Quiz 4

1. Ruby Program that prints the Time in different time zones

```
def showdifferentzone
        time1 = Time.now
        time2 = Time.utc(2000,"jan",1,20,15,1)
        time3 = Time.local(2000,"jan",1,20,15,1)
        time4 = Time.gm(2000,"jan",1,20,15,1)
        print time1
        print "\n"
        print time2
        print "\n"
        print time3
        print "\n"
        print time4
        print "\n"
end
showdifferentzone()
```

2. Write a program that iterates over an array and builds a new array that is the result of incrementing each value in the original array by a value of 2. You should have two arrays at the end of this program,

The original array and the new array you've created.Print both arrays to the screen using the p method instead of puts.

```
print "original array is:"

arrayfirst = Array[6,4,2,16,8,2,13,5,7,9]

p arrayfirst

print "after incrementing 2 of each value, array is:"

arraysecond = []

arrayfirst.each do |x|

arraysecond<< x+2

end

p arraysecond
```

3. Ruby program to find the leap year when start and end year are given.

```
puts 'This program will determine what leap years are between two specific dates that you assign.'

puts ''

puts 'What would you like your starting year to be?'

start_year = gets.chomp.to_i

puts ''

puts 'What would you like your ending year to be?'

end_year = gets.chomp.to_i

puts "Leap years between #{start_year} and #{end_year}"


if start_year > end_year
  puts 'Your closing date is before your start date. Please try again'
  return
end


leap_years = []

start_year.upto(end_year) do |year|
  if (year % 4 == 0)
    leap_years << year unless (year % 100 == 0) and (year % 400 != 0)
  end
end


puts leap_years.join(', ')
```

4. Ruby program that takes a numerical value and give the output as Roman number

```
@digits = {
  1000 => "M",
   900 => "CM", 500 => "D", 400 => "CD",  100 => "C",
    90 => "XC",  50 => "L",  40 => "XL",   10 => "X",
     9 => "IX",   5 => "V",   4 => "IV",    1 => "I"
}

def romanize(num)
  @digits.keys.each_with_object('') do |key, str|
    nbr, num = num.divmod(key)
    str << @digits[key]*nbr
  end
end

puts romanize(888)
puts romanize(999)
```

5. Write a your own ruby program that uses a Queue

```
Queue1 = Queue.new
Queue1.enq(703)
puts "Enqueing 703"
Queue1.enq(912)
puts "Enqueing 912"
Queue1.enq(213)
puts "Enqueing 213"
```

```
size = Queue1.length
for i in 0..size-1
        popped = Queue1.deq
        print "Popped:"
        puts popped
end
```

6. Write your own ruby program that uses each_with_index method to iterate through an array that prints each index and value

```
[:Shahed, :Sazzad, :Omor, :Satish].each_with_index do |value, index|
   puts "#{index}: #{value}"
end
```

7. Ruby Program that prints if duplicates existing in a array

```
array = [1, 2, 2, 2, 3, 4, 5, 8, 8, 11]

array.inject(Hash.new(0)) { |hash,val|
 hash[val] += 1;
 hash
}.each_pair { |val,count|
 puts "#{val} -> #{count}" if count > 1
}
```

8. Write a Ruby program that prints pascal triangle

```
def pascal(n)
 raise ArgumentError, "must be positive." if n < 1
 yield ar = [1]
```

```
  (n-1).times do
    ar.unshift(0).push(0) # tack a zero on both ends
    yield ar = ar.each_cons(2).map{|a, b| a + b }
  end
end

pascal(8){|row| puts row.join(" ").center(20)}
```

9. Write a Ruby program that prints the length of the common string when two strings are compared.

```
def self.find_longest_common_substring(s1, s2)
    if (s1 == "" || s2 == "")
      return ""
    end
    m = Array.new(s1.length){ [0] * s2.length }
    longest_length, longest_end_pos = 0,0
    (0 .. s1.length - 1).each do |x|
      (0 .. s2.length - 1).each do |y|
        if s1[x] == s2[y]
          m[x][y] = 1
          if (x > 0 && y > 0)
            m[x][y] += m[x-1][y-1]
          end
          if m[x][y] > longest_length
            longest_length = m[x][y]
            longest_end_pos = x
            puts longest_length
          end
        end
```

```
    end
  end
  return s1[longest_end_pos - longest_length + 1 .. longest_end_pos]
end
puts find_longest_common_substring("sazzad", "shahed")
```