**9.Program that print the different time zone :**

puts now = Time.now

puts now = now.gmtime

puts now = now.localtime

**sample output:**

2015-10-19 23:44:53 +0000

2015-10-19 23:44:53 UTC

2015-10-19 23:44:53 +0000

**10.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.**

arr = [1, 2, 3, 4, 5]

new\_arr = []

arr.each do |n|

 new\_arr << n + 2

end

p arr

p new\_arr

**11.# program to find the leap year when start and end year are given.**

puts 'Input a starting year:'

start\_year = gets.chomp

puts 'Input an ending year:'

end\_year = gets.chomp

puts ''

while start\_year.to\_i <= end\_year.to\_i

if start\_year.to\_f%400 == 0

puts start\_year

elsif start\_year.to\_f%100 == 0

elsif start\_year.to\_f%4 == 0

puts start\_year

end

start\_year = start\_year.to\_i + 1

end

**Sample output:**

Input a starting year:

1978

Input an ending year:

2000

1980

1984

1988

1992

1996

2000

**12.# Write a program to write give a numerical value and give the output as Roman number**

def new\_roman\_numeral num

 values =   [1000, 900, 500, 400, 100,90, 50, 40, 10, 9, 5, 4, 1]

 numerals = ["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"]

 roman\_numeral = ""

 i = 0

 while i <= 12

   while num >= values[i]

     num = num - values[i]

     roman\_numeral = roman\_numeral + numerals[i]

   end

   i = i + 1

 end

 roman\_numeral

end

print "Enter the number you want to convert to a new roman numeral: "

puts new\_roman\_numeral(gets.chomp.to\_i)

**Sample output:**

Enter the number you want to convert to a new roman numeral:  500

D

13.Write a program that uses Queue:

require 'thread'

# Queues are a First in First Out Data Structure

# Ruby provides you with synchronoized/thread-safe queues

testQueue = Queue.new

testQueue.enq(10)

puts "Enqueing 10"

testQueue.enq(12)

puts "Enqueing 12"

testQueue.enq(13)

puts "Enqueing 13"

while   ! testQueue.empty?

   popped = testQueue.deq

   puts "Popped : " + popped.to\_s

end

14.Write a program that find out the hour of the day and the number of minutes and seconds on that hour:

stop = Time.local( 20, 15, 1, 30, "jan", 2007, 2, 30, false, "MST")

puts stop.hour

puts stop.min

stop.sec

15.Program usingUse equal to check the references?

a = "Ruby"

b = c = "Ruby"

a.equal?(b)

b.equal?(c)

**16.Create a new array that contains any items from the array variable with a length a greater than 5 characters**

periodicTable = [ "Lithium", "Carbon", "Iron", "Tin", "Iridium" ]

elements = periodicTable.select { |e| puts e.length > 5 }

sample output:

true

true

false

false

true

17. # write a program to get a sum of the ages

ages = { John: 22, Mike: 49, Erin: 25}

puts ages.values.inject(:+)

sample output:

96

18. Using each\_with\_index method to iterate through an array that prints each index and value

top\_five\_games = ["mario brothers", "excite bike","ring king","castlevania","double dragon"]

top\_five\_games.each\_with\_index do | game, index |

 puts "#{index + 1}. #{game}"

end

19. # Define a hash and its members

pops = {"Maiden Bradley" => 335,

       "Malmesbury" => 4631,

       "Malmesbury St. Paul Without" => 1738,

       "Manningford" => 385,

       "Marden" => 124,

       "Market Lavington" => 2257,

       "Marlborough" => 8009,

       "Marston" => 145,

       "Marston Maisey" => 172,

       "Melksham (U.D.C. area 1881)" => 14204,

       "Melksham Without" => 6226}

# Show the contents - goodness only know what order

# they will be in, since it's 'hashed'

p pops

# Print an individual member out - note [ and ] rather than { and }

print "The population of Marston is #{pops["Marston"]}\n"

places = pops.keys

print "We know about #{places.join(", ")}\n"

**20.**Combining arrays using “and” & “or”

first = [1,3,5,7,9]

second = [2,3,4,6,7]

# | merges two arrays, but if there are equal elements

# in the two lists it removes the duplicates from the second

third = first | second

p third

# Let's do that again, but this time sort the merged list!

third = (first | second).sort

p third

# And this time we'll look for common elements using &

third = first & second

p third

**21.write a program that gives the width and height of box using to\_s**

class Box

  # constructor method

  def initialize(w,h)

     @width, @height = w, h

  end

  # define to\_s method

  def to\_s

     "(w:#@width,h:#@height)"  # string formatting of the object.

  end

end

# create an object

box = Box.new(10, 20)

# to\_s method will be called in reference of string automatically.

puts "String representation of box is : #{box}"

sample output:

String representation of box is : (w:10,h:20)

**22.Write a program that calculates area of box.**

# define a class

class Box

  # constructor method

  def initialize(w,h)

     @width, @height = w, h

  end

  # instance method by default it is public

  def getArea

     getWidth() \* getHeight

  end

  # define private accessor methods

  def getWidth

     @width

  end

  def getHeight

     @height

  end

  # make them private

  private :getWidth, :getHeight

  # instance method to print area

  def printArea

     @area = getWidth() \* getHeight

     puts "Big box area is : #@area"

  end

  # make it protected

  protected :printArea

end

# create an object

box = Box.new(10, 20)

# call instance methods

a = box.getArea()

puts "Area of the box is : #{a}"

# try to call protected or methods

**23.Program that prints if duplicates exist in a array**

def find\_duplicates(elements)

   encountered = {}

   # Examine all elements in the array.

   elements.each do |e|

    # If the element is in the hash, it is a duplicate.

    if encountered[e]

       puts "Dupe exists for: " << e

    else

       # Record that the element was encountered.

       encountered[e] = 1

    end

   end

end

24.Program that triple ADDS THE input

def triple\_add(a, b, c)

  x = a.to\_s + b.to\_s + c.to\_s

  "#{x}!"

End

25.write a program that gives the options of arithmetic operations and take the numbers and gives the output

def greeting

 puts "Hello!  Please type your name: "

 name = gets

 puts "It is nice to meet you #{name}.  I am a simple calculator application.  \n

 I can add, subtract, multiply, and divide.\n"

 name

end

# This method ask the user what type of calculation they would like to perform

# It returns the operation or an error for erroneous entry

def request\_calculation\_type

 puts "Type 1 to add, 2 to subtract, 3 to multiply, or 4 to divide two numbers: "

 operation\_selection = gets.to\_i

 if operation\_selection == 1

   "add"

 elsif operation\_selection == 2

   "subtract"

 elsif operation\_selection == 3

   "multiply"

 elsif operation\_selection == 4

   "divide"

 else

   "error"

 end

end

# This method performs the requested calculation

# It returns the result of the calculation

def calculate\_answer(operator, a, b)

 if operator == "add"

   a + b

 elsif operator == "subtract"

  a - b

 elsif operator == "multiply"

   a \* b

 elsif operator == "divide"

   a / b

 end

end

name = greeting

run\_calculator = 1

while run\_calculator == 1

 current\_calculation = request\_calculation\_type()

 if current\_calculation == "error"

   puts "I do not understand this type of calculation... Can we try again?"

 else

   puts "What is the first number you would you like to #{current\_calculation}: "

   first\_number = gets.to\_i

   puts "What is the second number you would like to #{current\_calculation}: "

   second\_number = gets.to\_i

   answer = calculate\_answer(current\_calculation, first\_number, second\_number)

   puts "The answer is #{answer}"

   puts "Type 1 to run another calcution or 2 to end: "

   run\_calculator = gets.to\_i

 end

end

26.Write a program that prints pascal triangle

def pascal n

     rows = []

     # generate data

     (0...n).each do |i|

        rows << if i.zero?

           [1]

        else

           rows[i-1].inject([0]) do |m, o|

              m[0...-1] << (m[-1] + o) << o

           end

        end

     end

     # calc field width

     width = rows[-1].max.to\_s.length

     # space out each row

     rows.collect! do |row|

        row.collect { |x| x.to\_s.center(2 \* width) }.join

     end

     # display triangle

     rows.each { |row| puts row.center(rows[-1].length) }

   end

   pascal (ARGV[0] || 10).to\_i

27.Write a program using switch

a = ["4"]

case a

when 1..4, 5

 puts "It's between 1 and 5"

when 6

 puts "It's 6"

when String

 puts "You passed a string"

else

 puts "You gave me #{a} -- I have no idea what to do with that."

End

28.Write a program that will read a file

begin

    input = File.new("x.txt", "r")

rescue

    print "Failed to open /etc/fstab for input. ", $!, "\n"

end

input.each {

    |i|

    puts i;

    }

input.close()

29:Give a file as input and that prints the whole file and then prints first 3 lines.

input\_file = ARGV.first

def print\_all(f)

 puts f.read

end

def rewind(f)

 f.seek(0)

end

def print\_a\_line(line\_count, f)

 puts "#{line\_count}, #{f.gets.chomp}"

end

current\_file = open(input\_file)

puts "First let's print the whole file:\n"

print\_all(current\_file)

puts "Now let's rewind, kind of like a tape."

rewind(current\_file)

puts "Let's print three lines:"

current\_line = 1

print\_a\_line(current\_line, current\_file)

current\_line = current\_line + 1

print\_a\_line(current\_line, current\_file)

current\_line = current\_line + 1

print\_a\_line(current\_line, current\_file)

30 program we can control how, if, or how many times we call the proc. For example, let's say there's something we want to do before and after some code is run.

def doSelfImportantly someProc

 puts 'Everybody just HOLD ON!  I have something to do...'

 someProc.call

 puts 'Ok everyone, I\'m done.  Go on with what you were doing.'

end

sayHello = Proc.new do

 puts 'hello'

end

sayGoodbye = Proc.new do

 puts 'goodbye'

end

doSelfImportantly sayHello

doSelfImportantly sayGoodbye

31. Write a method called convert that takes one argument which is a temperature in degrees Fahrenheit.

def convert(temperature\_in\_F)

 (temperature\_in\_F - 32) / 1.8 # This solution can cause problems.  Why?

End

32.

Ruby program that process the file and matches certain patterns

Dir.glob("\*.WMA") {|filename|

 file = File.new(filename)

 mtime = file.mtime

 new\_filename = "#{mtime.year}-#{mtime.month}-#{mtime.day}.wma"

 puts "Renaming #{filename} to #{new\_filename} ..."

 File.rename(filename, new\_filename)

}

33.Write a program that check if the string contains certain letters or numbers etc and write if the letter exist change it to another letter

my\_string = "abcdefg"

if my\_string.include? "cde"

  puts "String includes 'cde'"

end

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

def substing(s1, s2)

       return 0 if s1.empty? || s2.empty?

       num=Array.new(s1.size){Array.new(s2.size)}

       s1.scan(/./).each\_with\_index{|letter1,i|

           s2.scan(/./).each\_with\_index{|letter2,j|

                   if s1[i]==s2[j]

                       if i==0||j==0

                          num[i][j] = 1

                       else

                          num[i][j]  = 1 + num[i - 1][ j - 1]

                       end

                   else

                       if i==0 && j==0

                          num[i][j] = 0

                       elsif i==0 &&  j!=0  #First ith element

                          num[i][j] = [0,  num[i][j - 1]].max

                       elsif j==0 && i!=0  #First jth element

                           num[i][j] = [0, num[i - 1][j]].max

                       elsif i != 0 && j!= 0

                         num[i][j] = [num[i - 1][j], num[i][j - 1]].max

                       end

                   end

           }

       }

       num[s1.length - 1][s2.length - 1]

end

puts substing("room","roommate")

35.Write a program that Iterates over the hash

restaurant\_menu = { "Ramen" => 3, "Dal Makhani" => 4, "Coffee" => 2 }

restaurant\_menu.each do | item, price |

 puts "#{item}: $#{price}"

end