



**Master of Computer and Information Science**

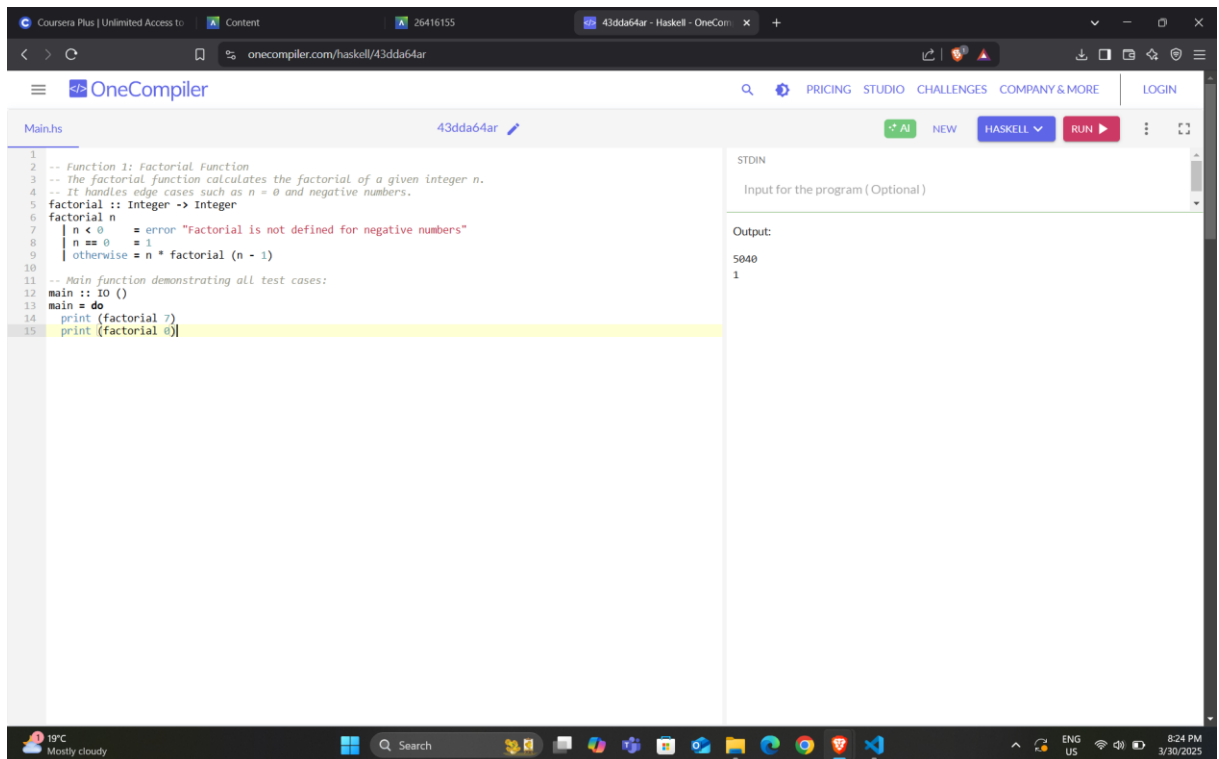
**CIS 524**

**Haskell Programming Assignment**

**Name – Kathan Urmishbhai Patel**

**CSU ID – 2886869**

## 1. Screenshot of Output of test case of Factorial function



The screenshot shows the OneCompiler interface with a Haskell file named `Main.hs`. The code defines a `factorial` function and a `main` function. The `factorial` function is defined as follows:

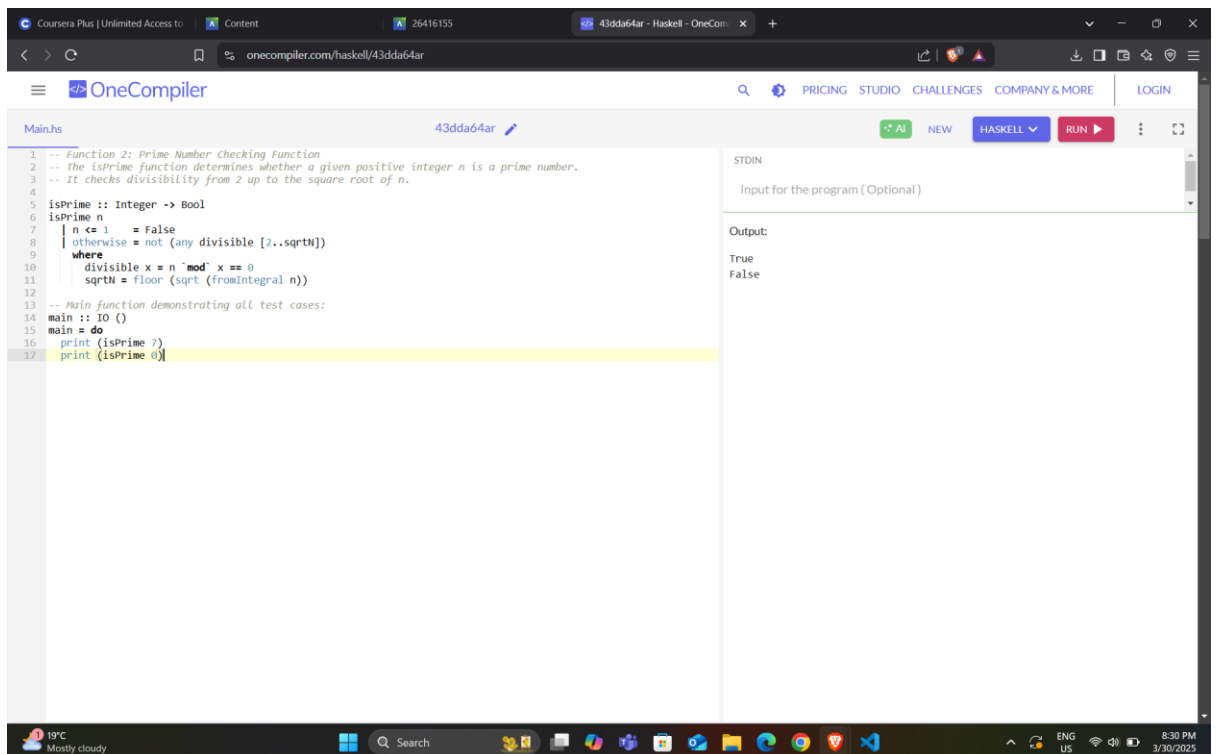
```
1 -- Function 1: Factorial Function
2 -- The factorial function calculates the factorial of a given integer n.
3 -- It handles edge cases such as n = 0 and negative numbers.
4 factorial :: Integer -> Integer
5 factorial n
6   | n < 0   = error "factorial is not defined for negative numbers"
7   | n == 0   = 1
8   | otherwise = n * factorial (n - 1)
9
10
11 -- Main function demonstrating all test cases:
12 main :: IO ()
13 main = do
14   print (factorial 7)
15   print (factorial 0)
```

The output of the program is shown on the right side of the interface:

```
STDIN
Input for the program (Optional)

Output:
5040
1
```

## 2. Screenshot of Output of test case of Prime Checking function



The screenshot shows the OneCompiler interface with a Haskell file named `Main.hs`. The code defines an `isPrime` function and a `main` function. The `isPrime` function is defined as follows:

```
1 -- Function 2: Prime Number Checking Function
2 -- The isPrime function determines whether a given positive integer n is a prime number.
3 -- It checks divisibility from 2 up to the square root of n.
4 isPrime :: Integer -> Bool
5 isPrime n
6   | n <= 1   = False
7   | otherwise = not (any divisible [2..sqrtN])
8   where
9     divisible x = n `mod` x == 0
10    sqrtN = floor (sqrt (fromIntegral n))
11
12 -- Main function demonstrating all test cases:
13 main :: IO ()
14 main = do
15   print (isPrime 7)
16   print (isPrime 0)
```

The output of the program is shown on the right side of the interface:

```
STDIN
Input for the program (Optional)

Output:
True
False
```

### 3. Screenshot of Output of test case of Fibonacci function

The screenshot shows the OneCompiler interface with a Haskell file named `Main.hs`. The code defines a recursive `fibonacci` function and a `main` function that tests it with `7` and `0`. The output on the right shows the results of these tests.

```
1 -- Function 3: Fibonacci Function
2 -- The fibonacci function generates the nth Fibonacci number using recursion.
3
4 fibonacci :: Integer -> Integer
5 fibonacci n
6   | n < 0  = error "Fibonacci is not defined for negative numbers"
7   | n == 0  = 0
8   | n == 1  = 1
9   | otherwise = fibonacci (n - 1) + fibonacci (n - 2)
10
11 -- Main function demonstrating all test cases:
12 main :: IO ()
13 main = do
14   print (fibonacci 7)
15   print (fibonacci 0)
```

STDIN  
Input for the program (Optional)

Output:  
13  
0

### 4. Screenshot of Output of test case of ReverseList function

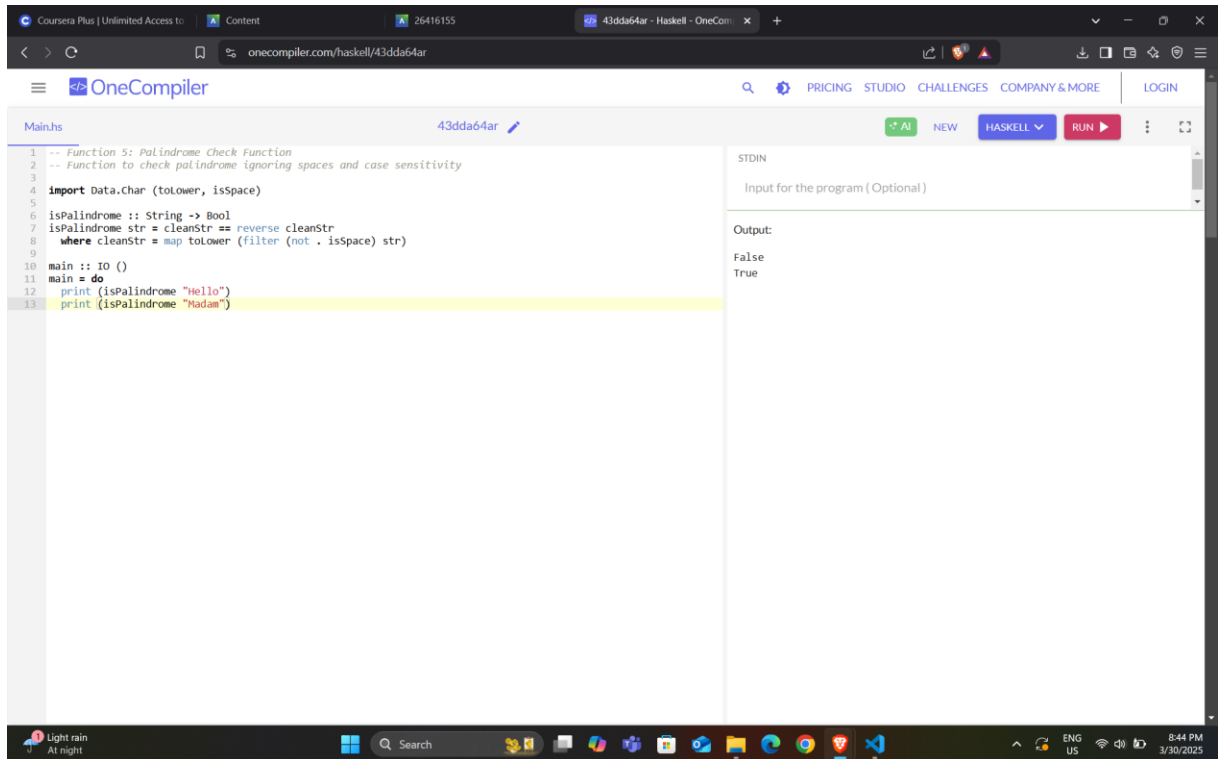
The screenshot shows the OneCompiler interface with a Haskell file named `Main.hs`. The code defines a recursive `reverseList` function and a `main` function that tests it with a list of integers and a list of characters. The output on the right shows the reversed lists.

```
1 -- Function 4: ReverseList Function
2 -- The reverseList function reverses a list using recursion.
3
4 reverseList :: [a] -> [a]
5 reverseList [] = []
6 reverseList (x:xs) = reverseList xs ++ [x]
7
8 -- Main function demonstrating all test cases:
9 main :: IO ()
10 main = do
11   print (reverseList [-5,3,25,11,7])
12   print (reverseList ['a','b','c','d','e'])
```

STDIN  
Input for the program (Optional)

Output:  
[7,11,25,3,-5]  
"edcba"

## 5. Screenshot of Output of test case of Palindrome function



The screenshot shows the OneCompiler web interface. The code editor on the left contains the following Haskell code:

```
1 -- Function 5: Palindrome Check Function
2 -- Function to check palindrome ignoring spaces and case sensitivity
3
4 import Data.Char (toLower, isSpace)
5
6 isPalindrome :: String -> Bool
7 isPalindrome str = cleanStr == reverse cleanStr
8   where cleanStr = map toLower (filter (not . isSpace) str)
9
10 main :: IO ()
11 main = do
12   print (isPalindrome "Hello")
13   print (isPalindrome "Madam")
```

The right-hand side of the interface shows the execution results. Under the 'STDIN' section, there is a text input field labeled 'Input for the program (Optional)'. Under the 'Output' section, the results of the two print statements are listed:

False  
True

The bottom of the image shows a Windows taskbar with the date and time as 8:44 PM on 3/30/2025.