**BST – Binary Search Tree**

import Data.List

data Arvore a = Nulo | No (Arvore a) a (Arvore a)

deriving (Show)

criar\_arvore :: (Ord a) => [a] -> Arvore a

criar\_arvore [] = Nulo

criar\_arvore (x:xs) = criar\_arvore\_aux (No Nulo x Nulo) xs

where

criar\_arvore\_aux arvore [] = arvore

criar\_arvore\_aux arvore (x:xs) = criar\_arvore\_aux (inserir arvore x) xs

inserir :: (Ord a) => Arvore a -> a -> Arvore a

inserir Nulo x = No Nulo x Nulo

inserir (No arv1 v arv2) x

| (v == x) = No arv1 v arv2

| (v < x) = No arv1 v(inserir arv2 x)

| (v > x) = No (inserir arv1 x) v arv2

ordem :: [Int] -> [Int]

ordem vetor = sort vetor

preordem :: [Int] -> [Int]

ordem vetor = vetor

main = do

putStrLn "Digite um vetor"

vetor <- getLine

(print.criar\_arvore) ((read vetor)::[Int])

putStrLn "Percurso em ordem:"

(print.ordem) ((read vetor)::[Int])

putStrLn "Percurso em pré-ordem:"

(print.preordem) ((read vetor)::[Int])