CMSC330 Spring 2023 Quiz 1

Proctoring TA: _____ **Name:** _____

UID: _____

Problem 1: Basics

[Total 5 pts]

Please circle **True** or **False** for the following statements:

Ruby uses a static type system	**True**	**False**
Procs and Codeblocks can be used interchangeably	**True**	**False**
`nil` is not an object in Ruby	**True**	**False**
In Ruby, types are associated with values	**True**	**False**
Ruby has built in support for Regular Expressions	**True**	**False**

Problem 2: Code Completion

[Total 10 pts]

Fill in the blanks of the following code so that is has the desired output

(a) Higher Order Programming [4 pts]

Fill in the blanks so `10` is printed. You cannot hard code blank 1 or blank 2. You must use `x` and the codeblock.

```
def myfunc(x)
    puts ___BLANK_1____
end

myfunc(3){__BLANK_2___}
```

blank 1:

```
yield x
```

Blank 2:

```
{|x| x + 7}
```

(b) Creation [2 pts]

Fill in the blanks so that `a` is a Hash with a default value of an Array of size 3

```
a = ___blank_1__
```

Blank 1:

Hash.new(Array.new(3))

(c) Objects [2 pts]

Fill in the blank so that square has a **class** variable called `length` with the value of `x`

```
class Square
    def initialize(x)
        __Blank_1__
    end
end
```

Blank 1:

@@length = x

(d) Regex [2 pts]

Fill in the blanks so that "Correct" is printed

```
rxp = /__Blank_1____/
line1 = "23 years of age"
line2 = "1 year of age"
if rxp =~ line1 && rxp =~ line2
    puts "Correct"
else
    puts "Failed"
```

Blank 1:

[0-9]+ years? of age

Problem 3: Coding

[Total 5 pts]

Write a method named `procHash(hash)`. The argument `hash` is a hash from a numerical key to a Proc. For each key and proc pair, print out **"RESULT is the result of Proc(KEY)"** where **KEY** is the key and **RESULT** is the result of calling the proc associated with the key on said key.

Example:

`procHash({1=>Proc.new{|x| x * 2}, 2=>Proc.new{|x| x + 3}})` prints out

```
2 is the result of Proc(1)
5 is the result of Proc(2)
```

```
def procHash(x)
```

```
    for k,v in x
        puts v.call(k).to_s + " is the result of Proc(" + k.to_s + ")"
    end
end
```