```haskell
{-
   Michael E. Sparks, 4 May 2021

   SSR_data_Euclid.hs - Haskell statements to prepare a matrix of
                        Euclidean distances from all pairwise
                        combinations of simple sequence repeat
                        records.
-}

import Data.List
import System.IO

-- for simplicity, let’s just hard code the input data
a1=[204,151,109,117,134]
a2=[203,154,111,117,135]
a3=[204,148,109,117,135]
a4=[203,145,113,117,135]
a5=[203,149,112,117,135]
b1=[210,158,94,110,135]
b2=[213,160,96,110,135]
b3=[211,159,96,110,135]
b4=[215,161,96,110,135]
b5=[215,160,90,113,135]
c1=[180,188,158,112,135]
c2=[184,185,158,113,135]
c3=[181,186,158,112,135]
c4=[180,188,156,118,135]
c5=[180,188,158,119,135]

ssrDataNames=[
  "a1","a2","a3","a4","a5",
  "b1","b2","b3","b4","b5",
  "c1","c2","c3","c4","c5"
  ]

ssrData=[
  a1,a2,a3,a4,a5,
  b1,b2,b3,b4,b5,
  c1,c2,c3,c4,c5
  ]

cellLabels = [(i,j) | i <- ssrDataNames, j <- ssrDataNames]

cellData = map (\ (x,y) -> sqrt $ sum [(a-b)^2 | (a,b) <- zip x y])
               [(i,j) | i <- ssrData, j <- ssrData]

matrix = zip cellLabels cellData

stringifyRow i = n ++ "\t" ++ dists ++ "\n"
  where n = ssrDataNames !! i
        dists = intercalate "\t" $
          map show $ map snd [x | x <- matrix, (fst . fst) x==n]

main = do
  -- print to stdout (i.e., terminal screen)
  putStrLn $ "\t" ++ (intercalate "\t" ssrDataNames)
  mapM_ (putStr . stringifyRow) [0..length ssrDataNames - 1]

  -- now for the filestream I/O; for simplicity,
  -- we'll just hard code the output filename, too
  let outfile="matrix.tab-delim.txt"
  writeFile outfile $ "\t" ++ (intercalate "\t" ssrDataNames) ++ "\n"
  mapM_ (\ i -> appendFile outfile $ stringifyRow i)
        [0..length ssrDataNames - 1]
```