# CS370 – ASSIGNMENT #10

## NAME: Kostiantyn Makrasnov

**GRADE:**

|  |  |  |
| --- | --- | --- |
| **CATEGORY** | **POINTS** |  |
| EX10\_01 |  | 30 |
| EX10\_02 |  | 30 |
| EX10\_03 |  | 40 |
| EC10\_01 |  | 10 |
| EC10\_02 |  | 20 |
|  |  |  |
| **TOTAL** |  | 100 |

## EXERCISES:

**EX10\_01 –** Write a Haskell function to calculate the sum of all numbers in a given list  
Do this in two ways:  
1. using the list pattern matching we saw earlier  
2. using the foldl function in Haskell

**Both done in ./EX\_01.**hs

**EX10\_02 –** Write a Haskell function to calculate the product of all numbers in a list  
Again do this two ways:  
1. using the list pattern matching we saw earlier  
2. using the foldl function in Haskell

**Both done in ./EX\_02.**hs

**EX10\_03 –** Write a Haskell function that takes a list and a binary operator (e.g. +) and calculates that operator over the entire list. Test your function with +, \*.

**Both done in ./EX\_03.**hs

**EC10\_01 –** Use the function written in EX10\_03 to write two new Haskell functions. The first takes a parameter *n* and computes ∑*n*, and the second takes a parameter *n* and computes *n*!

**Both done in ./EX\_04.**hs

**EC10\_02 –** Prove using induction that your function in EX10\_03 using the + function is equivalent to the one in EX10\_01(part 1)

Let LHS be EX10\_01 (part 1) and RHS be EX10\_03

**Base Case:**

LHS listSumPattern [] = 0

RHS listOpExec (+) [] = 0

LHS = RHS

**Inductive Case:**

Assume n is starting element and m is the last element and k is the second to last element in [n … k, m]

Assume listSumPattern [n ... k] = listOpExec (+) [n ... k] = sum of all elements from element n to element k

Then listSumPattern [n ... k] + m = listOpExec (+) [n ... k] + m

Sum (n-k) + m = Sum(n-k) + m

LHS = RHS