Ethan Roberts

Ruby Assignment Output_part02

CS 417

Roberts.E.assn05_part02

Problem 1 (source-code and output):

```
# Ethan Roberts
# CS 417 Topics in OOP

# This script finds all positive ints
# less than "n" that are not mcnugget numbers

# Ruby Assignment

# cite:  https://stackoverflow.com/questions/12112765/how-to-reference-
global-variables-and-class-variables
# used this link for syntax help with defining a global variable

# cite:  http://mathworld.wolfram.com/McNuggetNumber.html
# used this link to get non-mcnugget values

# cite:  https://launchschool.com/books/ruby/read/loops_iterators
# used to reference using "until" loop




$mcnuggetAry = [1,2,3,5,7,11]  #global array holding all non-mcnugget nums

def findNonMcnuggetNums(value)

  print "\n--User entry: "
  puts value
  print "--Non-mcnugget integer(s): "
  i = 0
  if value > 11
    $mcnuggetAry.each do |n|
      print " "
      print n
    end
   else
     while ($mcnuggetAry[i] < value)
       print $mcnuggetAry[i]
       print " "
       i = i + 1
```

```ruby
      end
    end
    print "\n\n"
end



# Loop for displaying output and
# accuracy of program

  x = 0
  y = 1
  until x > 14
    findNonMcnuggetNums(y)
    x = x + 1
    y = y + 1
  end
```

C:\WINDOWS\system32\cmd.exe

```
C:\Users\Ethan\Desktop\MyRepos\RubyAssignment2>ruby problem1Script.rb

--User entry: 1
--Non-mcnugget integer(s):


--User entry: 2
--Non-mcnugget integer(s): 1


--User entry: 3
--Non-mcnugget integer(s): 1 2


--User entry: 4
--Non-mcnugget integer(s): 1 2 3


--User entry: 5
--Non-mcnugget integer(s): 1 2 3


--User entry: 6
--Non-mcnugget integer(s): 1 2 3 5
```

C:\WINDOWS\system32\cmd.exe

```
--User entry: 6
--Non-mcnugget integer(s): 1 2 3 5


--User entry: 7
--Non-mcnugget integer(s): 1 2 3 5


--User entry: 8
--Non-mcnugget integer(s): 1 2 3 5 7


--User entry: 9
--Non-mcnugget integer(s): 1 2 3 5 7


--User entry: 10
--Non-mcnugget integer(s): 1 2 3 5 7


--User entry: 11
--Non-mcnugget integer(s): 1 2 3 5 7
```

C:\WINDOWS\system32\cmd.exe

```
--User entry: 10
--Non-mcnugget integer(s): 1 2 3 5 7


--User entry: 11
--Non-mcnugget integer(s): 1 2 3 5 7


--User entry: 12
--Non-mcnugget integer(s):  1 2 3 5 7 11


--User entry: 13
--Non-mcnugget integer(s):  1 2 3 5 7 11


--User entry: 14
--Non-mcnugget integer(s):  1 2 3 5 7 11


--User entry: 15
--Non-mcnugget integer(s):  1 2 3 5 7 11


C:\Users\Ethan\Desktop\MyRepos\RubyAssignment2>
```

Problem 2 (source-code and output):

```
# Ethan Roberts
# CS 417 Topics in OOP

# This script deals with light switches and lights


# Ruby Assignment

#  cite: https://ruby-doc.org/docs/ruby-doc-
bundle/Tutorial/part_02/user_input.html
#  Used this link to teach myself how to get user-input from terminal

#  cite: https://apidock.com/ruby/String/to_i
#  Used to learn how to convert terminal input to int


class Switch

def initialize  # self-note:  initialize is a ruby-defined "constructor"
     @maxToggle = 3  #@ sign means instance variable
     @numOfSwitches = 0
     @remainder = 0
end



def findNumberOfLights(entry)
   if entry < 2
     return 0
   elsif entry == 2
     return 1
   else
     @remainder = entry % @maxToggle  #checking to see if there is a
remainder
     if @remainder > 0
       return ((entry / @maxToggle) + 1)
     else
      return (entry / @maxToggle)
     end
   end
end


end




mySwitch = Switch.new()
myAnswer = 0
entry = 0
```

```ruby
i = 0

# get user input 12 times to show accuracy of program
while (i < 12)
  print "\nEnter number of lights you need to turn on: "
  entry = gets.to_i
  myAnswer = mySwitch.findNumberOfLights(entry)
  if (myAnswer == 0)
    puts "Entry must be two or more lights...try again."
  else
    print "Total light-switches needing flipped: "
    puts myAnswer
  end
  i = i + 1
end
```

C:\WINDOWS\system32\cmd.exe

```
C:\Users\Ethan\Desktop\MyRepos\RubyAssignment2>ruby problem2Script.rb

Enter number of lights you need to turn on: 1
Entry must be two or more lights...try again.

Enter number of lights you need to turn on: 2
Total light-switches needing flipped: 1

Enter number of lights you need to turn on: 0
Entry must be two or more lights...try again.

Enter number of lights you need to turn on: 3
Total light-switches needing flipped: 1

Enter number of lights you need to turn on: 4
Total light-switches needing flipped: 2

Enter number of lights you need to turn on: 5
Total light-switches needing flipped: 2

Enter number of lights you need to turn on: 6
Total light-switches needing flipped: 2

Enter number of lights you need to turn on: 27
Total light-switches needing flipped: 9
```

C:\WINDOWS\system32\cmd.exe

```
Total light-switches needing flipped: 9

Enter number of lights you need to turn on: 18
Total light-switches needing flipped: 6

Enter number of lights you need to turn on: 19
Total light-switches needing flipped: 7

Enter number of lights you need to turn on: 16
Total light-switches needing flipped: 6

Enter number of lights you need to turn on: 10
Total light-switches needing flipped: 4

C:\Users\Ethan\Desktop\MyRepos\RubyAssignment2>
```

Problem 3 (source-code and output):

```
# Ethan Roberts
# CS 417 Topics in OOP

# This script deals with a single-elimination tournament


# Ruby Assignment



class Tournament


def initialize  # self-note:  initialize is a ruby-defined "constructor"
     @playersPerMatch = 2
     @playersInTourney = 0
end


def getRounds(entry)

    counter = 0
    dividend = 0
    remainder = 0
    @playersInTourney = entry

    dividend = entry / @playersPerMatch # first round
    remainder = entry % 2
    dividend = dividend + remainder
    counter = counter + 1

    while (dividend > 1)
       remainder = dividend % 2
       dividend = dividend / @playersPerMatch
       dividend = dividend + remainder
       counter = counter + 1

    end

     return counter  # number of matches played in tournament

end



def getMatches
  return @playersInTourney - 1  # matches played formula:  number of players
- 1
end



end
```

```
myGame = Tournament.new()
myAnswer = 0
entry = 0
i = 0


#while-loop to show program accuracy...runs progam 10 times
while (i < 10)

  print "\nEnter number of players playing: "
  entry = gets.to_i

  myAnswer = myGame.getRounds(entry)
  print myAnswer
  print " round(s) in this tournament.\n"

  myAnswer = myGame.getMatches
  print myAnswer
  print " match(es) in this tournament.\n"


  myAnswer = myAnswer - 1 #formula to get second-best:  matches played - 1
  print myAnswer
  print " match(es) need to be played to get second-best player.\n"

  i = i + 1
 end
```

C:\WINDOWS\system32\cmd.exe

```
C:\Users\Ethan\Desktop\MyRepos\RubyAssignment2>ruby problem3Script.rb

Enter number of players playing: 2
1 round(s) in this tournament.
1 match(es) in this tournament.
0 match(es) need to be played to get second-best player.

Enter number of players playing: 3
2 round(s) in this tournament.
2 match(es) in this tournament.
1 match(es) need to be played to get second-best player.

Enter number of players playing: 4
2 round(s) in this tournament.
3 match(es) in this tournament.
2 match(es) need to be played to get second-best player.

Enter number of players playing: 5
3 round(s) in this tournament.
4 match(es) in this tournament.
3 match(es) need to be played to get second-best player.

Enter number of players playing: 6
3 round(s) in this tournament.
```

C:\WINDOWS\system32\cmd.exe

```
3 round(s) in this tournament.
5 match(es) in this tournament.
4 match(es) need to be played to get second-best player.

Enter number of players playing: 7
3 round(s) in this tournament.
6 match(es) in this tournament.
5 match(es) need to be played to get second-best player.

Enter number of players playing: 8
3 round(s) in this tournament.
7 match(es) in this tournament.
6 match(es) need to be played to get second-best player.

Enter number of players playing: 9
4 round(s) in this tournament.
8 match(es) in this tournament.
7 match(es) need to be played to get second-best player.

Enter number of players playing: 15
4 round(s) in this tournament.
14 match(es) in this tournament.
13 match(es) need to be played to get second-best player.
```

C:\WINDOWS\system32\cmd.exe

```
6 match(es) in this tournament.
5 match(es) need to be played to get second-best player.

Enter number of players playing: 8
3 round(s) in this tournament.
7 match(es) in this tournament.
6 match(es) need to be played to get second-best player.

Enter number of players playing: 9
4 round(s) in this tournament.
8 match(es) in this tournament.
7 match(es) need to be played to get second-best player.

Enter number of players playing: 15
4 round(s) in this tournament.
14 match(es) in this tournament.
13 match(es) need to be played to get second-best player.

Enter number of players playing: 14
4 round(s) in this tournament.
13 match(es) in this tournament.
12 match(es) need to be played to get second-best player.

C:\Users\Ethan\Desktop\MyRepos\RubyAssignment2>
```

Test Data:

Problem 1:
Enter any number (must be a positive integer) and the program will return back to the user the number of non-mcnugget numbers that are less than the user's entry.
(Data used to test: 1, 2, 3, 4 ,5 6, 7, 8, 9, 10, 11 ,12, 13, 14, 15)

Problem 2:
Enter any number (must be greater than 2) and the program will return back to the user the number of light-switches needing to be flipped in order to turn all lights on.
(Data used to test: 1, 2, 0, 3, 4, 5, 6, 27, 18, 19, 16, 10

Problem 3:
Enter any number (must be an interger greater than zero) and the program will return back to the user three different answers regarding number of matches needing to be played based on user-entered number of players in tournament, number of rounds in the tournament, and the number of matches needing to be played to get second-best player.
(Data used to test: 2, 3, 4 ,5 6, 7, 8, 9, 15, 14)