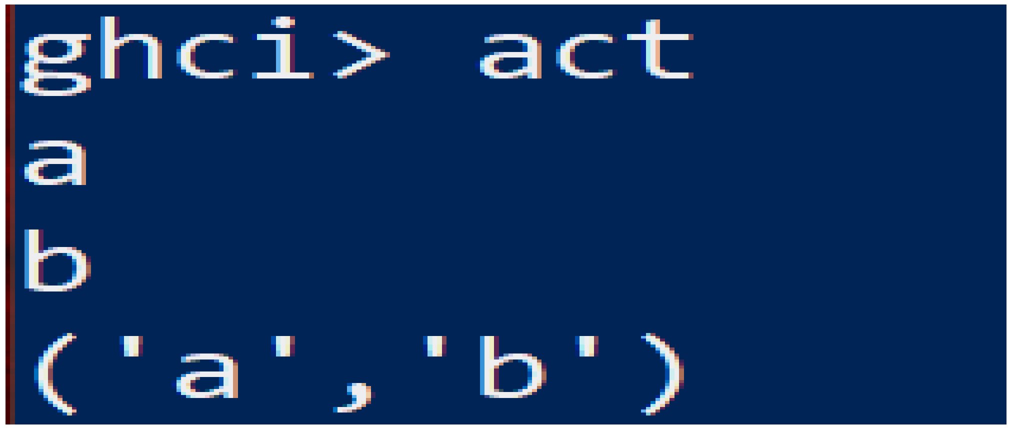
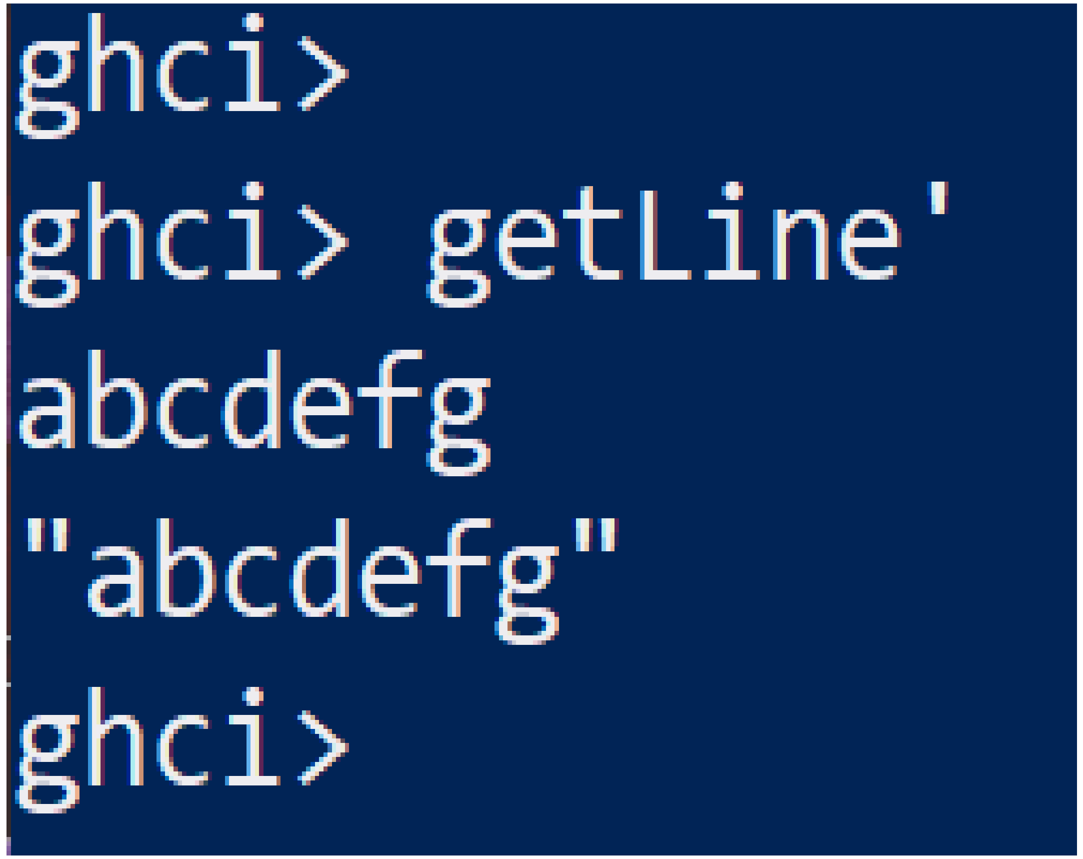
```
act :: IO (Char,Char)
act = do x <- getChar
         getChar
         y <- getChar
         getChar
         return (x,y)
```



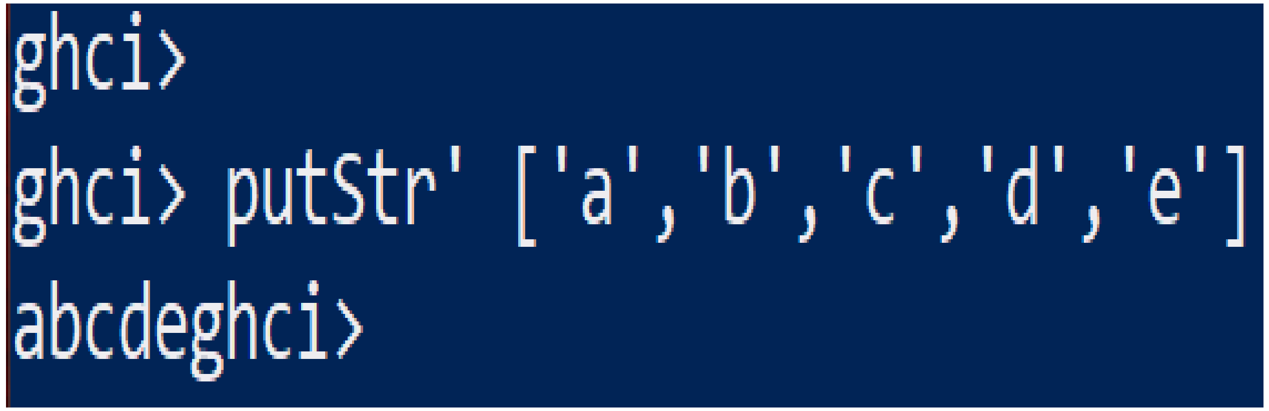
```
ghci> act
a
b
('a','b')
```

```
getLine' :: IO String
getLine' = do x <- getChar
              if x == '\n' then
                 return []
              else
                 do xs <- getLine'
                    return (x:xs)
```

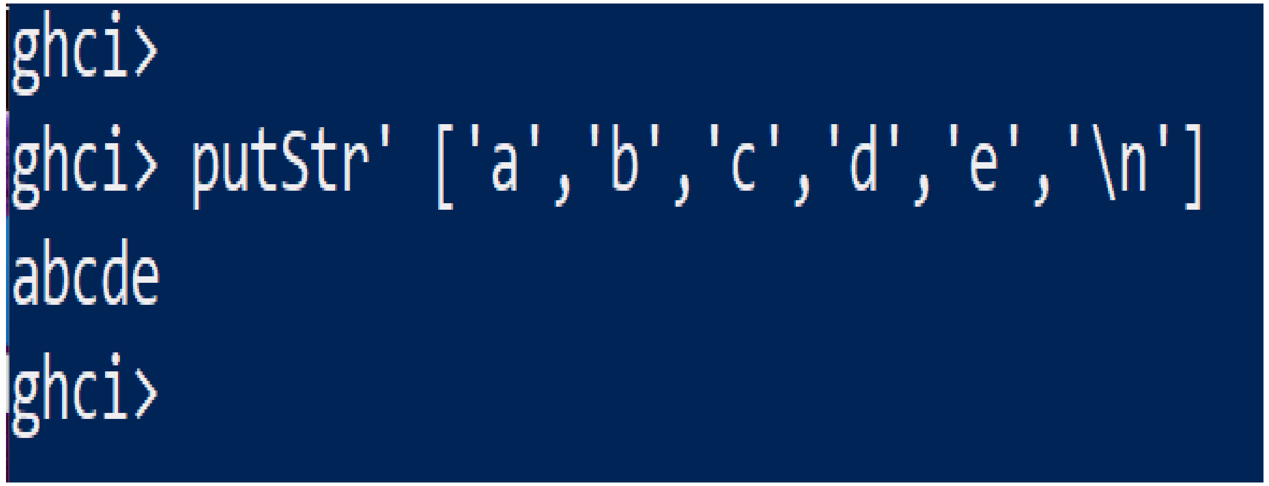


```
ghci>
ghci> getLine'
abcdefg
"abcdefg"
ghci>
```

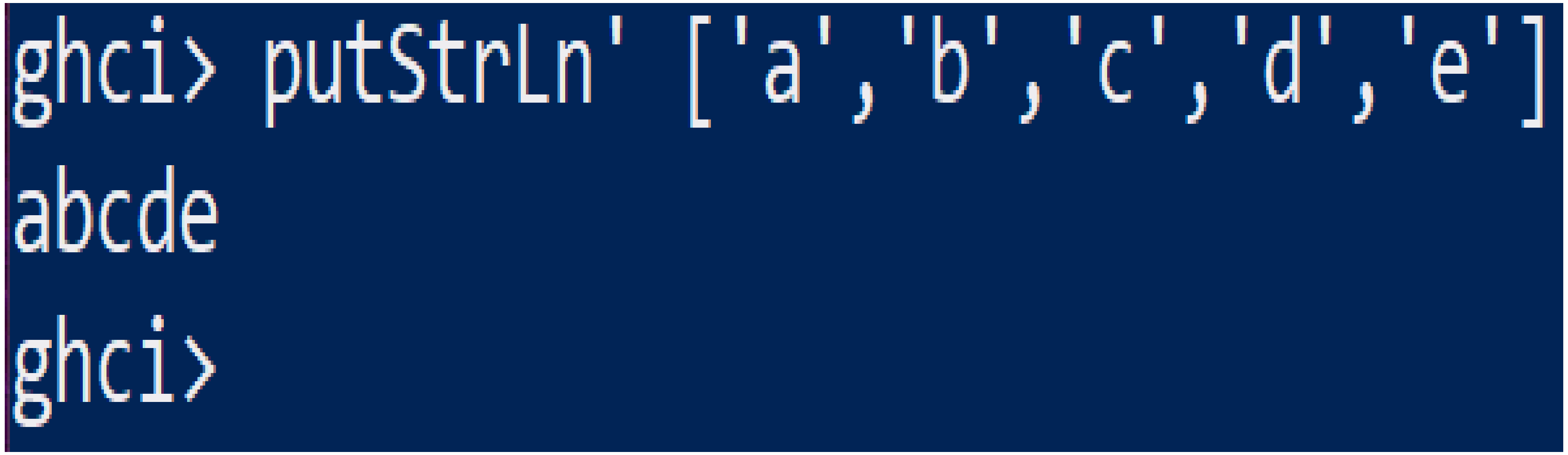
```haskell
putStr' :: String -> IO ()
putStr' [] = return ()
putStr' (x:xs) = do putChar x
                    putStr' xs
```



```
ghci>
ghci> putStr' ['a','b','c','d','e']
abcdeghci>
```

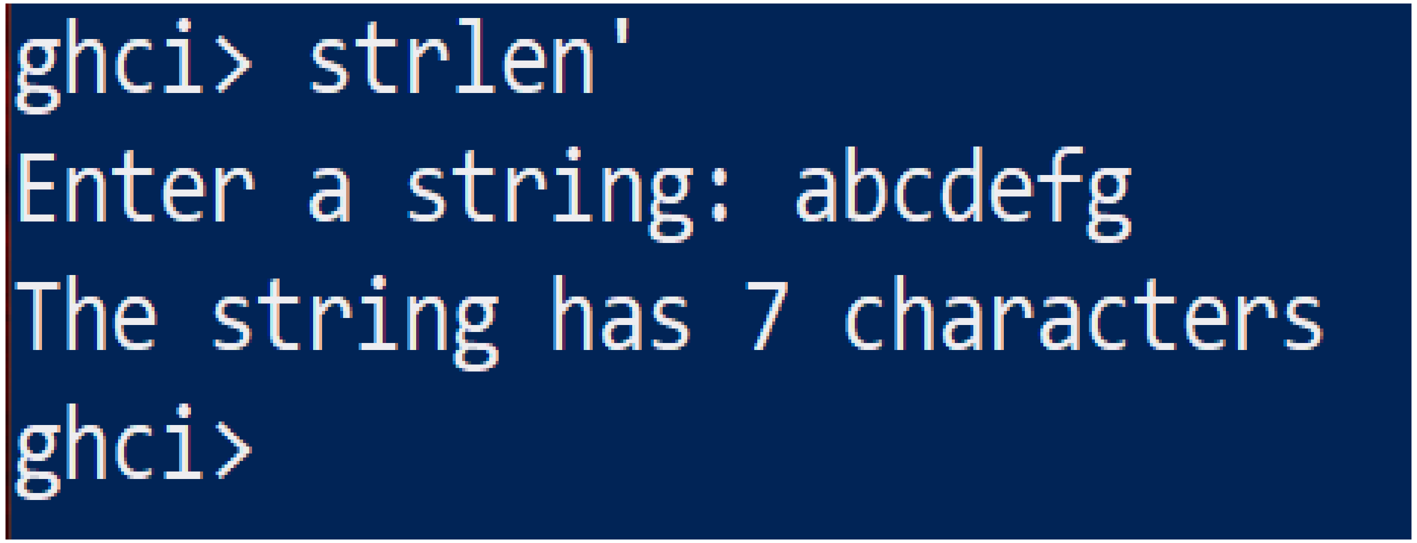


```
ghci>
ghci> putStr' ['a','b','c','d','e','\n']
abcde
ghci>
```



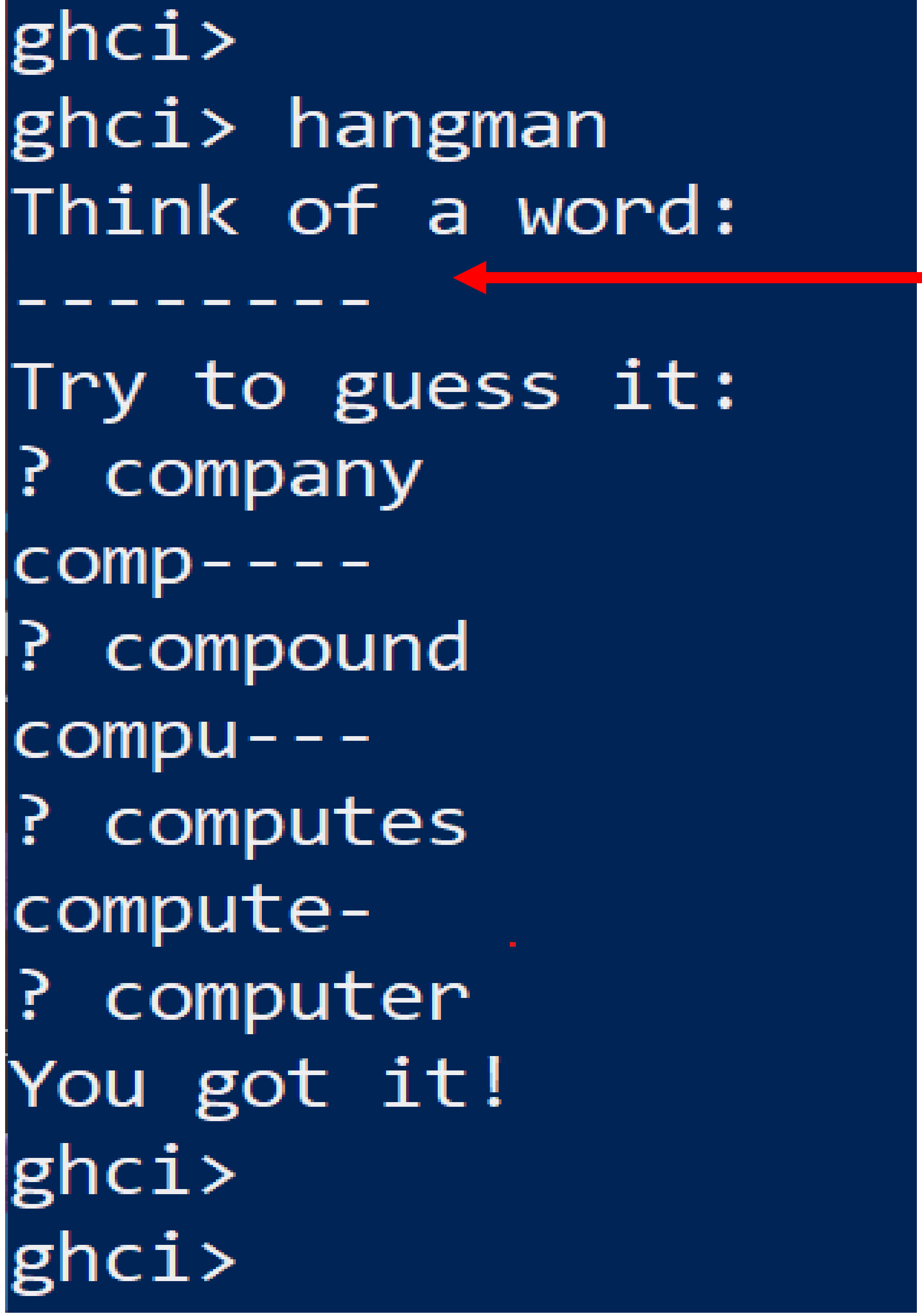
```
ghci> putStrLn' ['a','b','c','d','e']
abcde
ghci>
```

```
putStrLn' :: String -> IO ()
putStrLn' xs = do putStr' xs
                  putChar '\n'
```



```
ghci> strlen'
Enter a string: abcdefg
The string has 7 characters
ghci>
```

```haskell
strlen' :: IO ()
strlen' = do putStr "Enter a string: "
             xs <- getLine'
             putStr "The string has "
             putStr (show (length xs))
             putStrLn " characters"
```



```
ghci>
ghci> hangman
Think of a word:
--------
Try to guess it:
? company
comp----
? compound
compu---
? computes
compute-
? computer
You got it!
ghci>
ghci>
```