PRINSIP BAHASA PEMROGRAMAN

Week 6 – Latihan HASKELL

Disusun oleh:

Muhamad Mathar Rizqi

(221524014)

Jurusan Teknik Komputer dan Informatika

Program Studi D-4 Teknik Informatika

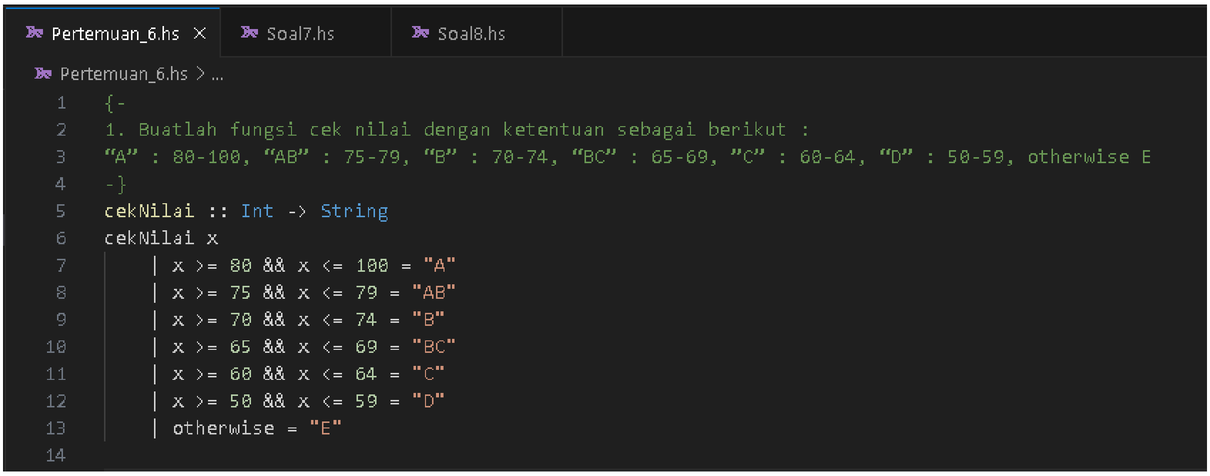
POLITEKNIK NEGERI BANDUNG

Jl. Gegerkalong Hilir, Ciwaruga, Kec. Parongpong, Kabupaten Bandung Barat, Jawa Barat
40559

DAFTAR ISI

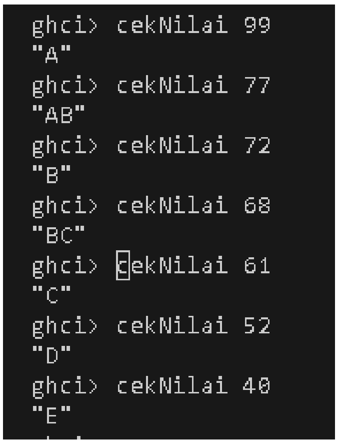
Soal 1

Code:



```haskell
{-
1. Buatlah fungsi cek nilai dengan ketentuan sebagai berikut :
“A” : 80-100, “AB” : 75-79, “B” : 70-74, “BC” : 65-69, ”C” : 60-64, “D” : 50-59, otherwise E
-}
cekNilai :: Int -> String
cekNilai x
    | x >= 80 && x <= 100 = "A"
    | x >= 75 && x <= 79 = "AB"
    | x >= 70 && x <= 74 = "B"
    | x >= 65 && x <= 69 = "BC"
    | x >= 60 && x <= 64 = "C"
    | x >= 50 && x <= 59 = "D"
    | otherwise = "E"
```

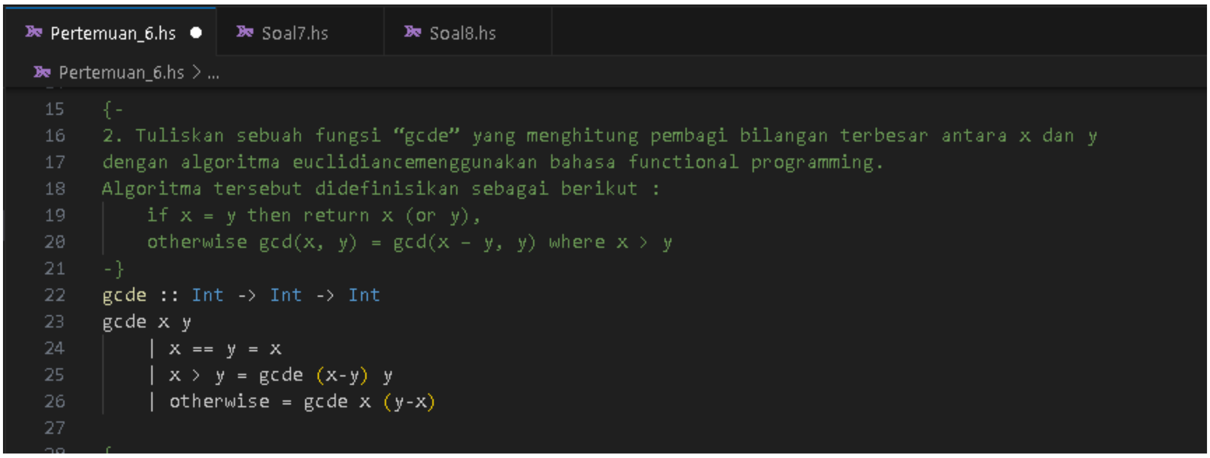
Output:



```
ghci> cekNilai 99
"A"
ghci> cekNilai 77
"AB"
ghci> cekNilai 72
"B"
ghci> cekNilai 68
"BC"
ghci> cekNilai 61
"C"
ghci> cekNilai 52
"D"
ghci> cekNilai 40
"E"
```

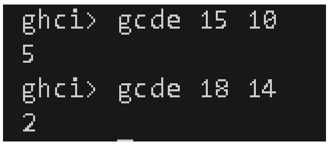
Soal 2

Code:



```haskell
{-
2. Tuliskan sebuah fungsi “gcde” yang menghitung pembagi bilangan terbesar antara x dan y
dengan algoritma euclidiancemenggunakan bahasa functional programming.
Algoritma tersebut didefinisikan sebagai berikut :
    if x = y then return x (or y),
    otherwise gcd(x, y) = gcd(x – y, y) where x > y
-}
gcde :: Int -> Int -> Int
gcde x y
    | x == y = x
    | x > y = gcde (x-y) y
    | otherwise = gcde x (y-x)
```

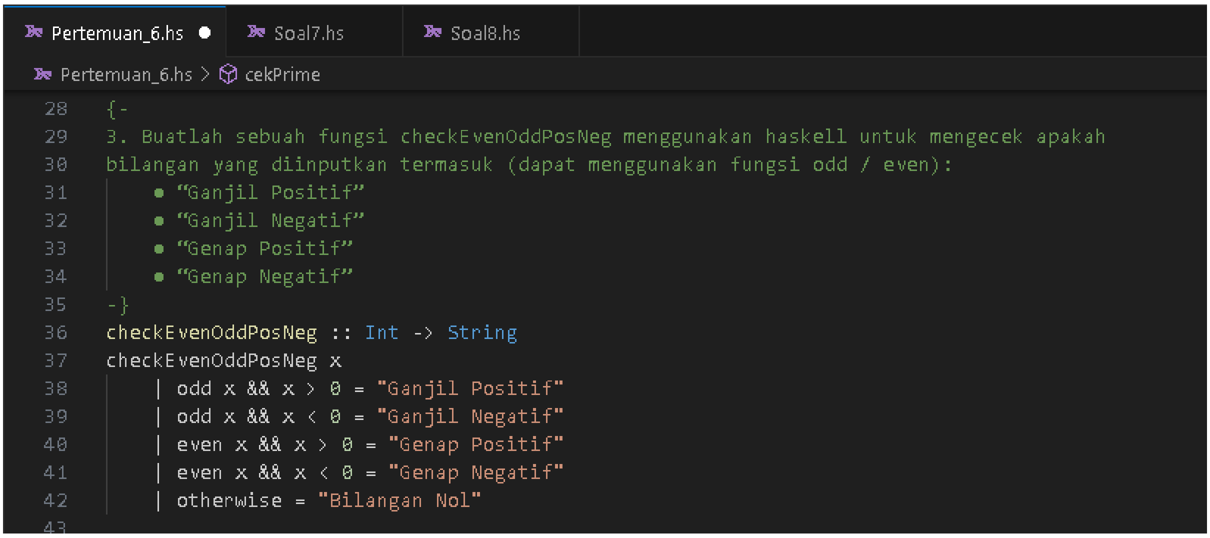
Output:



```
ghci> gcde 15 10
5
ghci> gcde 18 14
2
```

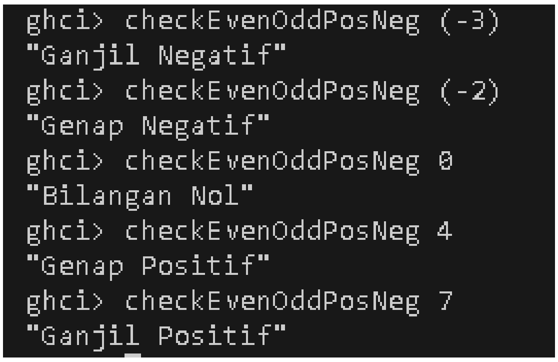
Soal 3

Code:



```haskell
{-
3. Buatlah sebuah fungsi checkEvenOddPosNeg menggunakan haskell untuk mengecek apakah
bilangan yang diinputkan termasuk (dapat menggunakan fungsi odd / even):
    • “Ganjil Positif”
    • “Ganjil Negatif”
    • “Genap Positif”
    • “Genap Negatif”
-}
checkEvenOddPosNeg :: Int -> String
checkEvenOddPosNeg x
    | odd x && x > 0 = "Ganjil Positif"
    | odd x && x < 0 = "Ganjil Negatif"
    | even x && x > 0 = "Genap Positif"
    | even x && x < 0 = "Genap Negatif"
    | otherwise = "Bilangan Nol"
```

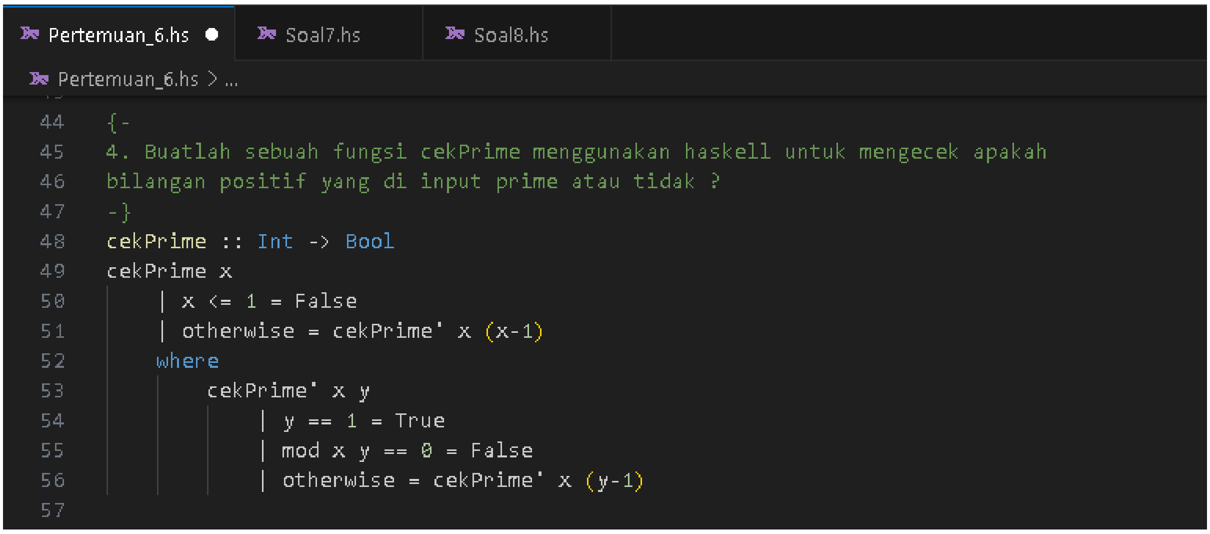
Output:



```
ghci> checkEvenOddPosNeg (-3)
"Ganjil Negatif"
ghci> checkEvenOddPosNeg (-2)
"Genap Negatif"
ghci> checkEvenOddPosNeg 0
"Bilangan Nol"
ghci> checkEvenOddPosNeg 4
"Genap Positif"
ghci> checkEvenOddPosNeg 7
"Ganjil Positif"
```

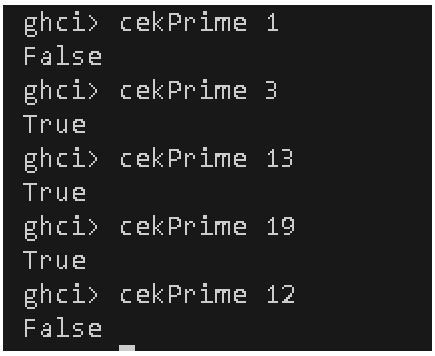
Soal 4

Code:



```haskell
{-
4. Buatlah sebuah fungsi cekPrime menggunakan haskell untuk mengecek apakah
bilangan positif yang di input prime atau tidak ?
-}
cekPrime :: Int -> Bool
cekPrime x
    | x <= 1 = False
    | otherwise = cekPrime' x (x-1)
    where
        cekPrime' x y
            | y == 1 = True
            | mod x y == 0 = False
            | otherwise = cekPrime' x (y-1)
```

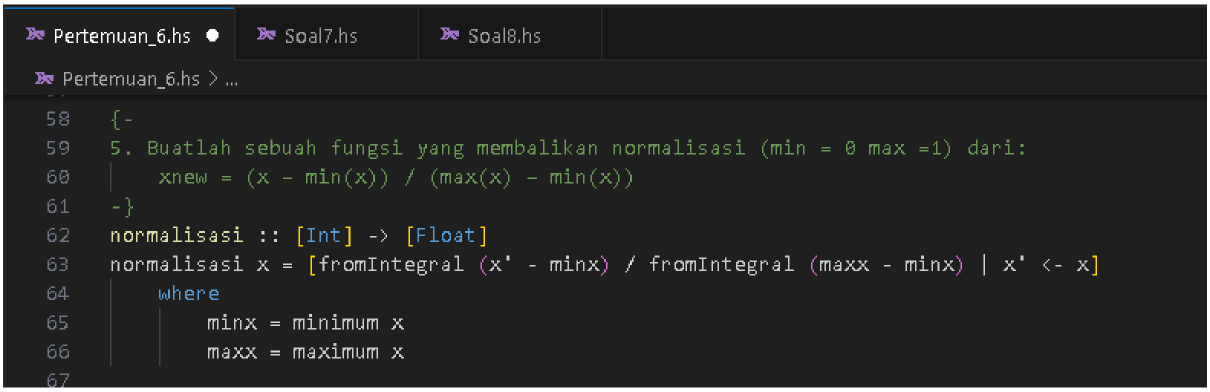
Output:



```
ghci> cekPrime 1
False
ghci> cekPrime 3
True
ghci> cekPrime 13
True
ghci> cekPrime 19
True
ghci> cekPrime 12
False
```

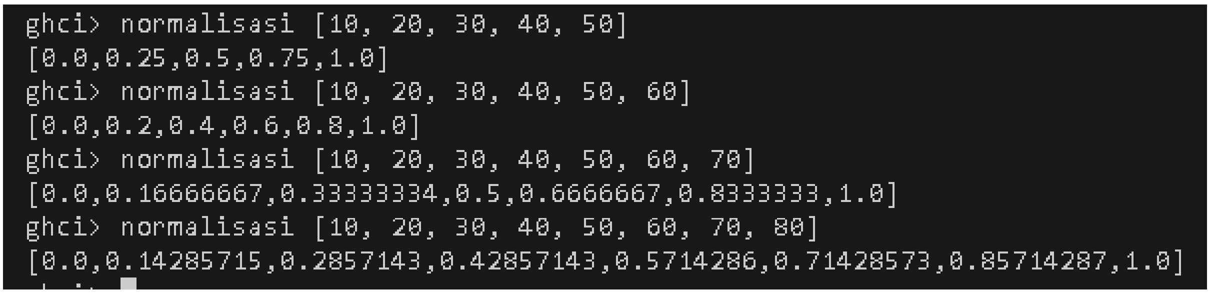
Soal 5

Code:



```haskell
{-
5. Buatlah sebuah fungsi yang membalikan normalisasi (min = 0 max =1) dari:
    xnew = (x – min(x)) / (max(x) – min(x))
-}
normalisasi :: [Int] -> [Float]
normalisasi x = [fromIntegral (x' - minx) / fromIntegral (maxx - minx) | x' <- x]
    where
        minx = minimum x
        maxx = maximum x
```

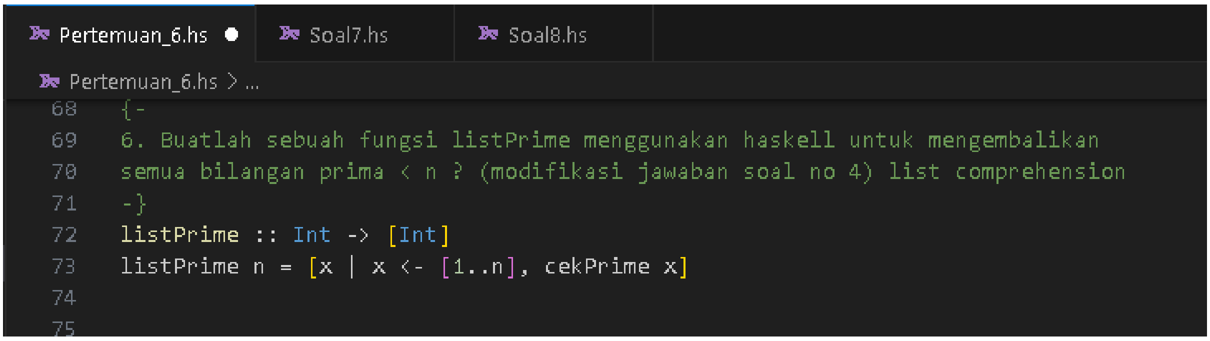
Output:



```
ghci> normalisasi [10, 20, 30, 40, 50]
[0.0,0.25,0.5,0.75,1.0]
ghci> normalisasi [10, 20, 30, 40, 50, 60]
[0.0,0.2,0.4,0.6,0.8,1.0]
ghci> normalisasi [10, 20, 30, 40, 50, 60, 70]
[0.0,0.16666667,0.33333334,0.5,0.6666667,0.8333333,1.0]
ghci> normalisasi [10, 20, 30, 40, 50, 60, 70, 80]
[0.0,0.14285715,0.2857143,0.42857143,0.5714286,0.71428573,0.85714287,1.0]
```

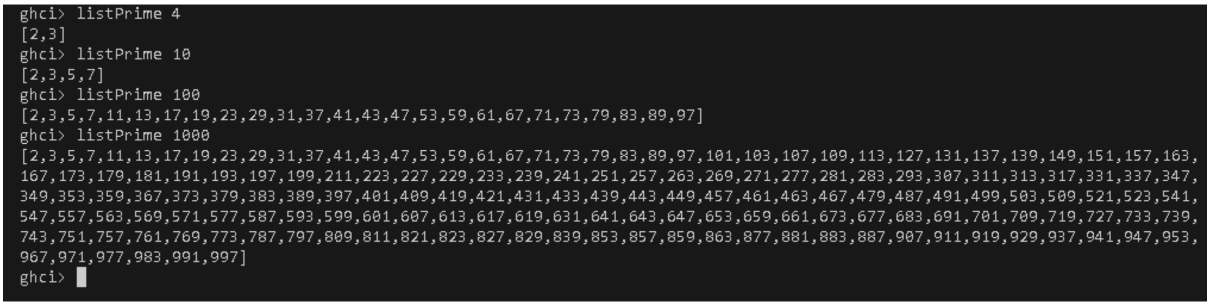
Soal 6

Code:



```
{-
6. Buatlah sebuah fungsi listPrime menggunakan haskell untuk mengembalikan
semua bilangan prima < n ? (modifikasi jawaban soal no 4) list comprehension
-}
listPrime :: Int -> [Int]
listPrime n = [x | x <- [1..n], cekPrime x]
```

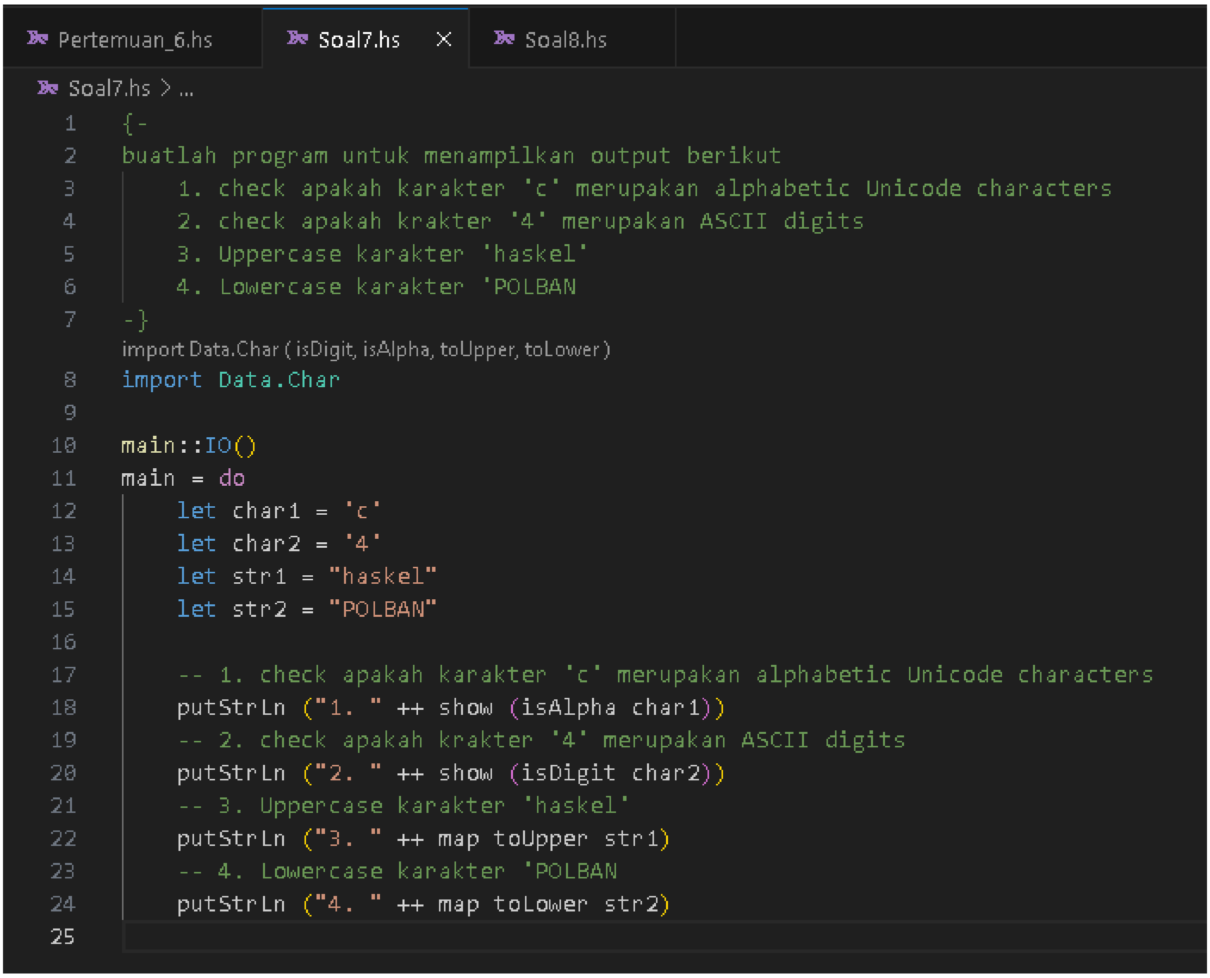
Output:



```
ghci> listPrime 4
[2,3]
ghci> listPrime 10
[2,3,5,7]
ghci> listPrime 100
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97]
ghci> listPrime 1000
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101,103,107,109,113,127,131,137,139,149,151,157,163,
167,173,179,181,191,193,197,199,211,223,227,229,233,239,241,251,257,263,269,271,277,281,283,293,307,311,313,317,331,337,347,
349,353,359,367,373,379,383,389,397,401,409,419,421,431,433,439,443,449,457,461,463,467,479,487,491,499,503,509,521,523,541,
547,557,563,569,571,577,587,593,599,601,607,613,617,619,631,641,643,647,653,659,661,673,677,683,691,701,709,719,727,733,739,
743,751,757,761,769,773,787,797,809,811,821,823,827,829,839,853,857,859,863,877,881,883,887,907,911,919,929,937,941,947,953,
967,971,977,983,991,997]
ghci> 
```

Soal 7

Code:



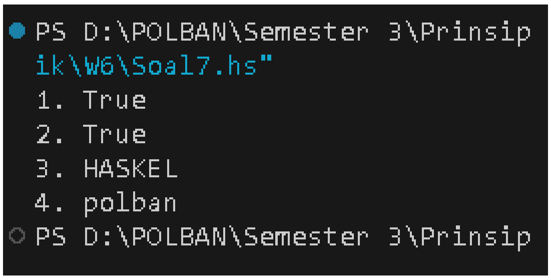
```haskell
{-
buatlah program untuk menampilkan output berikut
    1. check apakah karakter 'c' merupakan alphabetic Unicode characters
    2. check apakah krakter '4' merupakan ASCII digits
    3. Uppercase karakter 'haskel'
    4. Lowercase karakter 'POLBAN
-}
import Data.Char

main::IO()
main = do
    let char1 = 'c'
    let char2 = '4'
    let str1 = "haskel"
    let str2 = "POLBAN"

    -- 1. check apakah karakter 'c' merupakan alphabetic Unicode characters
    putStrLn ("1. " ++ show (isAlpha char1))
    -- 2. check apakah krakter '4' merupakan ASCII digits
    putStrLn ("2. " ++ show (isDigit char2))
    -- 3. Uppercase karakter 'haskel'
    putStrLn ("3. " ++ map toUpper str1)
    -- 4. Lowercase karakter 'POLBAN
    putStrLn ("4. " ++ map toLower str2)

```

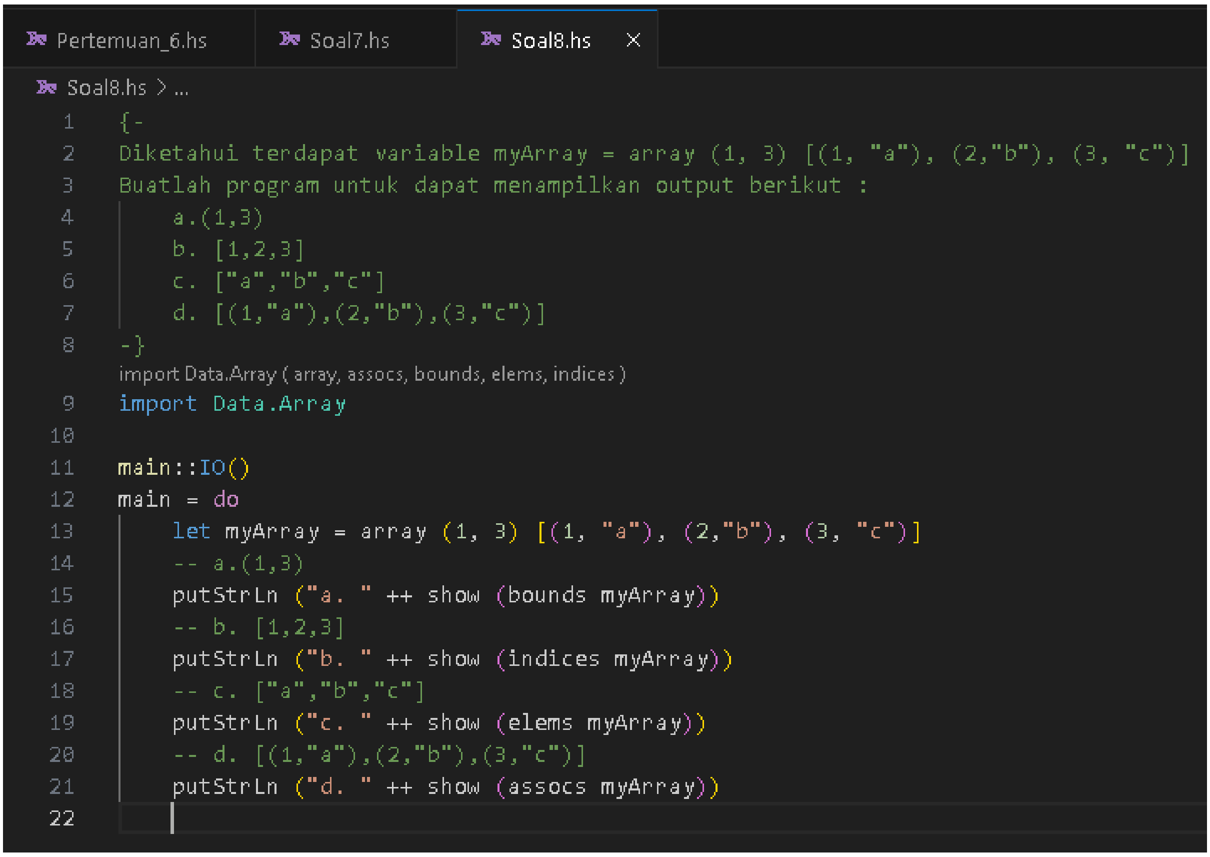
Output:



```
● PS D:\POLBAN\Semester 3\Prinsip
  ik\W6\Soal7.hs"
  1. True
  2. True
  3. HASKEL
  4. polban
○ PS D:\POLBAN\Semester 3\Prinsip
```

Soal 8

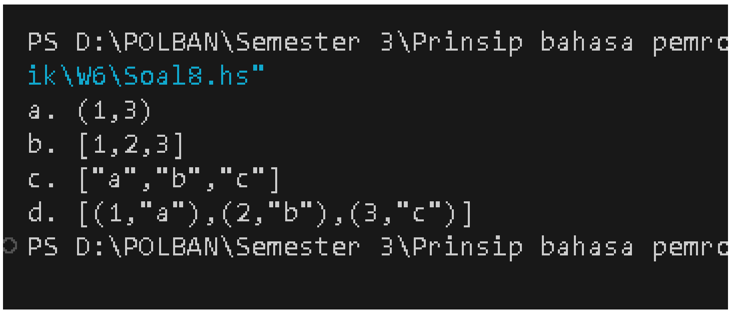
Code:



```haskell
{-
Diketahui terdapat variable myArray = array (1, 3) [(1, "a"), (2,"b"), (3, "c")]
Buatlah program untuk dapat menampilkan output berikut :
    a.(1,3)
    b. [1,2,3]
    c. ["a","b","c"]
    d. [(1,"a"),(2,"b"),(3,"c")]
-}
import Data.Array ( array, assocs, bounds, elems, indices )
import Data.Array

main::IO()
main = do
    let myArray = array (1, 3) [(1, "a"), (2,"b"), (3, "c")]
    -- a.(1,3)
    putStrLn ("a. " ++ show (bounds myArray))
    -- b. [1,2,3]
    putStrLn ("b. " ++ show (indices myArray))
    -- c. ["a","b","c"]
    putStrLn ("c. " ++ show (elems myArray))
    -- d. [(1,"a"),(2,"b"),(3,"c")]
    putStrLn ("d. " ++ show (assocs myArray))
    
```

Output:



```
PS D:\POLBAN\Semester 3\Prinsip bahasa pemro
ik\W6\Soal8.hs"
a. (1,3)
b. [1,2,3]
c. ["a","b","c"]
d. [(1,"a"),(2,"b"),(3,"c")]
PS D:\POLBAN\Semester 3\Prinsip bahasa pemro
```

GitHub

https://github.com/Matharrr/Week6_PBP