,2]
❯
```

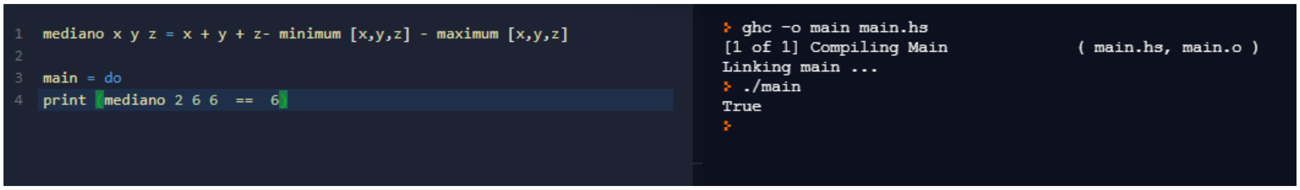
Ejercicio M



```haskell
extremos n xs = take n xs ++ drop (length xs - n) xs

main = do
print (extremos 3 [2,6,7,1,2,4,5,8,9,2,3])

```

```
❯ ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
❯ ./main
[2,6,7,9,2,3]
❯
```

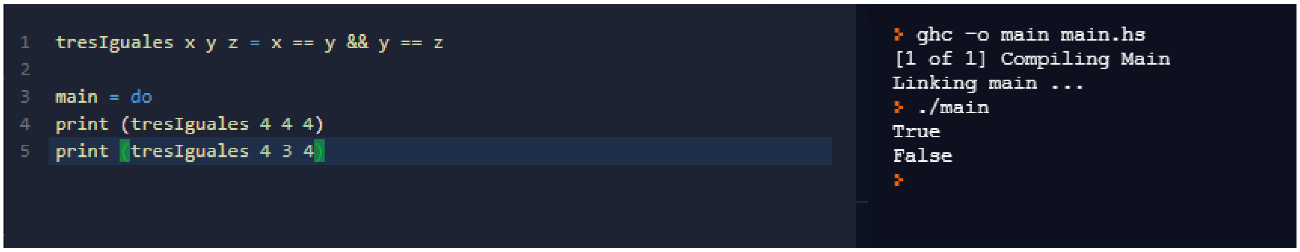
Ejercicio N



```haskell
mediano x y z = x + y + z- minimum [x,y,z] - maximum [x,y,z]

main = do
print (mediano 2 6 6  ==  6)
```

```
❯ ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
❯ ./main
True
❯
```

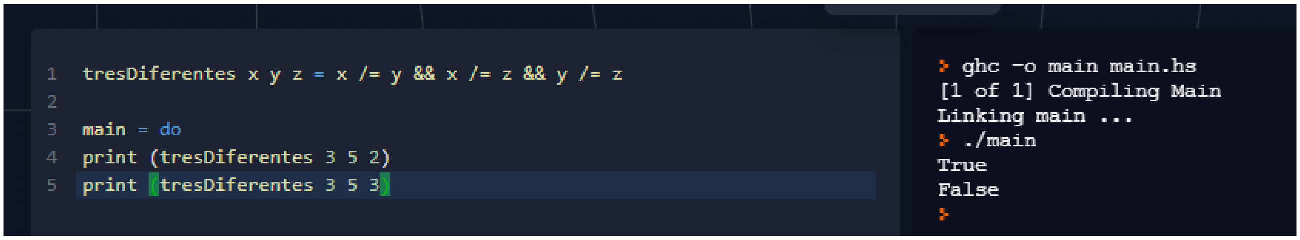
Ejercicio O



```haskell
tresIguales x y z = x == y && y == z

main = do
print (tresIguales 4 4 4)
print (tresIguales 4 3 4)
```

```
❯ ghc -o main main.hs
[1 of 1] Compiling Main
Linking main ...
❯ ./main
True
False
❯
```

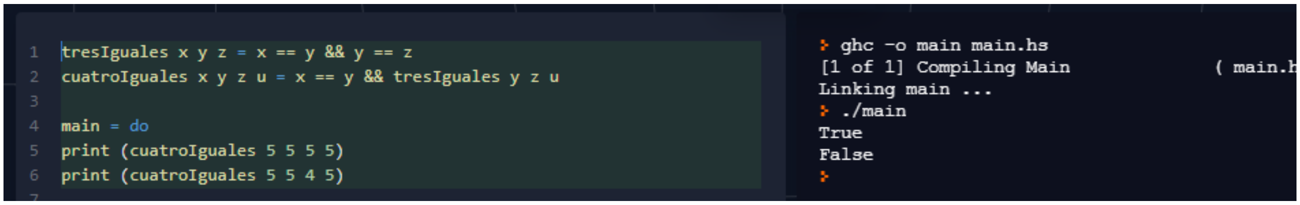
Ejercicio P



```haskell
tresDiferentes x y z = x /= y && x /= z && y /= z

main = do
print (tresDiferentes 3 5 2)
print (tresDiferentes 3 5 3)
```

```
❯ ghc -o main main.hs
[1 of 1] Compiling Main
Linking main ...
❯ ./main
True
False
❯
```

Ejercicio Q



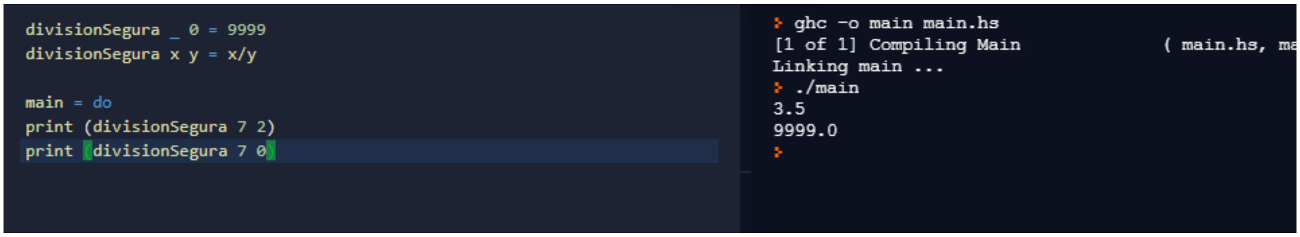
```haskell
tresIguales x y z = x == y && y == z
cuatroIguales x y z u = x == y && tresIguales y z u

main = do
print (cuatroIguales 5 5 5 5)
print (cuatroIguales 5 5 4 5)
```

```
❯ ghc -o main main.hs
[1 of 1] Compiling Main             ( main.h
Linking main ...
❯ ./main
True
False
❯
```

Ejercicio R

Ejercicio S

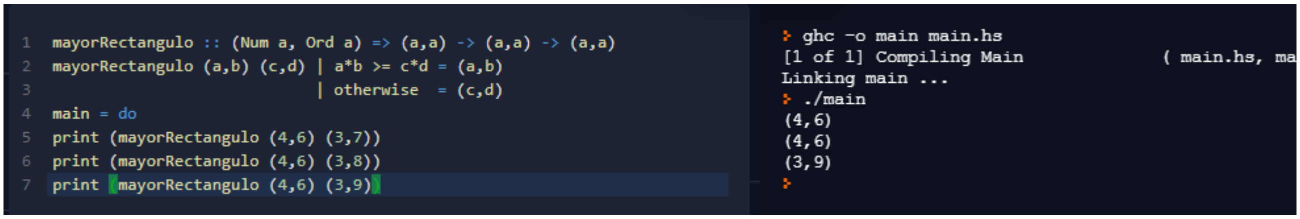
Ejercicio T



```haskell
divisionSegura _ 0 = 9999
divisionSegura x y = x/y

main = do
print (divisionSegura 7 2)
print (divisionSegura 7 0)
```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, ma
Linking main ...
› ./main
3.5
9999.0
›
```

Ejercicio U



```haskell
mayorRectangulo :: (Num a, Ord a) => (a,a) -> (a,a) -> (a,a)
mayorRectangulo (a,b) (c,d) | a*b >= c*d = (a,b)
                            | otherwise  = (c,d)
main = do
print (mayorRectangulo (4,6) (3,7))
print (mayorRectangulo (4,6) (3,8))
print (mayorRectangulo (4,6) (3,9))
```

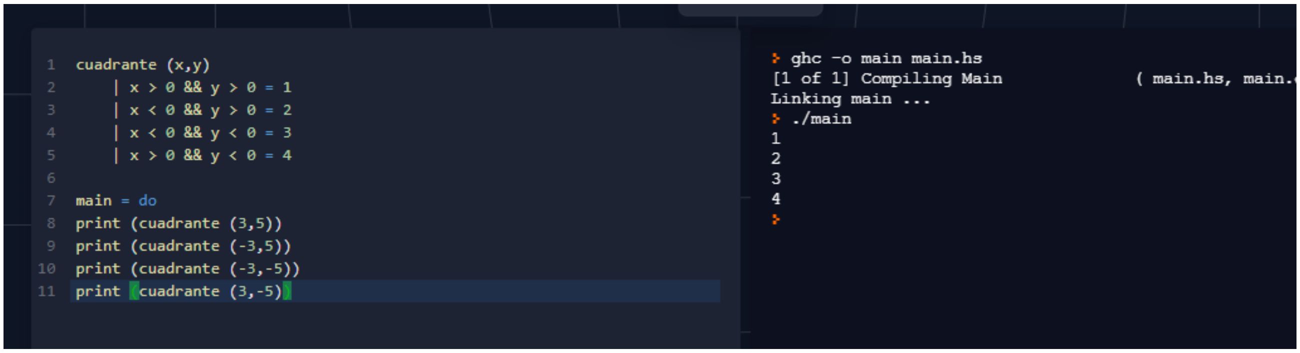
```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, ma
Linking main ...
› ./main
(4,6)
(4,6)
(3,9)
›
```

Ejercicio V

Ejercicio W

Ejercicio X

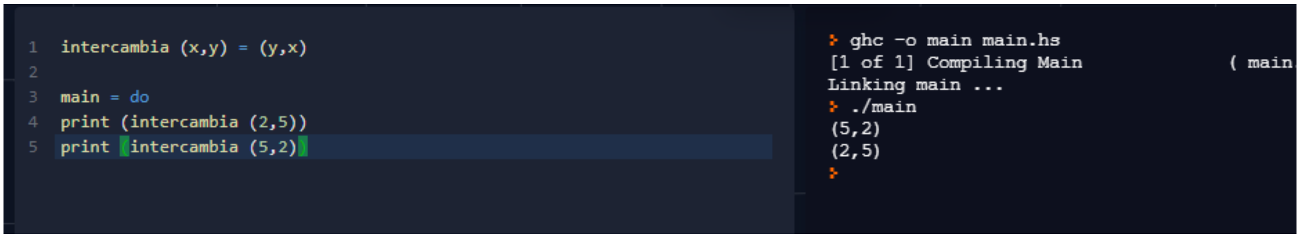
Ejercicio Y



```haskell
cuadrante (x,y)
    | x > 0 && y > 0 = 1
    | x < 0 && y > 0 = 2
    | x < 0 && y < 0 = 3
    | x > 0 && y < 0 = 4

main = do
print (cuadrante (3,5))
print (cuadrante (-3,5))
print (cuadrante (-3,-5))
print (cuadrante (3,-5))
```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main.hs, main.
Linking main ...
› ./main
1
2
3
4
›
```

Ejercicio Z



```haskell
intercambia (x,y) = (y,x)

main = do
print (intercambia (2,5))
print (intercambia (5,2))
```

```
› ghc -o main main.hs
[1 of 1] Compiling Main             ( main
Linking main ...
› ./main
(5,2)
(2,5)
›
```