Working with numbers

* What will be the result of followings, integer or float?
  + 3 / 2 :integer
  + 3.0 / 2 :float
  + 3 / 2.0 :float
  + 4 \* 2.0 :float
  + 0 + 1 :integer
* What will be the output of following?
  + 5 % 2 : 1
  + 15 % 2 : 1
  + 505 % 2 : 1
  + 8 % 5 : 3
  + 9 % 5 : 4
  + 10 % 5 : 0
  + 11 % 5 : 1
* How to find number is even or odd? puts X%2==0?“even”:”odd”
* Explain times method.

Ans: Times method works as a loop with specified iterations.

Ex: 3.times {puts “Abhinav”}

* Explain upto method.

Ans: upto method also works as a loop of given range

Ex: 1.upto(10) do |i|

puts i\*2

end

Working with Strings

* What will be the output of following for string s = “this is a test string”?
  + s << “: true”

Ans : “:true “ is appended to the given string

* + s << 33

Ans : It will take 33 as Ascii Code and will append “! “ to the string.

* + “hello” \* 3

Ans: string will be coverted to “hellohellohello”

* + “hello” \* 0

Ans: It will remove whole string (mul by zero) string of length 0

* + difference between single quoted string and double quoted string.

Ans: escape sequence does not work in single quoted string .

Ex: print ‘ a/nb’ #print a/nb as it is

print “a/nb” #print a in same line and b in next line

Working with Arrays

* write down various ways to insert value into an array.

Ans: a.push(3)

a << 3

a=1,2,3,4

a=(1..100).to\_a

* Convert array of strings to csv string.

Ans: string = “This is a test”

x= string.split(“ ”).to\_a

x={“this”,”is”,”a”,”test”}

* Explain various ways to define array.

Ans: a=[11,12,13]

a=[]

\*a=1,2,3

a=Array.new()

* Differentiate Array.new(3, Hash.new) and Array.new(3) { Hash.new }

Ans:

Array.new(3,Hash.new): In this all the values of a key will remain same

ex: a= Array.new(3,Hash.new)

a[0][‘cat’]=’black’

#then a=[{‘cat’=’black’},{‘cat’=’black’},{‘cat’=’black’}]

Array.new(2){Hash.new} : Different values of all hashes if defined else blank

ex: a=Array.new(2){Hash.new}

a[0][‘cat’]=’white’

#then a =[{‘cat’=’white’},{}]

a[1],[‘cat’]=’brown’

#then a=[{‘cat’=’white’},{‘cat’=’brown’}]

* Explain with examples
  + compact : return all element except nil

ex: a=[5,nil,2,nil,3]

puts a.compact #[5,2,3]

* + collect :does not change value of array only change in block and make copy

ex: a=[1,2,3,4]

puts a.collect {|x| x+2} #3,4,5,6

* + push : insert element into array

ex: a=[1,2,3]

a.push(4)

puts a #[1,2,3,4]

* + pop : delete element from array

ex: a=[1,2,3]

a.pop(2)

puts a #[1]

* + flatten : convert multidimensional to single dimension array unless a parameter is passed to which extent dimensionality is to be reduced.

ex: a=[1,2,3,[4,5]]

puts a.flatten #{1,2,3,4,5}

* + include? : return true or false if element is present in that class or/array

a=[‘a’,’b’,’c’]

puts a.include?(‘c’) #true

* + reject : reject those element which in self block are true

ex: a=[1,2,3,4,5,6,7,8]

puts a.reject {|x| x%2==0} #[1,3,5,7]

* + delete\_if : delete element if block is true

ex: a=[2,4,6,8,10,1]

puts a.delete\_if {|x| x%2==0} #[1]

* + shift : return element and removes it from array and shifting all elements by one

ex: a=[1,2,3,4,5,6]

puts a.shift(1) # 1

puts a #[2,3,4,5,6]

* + sort : sort elements in a group

ex: a=[9,4,6,23,4,1]

puts a.sort() #{1,4,4,6,9,23}

* + size : return size of array or hash

ex: a=[1,2,3,4]

puts a.size() #4

Working with Hashes

* How to know whether a key exists in the hash or not?

Ans: By using HASHNAME.key?(“key\_name”) method

* Generate a hash with values of length of > 4 from { a: ‘amit’, b: ‘vikram’, c: ‘Arun’, d: ‘Viraj’ }

Ans: my\_hash={}

my\_hash[:a]=’amit’

my\_hash[:b]=’vikram’

my\_hash[:c]=’Arun’

my\_hash[:d]=’Viraj’

* Code to delete keys with length > 4 from above hash

Ans :my\_hash[:hello]=’No need Delete it’

my\_hash.delete\_if(|x| x.length>4)

* Code to join two hashes.

Ans: my\_newhash=my\_hash.merge(old\_hash)

It overrides the new key value if present

* Is there any method to revert the hash key, values.

Ans: By using the invert() method

ex:

my\_hash={a:’navyug’,b:’info’,c:’solutions’}

puts my\_hash.invert() #{navyug:’a’,info:’b’,solutions:’c’}.

If two keys have same values the value 1 is discarded and second value is taken.