**GO Util Package Documentation**

1. Utility Package Integration

A screen shot of a computer

Description automatically generated

IsPrime

A computer screen shot of a code

Description automatically generated

Uses a loop to check for division from 2 to check if there are divisors and returns it again indicating that it is prime

Reverse String

A computer screen with text

Description automatically generated

Revereses the character of any given string iterates through the length of the string to swap the characters from beginning to end.

SumOfSlice

A computer screen with text

Description automatically generated

Computes the sum of all integers in a given slice it uses a for loop in range of the slice to add all the elements and return the results.

Factorial

A screen shot of a computer code

Description automatically generated

Calculates the factorial of a given integer n.

1. Unit Tests

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

IsPrime

A screen shot of a computer program

Description automatically generated

ReverseString

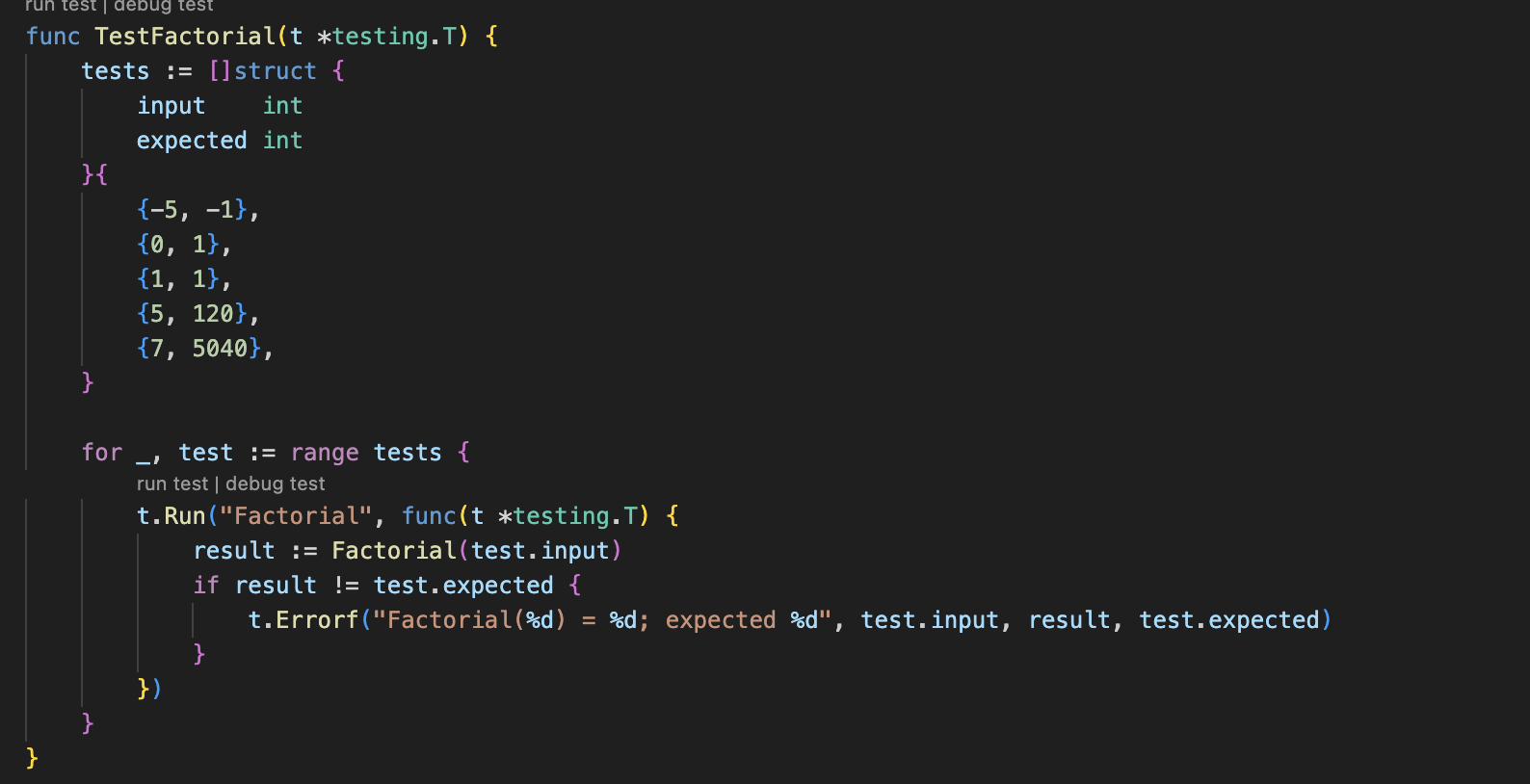


SumOfSlice

A computer screen shot of a program code

Description automatically generated

Factorial



1. Benchmarks

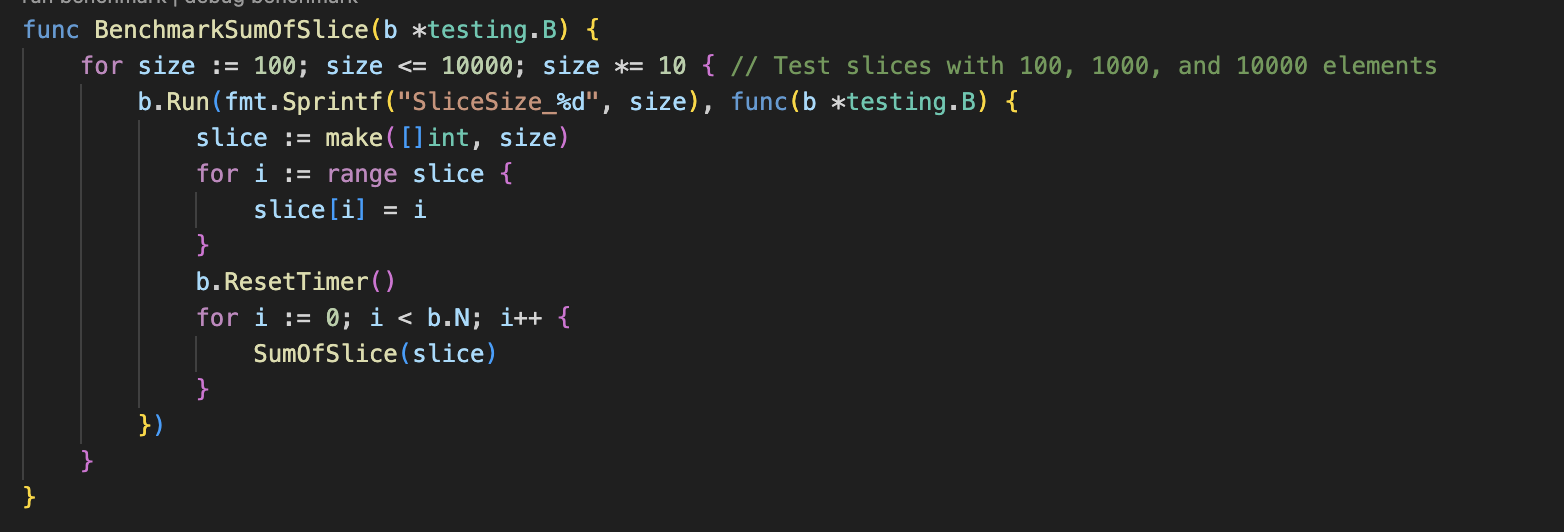
ReverseString

A computer code on a black background

Description automatically generated  
IsPrime

A black screen with numbers and text

Description automatically generated  
SumOfSlice



1. Coverage Analysis

Output from “go test -coverprofile=coverage.out”

A screenshot of a computer

Description automatically generated

Output from “got test -cover”

A screen shot of a computer

Description automatically generated

We achieved the necessary requirement of 90% and achieved 94.7%

As seen in the output from coverage.out there is 1 0 next to 4.12,16.3 which is this specific part of the code below

A black screen with colorful text

Description automatically generated

It wasn’t covered due to no test cases being provided where N is equal to zero.

**Table Driven Tests**

Provided a clear and organized way to to test various different scenarios for each of the functions. By defining the inputs and expected outputs in a format, it became easier to cover edge cases and explore different testing scenarios. This approach saved time and ensured proper and a well comprehensed testing for the functions.

**Key Finding from Benchmark Tests**

The Benchmark tests revealed that each of the functions performed exceptionally with the exceptions of IsPrime where it showed slower performance for larger numbers due to the loop used.