Exercises
Topic 9 - Interactive Programming

Exercise 1:

Write an I/O program which will read a line of input and test whether the input is a palindrome. The program should 'prompt' the user for its input and also output an appropriate message.

Solution 1:

```
interactivePalCheck :: IO ()

interactivePalCheck
  = do putStr "Input a string for palindrome check: "
       st <- getLine
       if st == reverse st
          then putStr "Palindrome.\n"
          else putStr "Not a palindrome.\n"
```

Exercise 2:

Write an I/O program which will read two integers, each on a separate line and output their sum. The program should prompt for input and explain its output.

Solution 2:

```
interactiveIntSum :: IO ()

interactiveIntSum
  = do putStr "Input an integer (followed by Return): "
       st1 <- getLine
       let int1 = (read st1) :: Int
       putStr "Input another integer (followed by Return): "
       st2 <- getLine
       let int2 = read st2 :: Int
       putStrLn ("The sum of these integers is "++ show (int1+int2))
```

Exercise 3:

Define a function

```
putNtimes :: Integer -> String -> IO ()
```

so that the effect of

```
 putNtimes n str
```

is to output a string *str*, *n* times, one per line.
Hint: You can use recursion in the definition.

Solution 3:

```
putNtimes :: Integer -> String -> IO ()

putNtimes n st
  = if n<=0
       then return ()
       else do putStrLn st
               putNtimes (n-1) st
```

Alternatively

```
putNTimes :: Integer -> String -> IO ()
putNTimes 0 str  = return ()
putNTimes n str  = do putStr str
                      putNTimes (n-1) str
```

Exercise 4:

Write an I/O program which will first read a positive integer, *n*, and then read n integers and write their sum. The program should prompt for input and explain its output.
Hint: use auxillary functions, e.g.

```
 getInteger :: String -> IO Integer
 sumNInts ::        -- .... which sums n ints
```

Solution 4:

```
-- Instead of solving this as a single function, worth thinking about how you ca
-- decompose the problem: write a function to get an integer, and another
-- to do the summing.

-- Useful auxiliary function, taking the prompt as parameter.
```

```
getInteger :: String -> IO Integer

getInteger prompt
  = do putStr prompt
       st <- getLine
       return (read st :: Integer)

-- Sum N integers: prompt, number to sum and and "sum so far" are the parameters

sumNints :: String -> Integer -> Integer -> IO Integer

sumNints prompt n s
  = if n<=0
       then return s
       else do m <- getInteger prompt
               sumNints prompt (n-1) (s+m)


--- The function itself

getNints :: IO ()

getNints
  = do bound <- getInteger "Input the number of integers to add: "
       sum <- sumNints "Input an integer: " bound 0
       putStrLn ("The sum of these integers is "++ show sum)
```