LAPORAN PERTEMUAN 6
PRAKTIKUM HASKELL

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah Prinsip Bahasa Pemrograman

Disusun oleh:

Suci Awalia Gardara 211524027

PROGRAM STUDI D4 TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2022

DAFTAR ISI

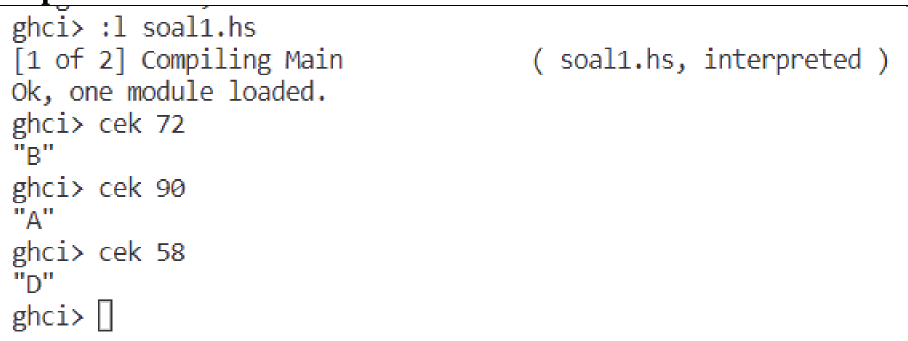
Soal 1

Jawab :

Source Code :

```
cek score =
    if score >= 80 && score <= 100 then "A"
    else if score >= 75 && score <= 79 then "AB"
    else if score >= 70 && score <= 74 then "B"
    else if score >= 65 && score <= 69 then "BC"
    else if score >= 60 && score <= 64 then "C"
    else if score >= 50 && score <= 59 then "D"
    else "E"
```

Output :



```
ghci> :l soal1.hs
[1 of 2] Compiling Main             ( soal1.hs, interpreted )
Ok, one module loaded.
ghci> cek 72
"B"
ghci> cek 90
"A"
ghci> cek 58
"D"
ghci> 
```

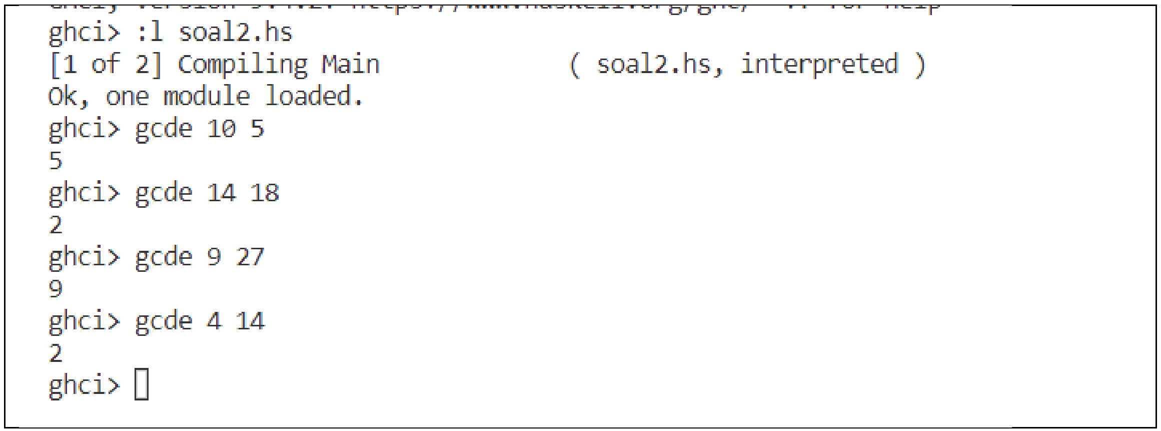
Soal 2

Jawab :

Source Code :

```
gcde :: Int -> Int -> Int
gcde x y =
    if  (x == y)  then  x
    else if (x > y)  then gcde (x - y) y
    else  gcde y x
```

Output :



```
ghci> :l soal2.hs
[1 of 2] Compiling Main             ( soal2.hs, interpreted )
Ok, one module loaded.
ghci> gcde 10 5
5
ghci> gcde 14 18
2
ghci> gcde 9 27
9
ghci> gcde 4 14
2
ghci> 
```

Soal 3

Jawab :

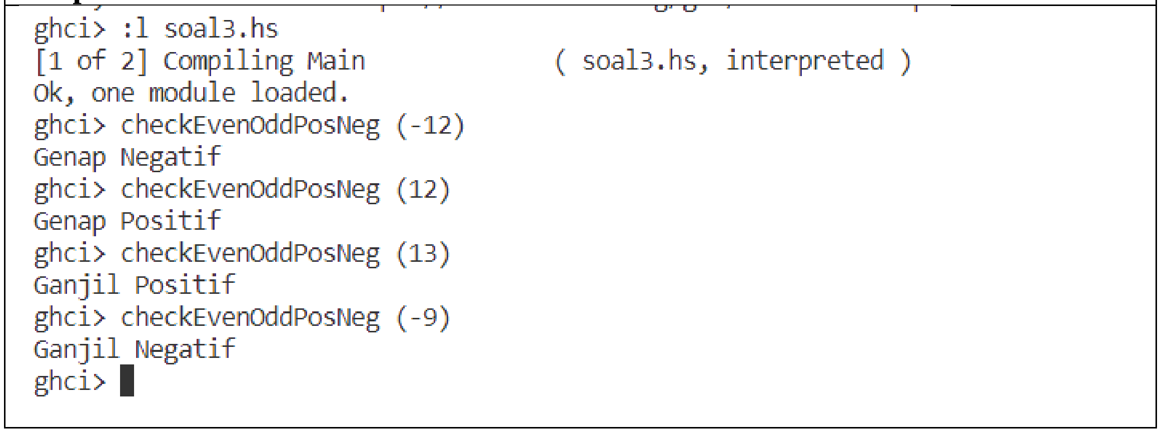
Source Code :

```
checkEvenOdd :: Int -> String
checkEvenOdd x  | even x = "Genap "
                | odd x  = "Ganjil "

checkPosNeg :: Int -> String
checkPosNeg x   | x > 0  = "Positif"
                | x < 0  = "Negatif"
                | x == 0 = "Nol"

checkEvenOddPosNeg :: Int -> IO()
checkEvenOddPosNeg x = putStrLn ((checkEvenOdd x) ++ (checkPosNeg x))
```

Output :



```
ghci> :l soal3.hs
[1 of 2] Compiling Main             ( soal3.hs, interpreted )
Ok, one module loaded.
ghci> checkEvenOddPosNeg (-12)
Genap Negatif
ghci> checkEvenOddPosNeg (12)
Genap Positif
ghci> checkEvenOddPosNeg (13)
Ganjil Positif
ghci> checkEvenOddPosNeg (-9)
Ganjil Negatif
ghci> 
```

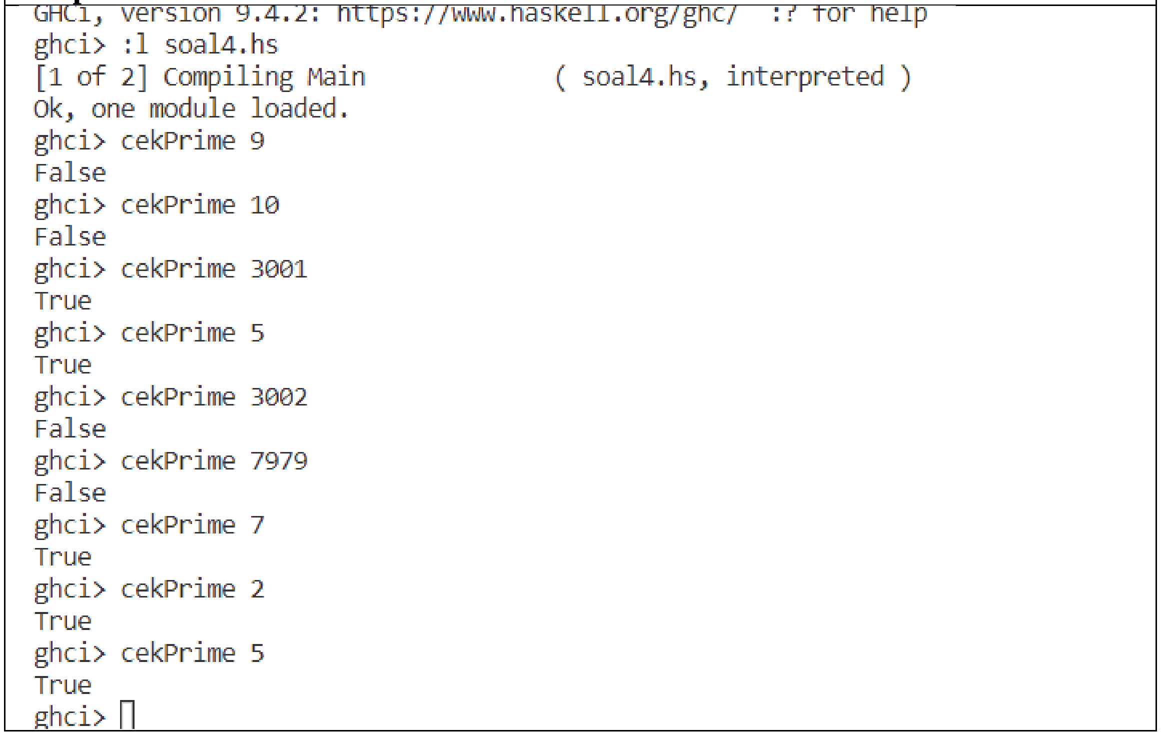
Soal 4

Jawab :

Source Code :

```haskell
cekPrime :: Int -> Bool
cekPrime 1 = False
cekPrime 2 = True
cekPrime n | (length [x | x <- [2 .. n-1], mod n x == 0]) > 0 = False
             | otherwise = True
```

Output :



```
GHCi, version 9.4.2: https://www.haskell.org/ghc/  :? for help
ghci> :l soal4.hs
[1 of 2] Compiling Main             ( soal4.hs, interpreted )
Ok, one module loaded.
ghci> cekPrime 9
False
ghci> cekPrime 10
False
ghci> cekPrime 3001
True
ghci> cekPrime 5
True
ghci> cekPrime 3002
False
ghci> cekPrime 7979
False
ghci> cekPrime 7
True
ghci> cekPrime 2
True
ghci> cekPrime 5
True
ghci> 
```

Soal 5

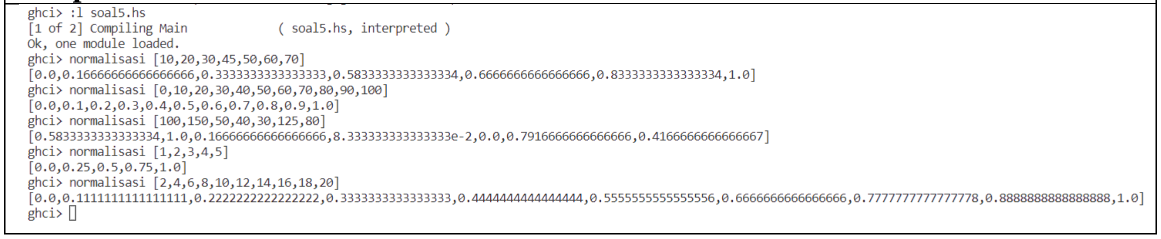
Jawab :

Source Code :

```haskell
normalis x xMax xMin = [(a - xMin) / (xMax - xMin) | a <- x]

normalisasi (x) = normalis x (maximum x) (minimum x)
```

Output :



```
ghci> :l soal5.hs
[1 of 2] Compiling Main             ( soal5.hs, interpreted )
Ok, one module loaded.
ghci> normalisasi [10,20,30,45,50,60,70]
[0.0,0.16666666666666666,0.3333333333333333,0.5833333333333334,0.6666666666666666,0.8333333333333334,1.0]
ghci> normalisasi [0,10,20,30,40,50,60,70,80,90,100]
[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]
ghci> normalisasi [100,150,50,40,30,125,80]
[0.5833333333333334,1.0,0.16666666666666666,8.333333333333333e-2,0.0,0.7916666666666666,0.4166666666666667]
ghci> normalisasi [1,2,3,4,5]
[0.0,0.25,0.5,0.75,1.0]
ghci> normalisasi [2,4,6,8,10,12,14,16,18,20]
[0.0,0.1111111111111111,0.2222222222222222,0.3333333333333333,0.4444444444444444,0.5555555555555556,0.6666666666666666,0.7777777777777778,0.8888888888888888,1.0]
ghci> 
```

Soal 6

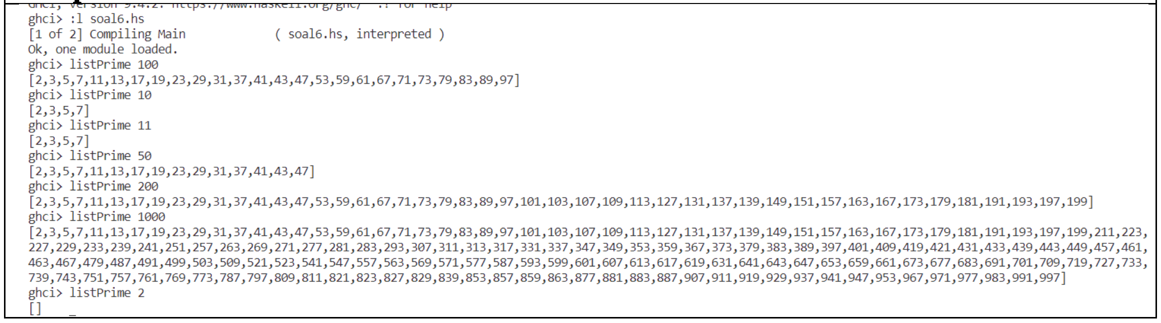
Jawab :

Source Code :

```haskell
cekPrime :: Int -> Bool
cekPrime 1 = False
cekPrime 2 = True
cekPrime n | (length [x | x <- [2 .. n-1], mod n x == 0]) > 0 = False
             | otherwise = True

listPrime n = [x | x <- [1..n-1], (cekPrime x)]
```

Output :



```
ghci> :l soal6.hs
[1 of 2] Compiling Main             ( soal6.hs, interpreted )
Ok, one module loaded.
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 10
[2,3,5,7]
ghci> listPrime 11
[2,3,5,7]
ghci> listPrime 50
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
ghci> listPrime 200
[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]
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> listPrime 2
[]
```

Soal 7

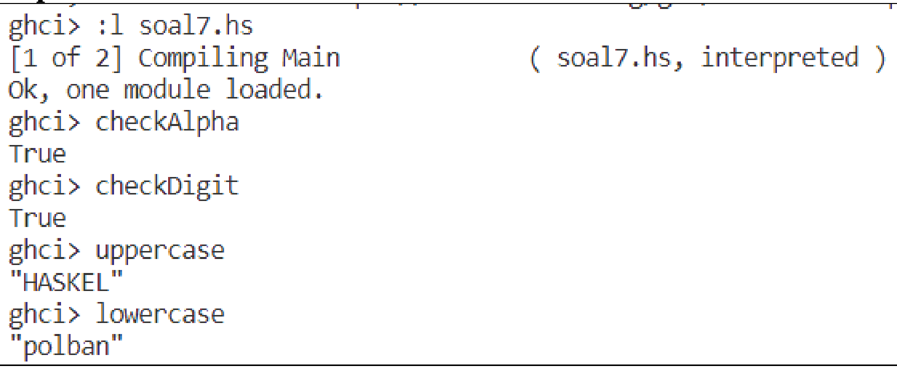
Source Code :

```haskell
import Data.Char

checkAlpha = isAlpha 'c'
checkDigit = isDigit '4'

uppercase = [toUpper c | c <- "haskel"]
lowercase = [toLower c | c <- "POLBAN"]
```

Output :



```
ghci> :l soal7.hs
[1 of 2] Compiling Main             ( soal7.hs, interpreted )
Ok, one module loaded.
ghci> checkAlpha
True
ghci> checkDigit
True
ghci> uppercase
"HASKEL"
ghci> lowercase
"polban"
```

Soal 8

Jawab :

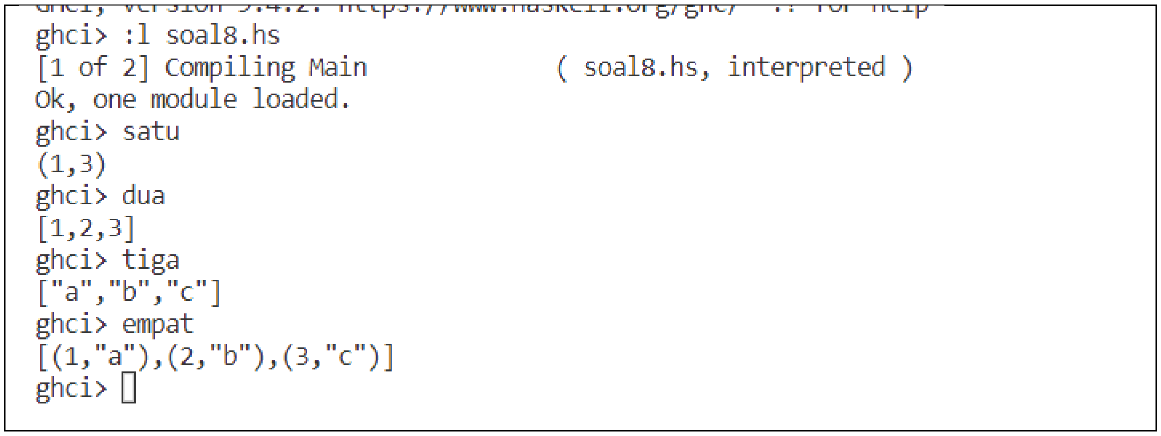
Source Code :

```haskell
import Data.Array

myArray = array (1, 3) [(1, "a"), (2, "b"), (3, "c")]

satu = bounds myArray
dua = indices myArray
tiga = elems myArray
empat = assocs myArray
```

Output :



```
GHCi, version 9.4.2: https://www.haskell.org/ghc/  :? for help
ghci> :l soal8.hs
[1 of 2] Compiling Main             ( soal8.hs, interpreted )
Ok, one module loaded.
ghci> satu
(1,3)
ghci> dua
[1,2,3]
ghci> tiga
["a","b","c"]
ghci> empat
[(1,"a"),(2,"b"),(3,"c")]
ghci> 
```