**14. when number is given it should find the day of the month**

**Program:**

print "Please enter an integer "

value = gets.chomp.to\_i

print "Testing the number #{value}\n"

# normal if, with elif and else

if value > 5

puts "Value is greater than 5"

elsif value > 2

puts "Greater that 2 but less than 5"

else

puts "Less than or equal to 2"

end

# an if that's a code modifier

puts "That's very big" if value > 10

# An unless and else

unless value > 5

puts "Value is not greater than 5"

else

puts "That's greater that 5"

end

# an unless that's a code modifier

puts "That's not very big" unless value > 10

# Case - any one condition (or 'else' if none

# of the conditions matches exactly)

case value

when 1,21,31

suffix = "st"

when 2,22

suffix = "nd"

when 3,23

suffix = "rd"

else

suffix = "th"

end

puts "It is the #{value}#{suffix} of the month"

**16. Write a program using case statement**

**Program:**

print "Enter top level domain: "

domain = gets.chomp

case domain

when "us"

puts "United States"

when "de"

puts "Germany"

when "sk"

puts "Slovakia"

when "hu"

puts "Hungary"

else

puts "Unknown"

end

**17. Program iterates numbers with upto**

**Program:**

# Go up from 3 to 5.

3.upto(5) do |x|

# Go up from x to x + 2.

x.upto(x + 2) do |y|

# Display variable.

print y, " "

end

# End the line.

print "\n"

end

sample out put:

3 4 5

4 5 6

5 6 7

**18. Write a program to print numbers from 1 to 50 and also reverse order**

**Program:**

i = 1

while i <= 50 do

print i

i += 1

end

puts " "

i = 50

while i > 0 do

print i

i -= 1

end

**19. Write a program using loop and iterator**

**Program:**

count = 750000

n1 = Time.now.usec

# Version 1: use times iterator.

v = 0

count.times do

v += 1

end

puts v

n2 = Time.now.usec

# Version 2: use for-loop.

v = 0

for i in 0..count-1

v += 1

end

puts v

n3 = Time.now.usec

# Compute millisecond timings.

puts ((n2 - n1) / 1000)

puts ((n3 - n2) / 1000)

**20. Write a program using iterator**

**Program:**

def addthree(max)

# Return a sequence incremented by three up to the max.

i = 0

while i <= max

yield i

i += 3

end

end

# Display sequence up to 20.

addthree(20) do |n|

puts n

end

**21. Write a program to find the length of a string**

**Program:**

#Lenth of a string

"dog park".length

**22.Write a program that checks if the sequence of characters**

**Program:**

def check\_in(word)

if /lab/ =~ word

puts word

else

puts "No match"

end

end

check\_in("laboratory")

check\_in("experiment")

check\_in("Pans Labyrinth")

check\_in("elaborate")

check\_in("polar bear")

**23. Program to loop through hash**

**Program:**

stones = { 1 => "garnet", 2 => "topaz",

3 => "opal", 4 => "amethyst"

}

stones.each { |k, v| puts "Key: #{k}, Value: #{v}" }

stones.each\_key { |key| puts "#{key}" }

stones.each\_value { |val| puts "#{val}" }

stones.each\_pair { |k, v| puts "Key: #{k}, Value: #{v}" }

Sample output:

Key: 1, Value: garnet

Key: 2, Value: topaz

Key: 3, Value: opal

Key: 4, Value: amethyst

1

2

3

4

garnet

topaz

opal

amethyst

Key: 1, Value: garnet

Key: 2, Value: topaz

Key: 3, Value: opal

Key: 4, Value: amethyst

**24. Program to find highest power**

**Program:**

# Calculate the highest power of 3 less than 10,000

number = 1

while number < 10\_000 do

number = number \* 3

end

# Now number is greater than 10,000, so

# we must divide by 3 to get back

# to the value we want.

number = number / 3

puts "#{number} is the highest power of 3 less than 10,000."

sample output:

6561 is the highest power of 3 less than 10,000.

**25. Write a program to update user input**

**Program:**

answer = "" # the empty string

while answer != "M" and answer != "F" do

print "What is your gender? M)ale or F)emale: "

answer = gets.chomp

end

if answer == "M"

puts "Hi, fella!"

else

puts "Hello, lady!"

end

**26. Write a program to loop the loop with loop**

**Program:**

x = 0

loop do

30.times {print "abc" }

break if x <=30

end

**27. Write a program to convert Celsius temperature to Fahrenheit**

**Program:**

class Celsius

def initialize(temperature)

@temperature = temperature

end

def to\_fahrenheit()

return(@temperature \* 1.8 + 32)

end

end

celsius = Celsius.new(10)

puts celsius.to\_fahrenheit()

**28. Write a program to define Calculator class with add(), substract(), multiply() and divide()**

**Program:**

class Calculator

def self.add(x, y)

return(x + y)

end

end

puts Calculator.add(3, 4)

**29. Create a class which should print some output**

**Program:**

class Person

def initialize(fname, lname)

@fname = fname

@lname = lname

end

def to\_s

"Person: #@fname #@lname"

end

end

person = Person.new("Augustus","Bondi")

print person

sample output:

Person: Augustus Bondi