Homework 5

Your task is to define the Haskell functions below.

Please submit all function definitions in a single file with your team name included in the top level comment.

### **Question 1:**

Write a Haskell function top that takes a list of elements and a threshold as arguments and returns a new list that contains the elements of the original list that are greater or equal to the given threshold.

Here are some test examples:

> top [8, 7, 4, 5, 10] 7

**[8,7,10]**

> top "SPARTANS" 'P'

**"SPRTS"**

> top [1..50] 80

**[]**

**To get any credit on this question, you must use a built-in higher order function (in a non-trivial way) in your implementation.** Please include the type declaration for the function for full credit.

### **Question 2:**

Write a Haskell function skip that takes a list of elements of arbitrary types and returns a new list that contains every other element of the original list starting with the first one.

Here are some test examples:

> skip "abcdefgh"

**"aceg"**

> skip [1..10]

**[1,3,5,7,9]**

> skip [1]

**[1]**

> skip [5, 10, 2]

**[5, 2]**

> skip []

**[]**

Please include the type declaration for the function for full credit.

### **Question 3:**

Write a Haskell function  mix  that takes two lists, and returns a new list formed by alternating the elements of both lists. If one of the lists is longer than the other, the extra elements of the longer list are included at the end of the new list.  Here are some examples:

> mix [1..3] [50, 60..100]

**[1,50,2,60,3,70,80,90,100]**

> mix [1..10] [50..55]

**[1,50,2,51,3,52,4,53,5,54,6,55,7,8,9,10]**

> mix [1] [2]

**[1,2]**

> mix "Go" "Spartans"

"GSopartans"

Please include the type declaration for the function for full credit.

### **Question 4:**

Write a Haskell function  inCommon  that takes two lists of the same type, and returns a new list formed by the elements that appear in both lists.  **The new list does not contain any duplicates.**

Here are some examples:

> inCommon "Hello World!" "Go Spartans!"

**" or!"**

> inCommon [1..10] [2, 4..20]

**[2,4,6,8,10]**

> inCommon [1, 2, 3, 4, 1, 2, 3, 4, 3, 2] [1..3]

**[1,3,2]**

Please include the type declaration for the function for full credit.

### **Question 5:**

Write a Haskell function shortest that takes a list of lists as its argument.  If the list is empty, the function returns Nothing.  Otherwise, the function returns Just the shortest list.

Here are some examples:

> shortest ["SJSU", "Go", "Spartans"]

**Just "Go"**

> shortest [[1..1000], [100..103], [80..100]]

**Just [100,101,102,103]**

> shortest []

**Nothing**

Please include the type declaration for the function for full credit.

### **Question 6:**

Consider the data type ExpTree defined as follows:

data ExpTree  = Atom Double  
               | Plus ExpTree  ExpTree  
               | Times ExpTree  ExpTree  
        deriving Show

Write a function evaluate that takes **one argument** of type ExpTree and returns a Double representing the value of the corresponding expression.  The value of an atom is simply the number associated with that Atom.  The value of a Plus node is the sum of the values of the corresponding subexpressions.  The value of a Times node is the product of the values of the corresponding subexpressions.

Here are some examples:

> evaluate (Atom 10)

**10.0**

> evaluate (Plus (Atom 10) (Atom 5))

**15.0**

> evaluate (Times (Plus (Atom 10) (Atom 5)) (Atom 4))

**60.0**

> evaluate (Plus (Atom 1) (Times (Plus (Atom 10) (Atom 5)) (Times (Atom 2) (Atom 3))))

**91.0**

Please include the type declaration for the function for full credit.

### **Question 7:**

Write a Haskell function allNegative that takes a list of numbers as its arguments.  The function returns True if all the numbers in the less are less than 0.  The function returns False otherwise.  **To get any credit on this question, your implementation must use one of the fold functions** (in a non-trivial way).

Here are some examples:

> allNegative []

**True**

> allNegative [-6.5]

**True**

> allNegative [-6, -2, -100]

**True**

> allNegative [-2..10]

**False**

Please include the type declaration for the function for full credit.

### **Question 8:**

Write a Haskell function geoSequence that takes two numbers, say a and r as arguments.  The function returns an infinite list that represents the geometric sequence: [a, ar, ar2, ar3, ar4, ar5, ...].

Because the function returns an infinite list, be careful how you test it.

Here are some test cases:

> take 10 (geoSequence 5 10)

**[5,50,500,5000,50000,500000,5000000,50000000,500000000,5000000000]**

> take 5 (geoSequence 1 0.5)

**[1.0,0.5,0.25,0.125,6.25e-2]**

> sum(take 1000 (geoSequence 1 0.5))

**2.0**

Please include the type declaration for the function for full credit.





