```haskell

-- 2.a --
createSquaredList :: (Integral a) => [a] -> [a]
createSquaredList [] = []
createSquaredList theList = [x^2 | x <- theList]

-- 2.b --
createSquareRootList :: (Integral a) => [a] -> [a]
createSquareRootList theList = [truncate (sqrt (fromIntegral n)) | n <- theList, isSquare n]
isSquare :: (Integral a) => a -> Bool
isSquare num = (floor (sqrt (fromIntegral num))) ^ 2 == num

-- 2.c --
getVowelFreeString :: String -> String
getVowelFreeString theString = [c | c <- theString, not $ isVowel c]
isVowel :: Char -> Bool
isVowel c = c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u'
isConsonant :: Char -> Bool
isConsonant c = not $ isVowel c

-- 2.d --
generateTwoSixSidedDiceThatValueSeven :: [[Int]]
generateTwoSixSidedDiceThatValueSeven = [[dieOne, dieTwo]
      | dieOne <- [1..6],
        dieTwo <- [1..6],
        dieOne + dieTwo == 7]

-- 2.e --
makingThreeSimpleSentences :: [String] -> [String] -> [(String, String, String)]
makingThreeSimpleSentences namesList verbsList = take 3 [(subject, verb, object)
      | subject <- namesList,
        verb <- verbsList,
        object <- namesList,
        subject /= object]

```