Functional Programming HW

Ashley, Harrison, Jenna, Jing, Parmanand

HOMEWORK INSTRUCTIONS

1. Please select one of the three levels below to complete:

   Explorer Adventurer Pioneer

2. To submit your homework, please create a file named “homework.md” and save it in the functional_programming folder located within your class GitHub repository.

3. Copy and paste both your response to the writing prompt and your code to the homework.md file.

Level Explorer

Task 1: Thoughts?

How would you rate coding in Racket from a scale of 1 to 5? In your opinion, how does coding in Racket compare to coding in Java or Python? Do you find Racket's syntax and approach more or less intuitive than those of the other languages? Explain your reasoning and provide examples to support your views.

Task 2: The Code

1. Define a variable called 'num1' and assign it the value 5.
2. Define another variable called 'num2' and assign it the value 10.
3. Create a function called 'add-numbers' that takes 'num1' and 'num2' as arguments and returns their sum.
4. Test the 'add-numbers' function by calling it with 'num1' and 'num2' as arguments and printing the result.
5. Apply the same approach to create a function called 'double' that takes an integer as its input and returns twice the value of that integer. Then use this function to print the value of 'double 5'.

Level Adventurer

Task 1: Thoughts?

How would you rate coding in Racket from a scale of 1 to 5? In your opinion, how does coding in Racket compare to coding in Java or Python? Do you find Racket's syntax and approach more or less intuitive than those of the other languages? Explain your reasoning and provide examples to support your views.

Task 2: The Code

You can choose to complete **either one** of the two programming exercises listed below:

1. is-divisible
2. is-triangle

Hint: You may find it helpful to refer to your Java code from the summer pre-work assignment "pre05" as a reference while working on this task!

Level Pioneer

Task 1: Thoughts?

How would you rate coding in Racket from a scale of 1 to 5? In your opinion, how does coding in Racket compare to coding in Java or Python? Do you find Racket's syntax and approach more or less intuitive than those of the other languages? Explain your reasoning and provide examples to support your views.

Task 2: The Code

Please complete 'ack' programming exercise.

Hint: You may find it helpful to refer to your Java code from the summer pre-work assignment "pre05" as a reference while working on this task!

is-divisible

Write a method named `is-divisible` that takes two integers, `n` and `m`, and that returns `true` if `n` is divisible by `m`, and `false` otherwise

Sample Outputs:

```
 8  (is-divisible 4 2)   ; #t
 9  (is-divisible 12 4)  ; #t
10  (is-divisible 7 3)   ; #f
11  (is-divisible 32 7)  ; #f
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 256 MB.
#t
#t
#f
#f
```

is-triangle

If you are given three sticks, you may or may not be able to arrange them in a triangle. For any three lengths, there is a simple test to see if it is possible to form a triangle:

If any of the three lengths is greater than the sum of the other two, you cannot form a triangle.

Write a method name `is-triangle` that takes three integers as arguments and return either `true` or `false`, depending on whether you can or cannot form a triangle from sticks with the given lengths.

Sample Outputs:

```
14 (is-triangle 3 4 5)   ; #t
15 (is-triangle 7 1 2)   ; #f
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 256 MB.
#t
#f
```

ack

The Ackermann function is defined for non-negative integers as follows:

$$A(m, n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m, n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

Write a method called `ack` that takes two integers, `m` and `n`, as parameters and that computes and returns the value of the Ackermann function. Define your function recursively with conditions.

Sample Outputs:

```
19 (ack 1 2) ; 4
20 (ack 2 3) ; 9
21 (ack 3 4) ; 125
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 256 MB.
4
9
125
```