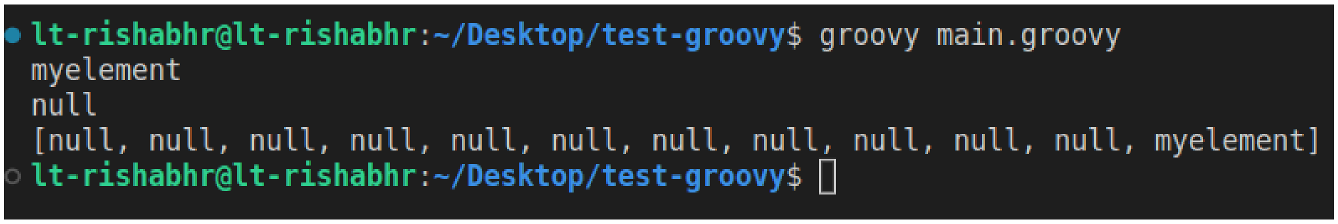
Rishabh Rajpurohit
Groovy Collections Assignment

A0.

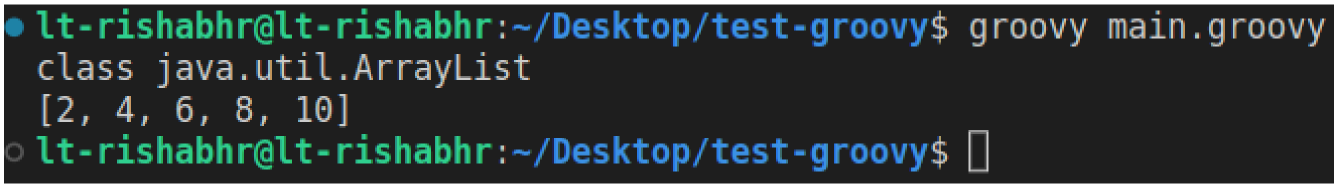
```
List l = []
l[11] = "rishabh"
println l[11]
println l.get(5)
println l
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
myelement
null
[null, null, null, null, null, null, null, null, null, null, null, myelement]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A1.

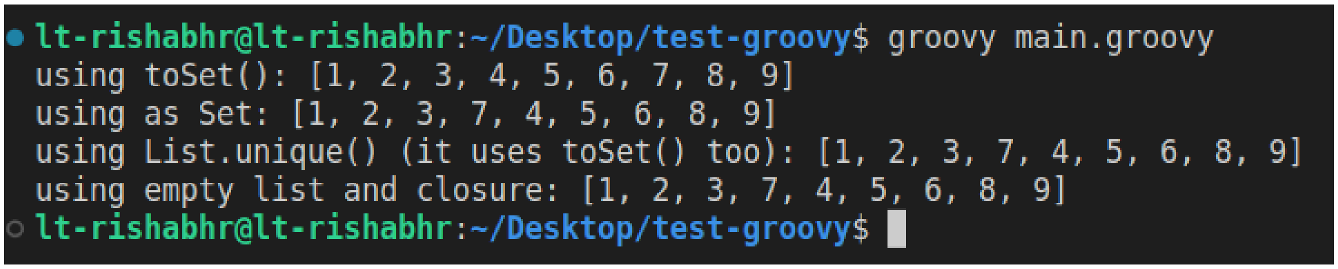
```
Range r = 1..10
def list = r.toList()
println list.getClass()
def evenList = list.findAll{item->~item%2}
println evenList
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
class java.util.ArrayList
[2, 4, 6, 8, 10]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

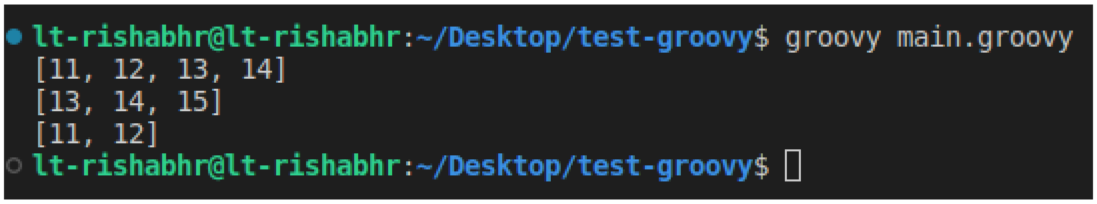
A2.

```
List l = [1,2,3,7,7,7,4,5,6,7,1,2,5,4,3,8,9]
Set s = l.toSet()
println "using toSet(): ${s}"
println "using as Set: ${l as Set}"
println "using List.unique() (it uses toSet() too): ${l.unique()}"

def seen = []
def uniqueList = l.findAll { !seen.contains(it) && seen.add(it) }
println "using empty list and closure: ${uniqueList}"
```



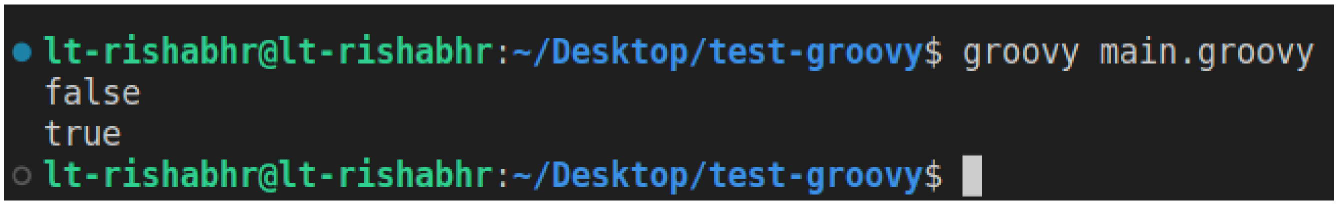
```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
using toSet(): [1, 2, 3, 4, 5, 6, 7, 8, 9]
using as Set: [1, 2, 3, 7, 4, 5, 6, 8, 9]
using List.unique() (it uses toSet() too): [1, 2, 3, 7, 4, 5, 6, 8, 9]
using empty list and closure: [1, 2, 3, 7, 4, 5, 6, 8, 9]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

```
A3.
List l1 = [11,12,13,14]
List l2 = [13,14,15]
List result = (l1+l1).unique() - l2
println l1
println l2
println result
```



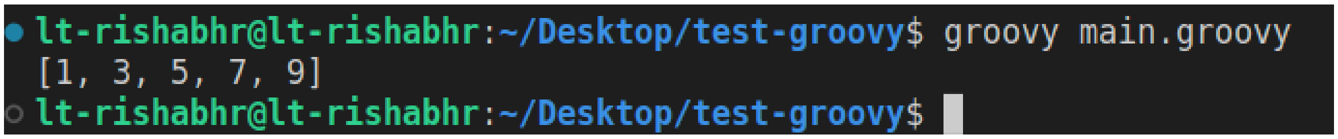
```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
[11, 12, 13, 14]
[13, 14, 15]
[11, 12]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

```
A4.
List l1 = [1,2,3,4,5]
List l2 = [7,8,9,4,10]
List l3 = [11,22,33,44,55]

println l1.disjoint(l2)
println l1.disjoint(l3)
```



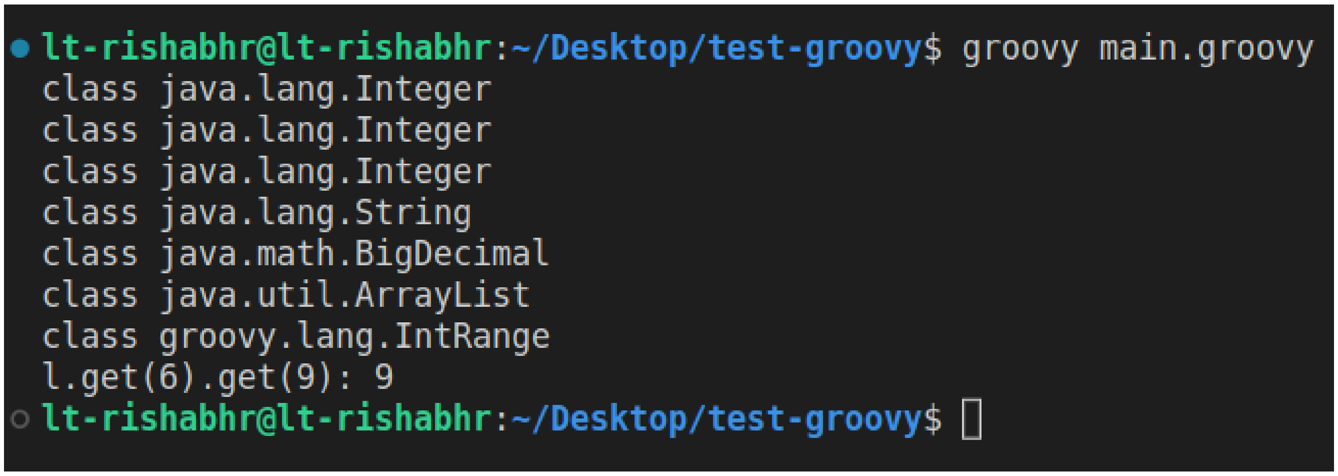
```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
false
true
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

```
A5.
def list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
def result = []
list.eachWithIndex { item, index ->
    if (index % 2 == 0) {
        result.add(item)
    }
}
println result
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
[1, 3, 5, 7, 9]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

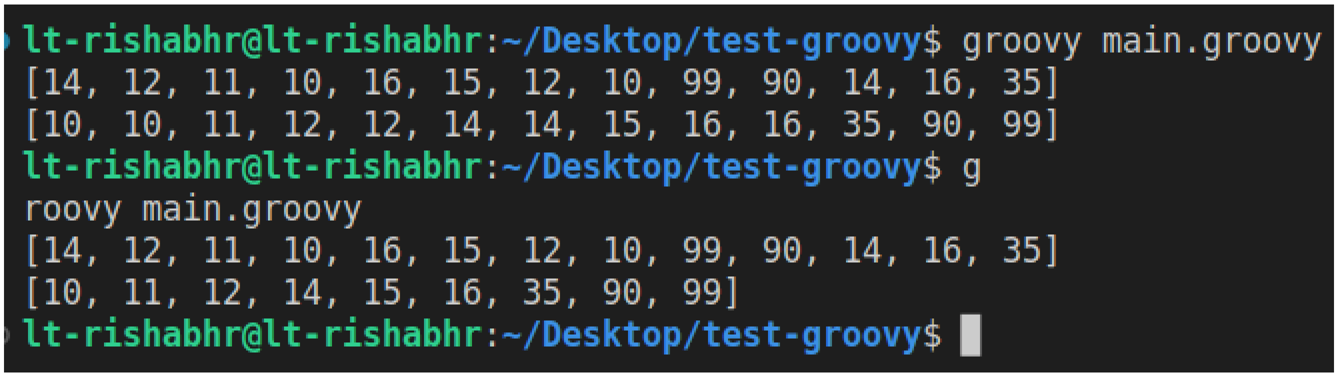
```
A6.
List l = [1,2,3,"element1",0.3,[2,4,6],0..10]
l.each{element->println element.getClass()}
println "l.get(6).get(9): ${l.get(6).get(9)}"
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
class java.lang.Integer
class java.lang.Integer
class java.lang.Integer
class java.lang.String
class java.math.BigDecimal
class java.util.ArrayList
class groovy.lang.IntRange
l.get(6).get(9): 9
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A7.

```
List l = [14,12,11,10,16,15,12,10,99,90,14,16,35]
println l
l.sort().unique()
println l
```

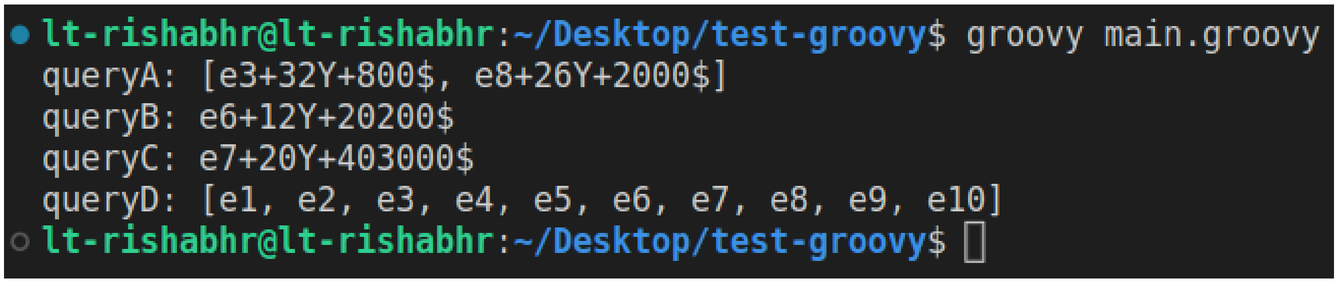


```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
[14, 12, 11, 10, 16, 15, 12, 10, 99, 90, 14, 16, 35]
[10, 10, 11, 12, 12, 14, 14, 15, 16, 16, 35, 90, 99]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ g
roovy main.groovy
[14, 12, 11, 10, 16, 15, 12, 10, 99, 90, 14, 16, 35]
[10, 11, 12, 14, 15, 16, 35, 90, 99]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A8.

```
class Emp {
    String name
    Integer age
    Integer salary
    public Emp(String name, Integer age, Integer salary){
        this.name = name
        this.age = age
        this.salary = salary
    }
    public String toString(){
        return "${this.name}+${this.age}Y+${this.salary}\$"
    }
}
```

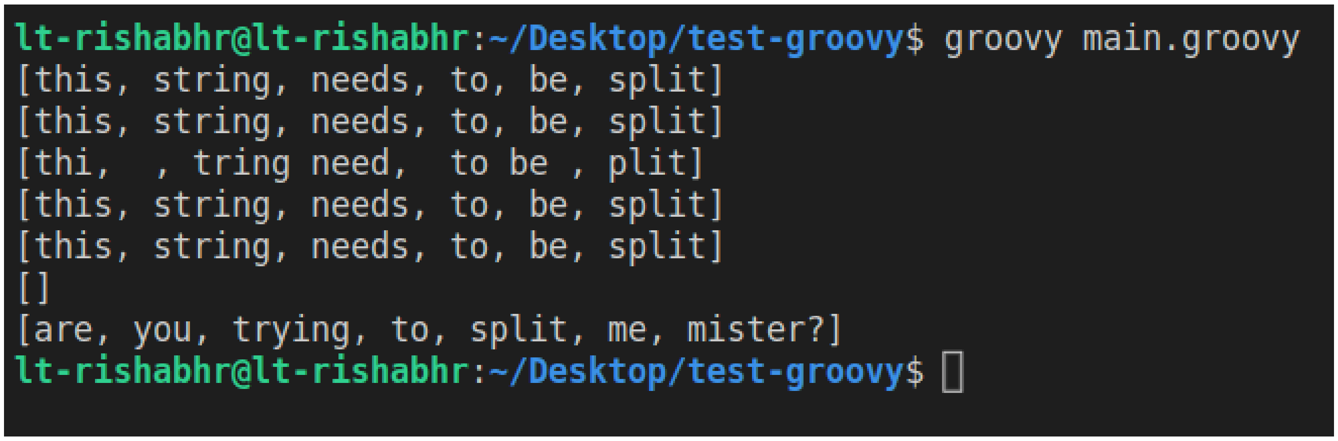
```
Emp e1 = new Emp("e1",28,20000)
Emp e2 = new Emp("e2",52,50000)
Emp e3 = new Emp("e3",32,800)
Emp e4 = new Emp("e4",25,200000)
Emp e5 = new Emp("e5",18,10000)
Emp e6 = new Emp("e6",12,20200)
Emp e7 = new Emp("e7",20,403000)
Emp e8 = new Emp("e8",26,2000)
Emp e9 = new Emp("e9",24,49000)
Emp e10 = new Emp("e10",22,93000)
List l = [e1,e2,e3,e4,e5,e6,e7,e8,e9,e10]

def queryA = l.findAll{it->it.salary<5000}
println "queryA: ${queryA}"

def queryB = l.min{it.age}
println "queryB: ${queryB}"

def queryC = l.max{it.salary}
println "queryC: ${queryC}"

def queryD = l.collect{it.name}
println "queryD: ${queryD}"
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
queryA: [e3+32Y+800$, e8+26Y+2000$]
queryB: e6+12Y+20200$
queryC: e7+20Y+403000$
queryD: [e1, e2, e3, e4, e5, e6, e7, e8, e9, e10]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A9.

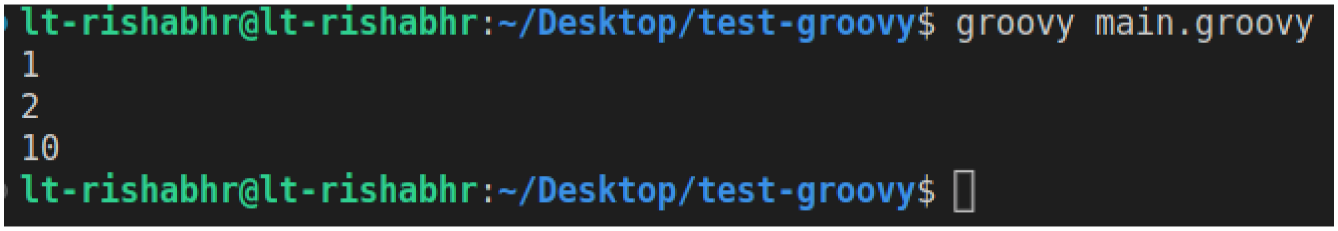
```
String s = "this string needs to be split"
println s.tokenize(" ")
println s.tokenize()
println s.tokenize(/\s/)
println s.split(" ")
println s.split(/\s/)
s = "are.you.trying.to.split.me.mister?"
println s.split(".")
println s.tokenize(".")
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
[this, string, needs, to, be, split]
[this, string, needs, to, be, split]
[thi,  , tring need,  to be , plit]
[this, string, needs, to, be, split]
[this, string, needs, to, be, split]
[]
[are, you, trying, to, split, me, mister?]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A10.

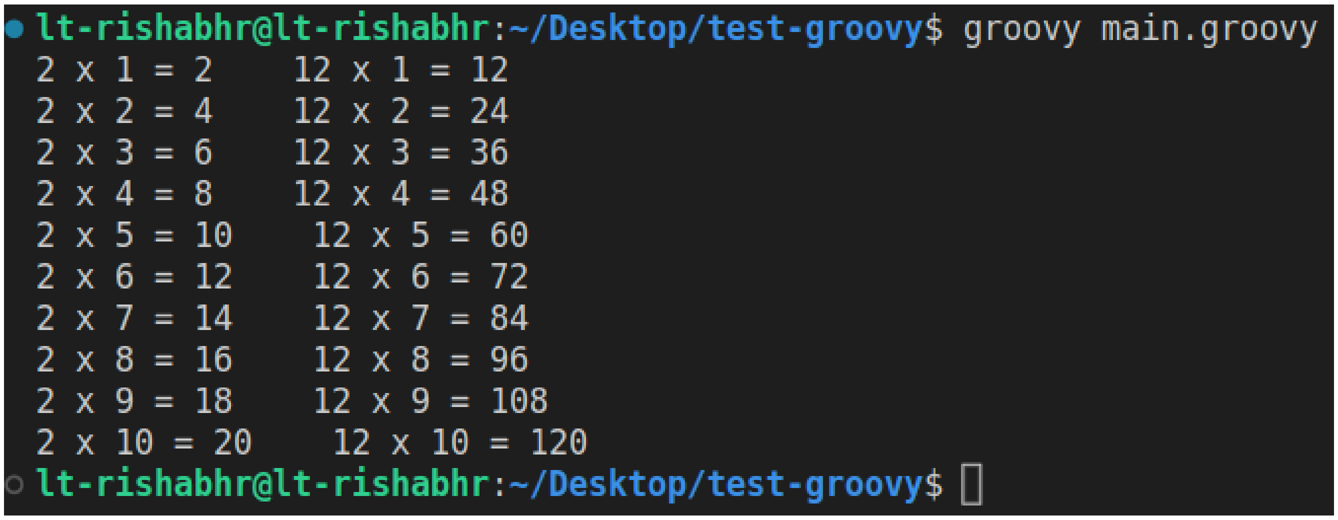
```
Range r = 1..10
println r.get(0)
println r.get(1)
println r.get(r.size()-1)
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
1
2
10
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

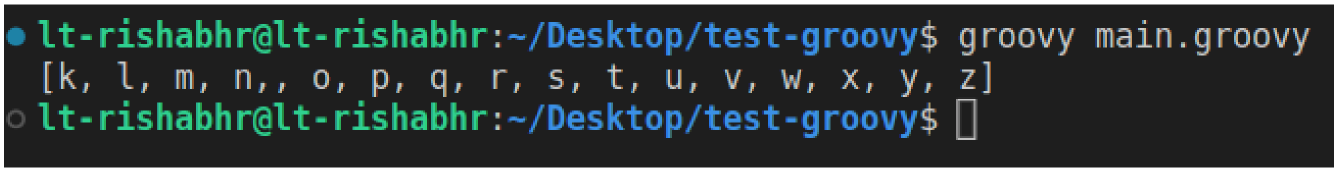
A11.

```
(1..10).each {
    println "2 x $it = ${2 * it}    12 x $it = ${12 * it}"
}
```



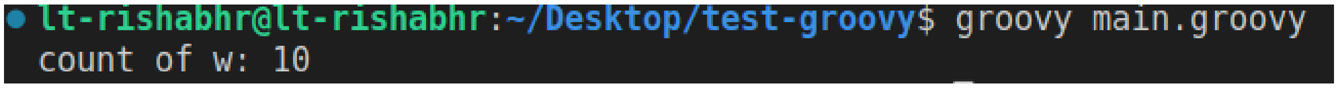
```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
2 x 1 = 2    12 x 1 = 12
2 x 2 = 4    12 x 2 = 24
2 x 3 = 6    12 x 3 = 36
2 x 4 = 8    12 x 4 = 48
2 x 5 = 10    12 x 5 = 60
2 x 6 = 12    12 x 6 = 72
2 x 7 = 14    12 x 7 = 84
2 x 8 = 16    12 x 8 = 96
2 x 9 = 18    12 x 9 = 108
2 x 10 = 20    12 x 10 = 120
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A12.
```
List l = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n,','o','p','q','r','s','t','u','v','w','x','y','z']
def listAfterJ = l.findAll{it->it>'j'}
println listAfterJ
```



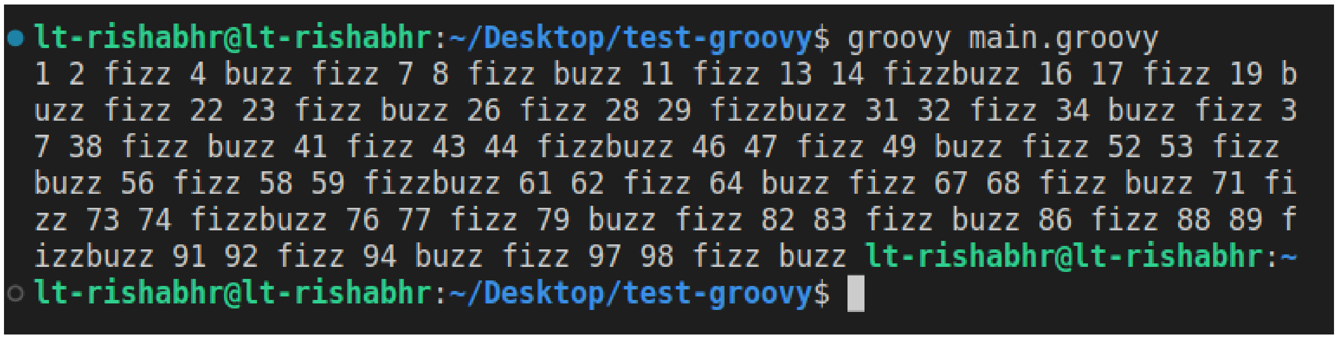
```
● lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
  [k, l, m, n,, o, p, q, r, s, t, u, v, w, x, y, z]
○ lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ 
```

A13.
```
String someString = "efbwefbwfiwfwfhwfiwninviwvowifowienfw"
def whatToFind = 'w'
def count = someString.findAll{it->it==whatToFind}.size()
println "count of ${whatToFind}: ${count}"
```



```
● lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
  count of w: 10
```

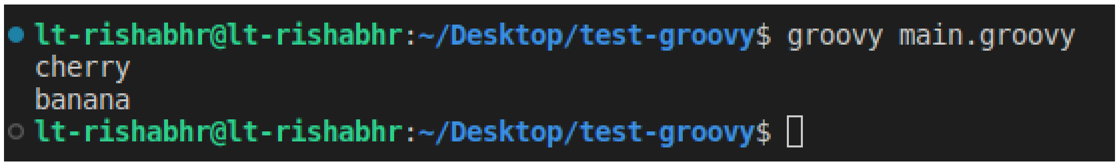
A14.
```
(1..100).each{it->
   if((it%3==0)&&(it%5==0)) print "fizzbuzz "
   else if(it%3==0) print "fizz "
   else if(it%5==0) print "buzz "
   else print "${it} "
}
```



```
● lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
  1 2 fizz 4 buzz fizz 7 8 fizz buzz 11 fizz 13 14 fizzbuzz 16 17 fizz 19 b
  uzz fizz 22 23 fizz buzz 26 fizz 28 29 fizzbuzz 31 32 fizz 34 buzz fizz 3
  7 38 fizz buzz 41 fizz 43 44 fizzbuzz 46 47 fizz 49 buzz fizz 52 53 fizz
  buzz 56 fizz 58 59 fizzbuzz 61 62 fizz 64 buzz fizz 67 68 fizz buzz 71 fi
  zz 73 74 fizzbuzz 76 77 fizz 79 buzz fizz 82 83 fizz buzz 86 fizz 88 89 f
  izzbuzz 91 92 fizz 94 buzz fizz 97 98 fizz buzz lt-rishabhr@lt-rishabhr:~
○ lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ 
```

A15.
```
class Stack {
   List<Object> stackList = []

   def push(Object item) {
      stackList.add(item)
   }
```

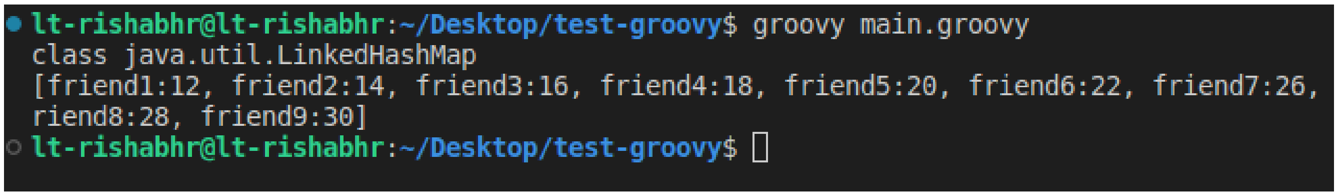
```
    def pop() {
        stackList.removeAt(stackList.size() - 1)
    }

    def top() {
        stackList[stackList.size() - 1]
    }
}

def myStack = new Stack()

myStack.push("apple")
myStack.push("banana")
myStack.push("cherry")

println myStack.top() // Output: cherry

myStack.pop()

println myStack.top() // Output: banana
```



```
● lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
  cherry
  banana
○ lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ 
```

A16.

```
def friends =
['friend1':12,'friend2':14,'friend3':16,'friend4':18,'friend5':20,'friend6':22,'friend7':24,'friend7':26,'friend
8':28,'friend9':30]
println friends.getClass()
println friends
```

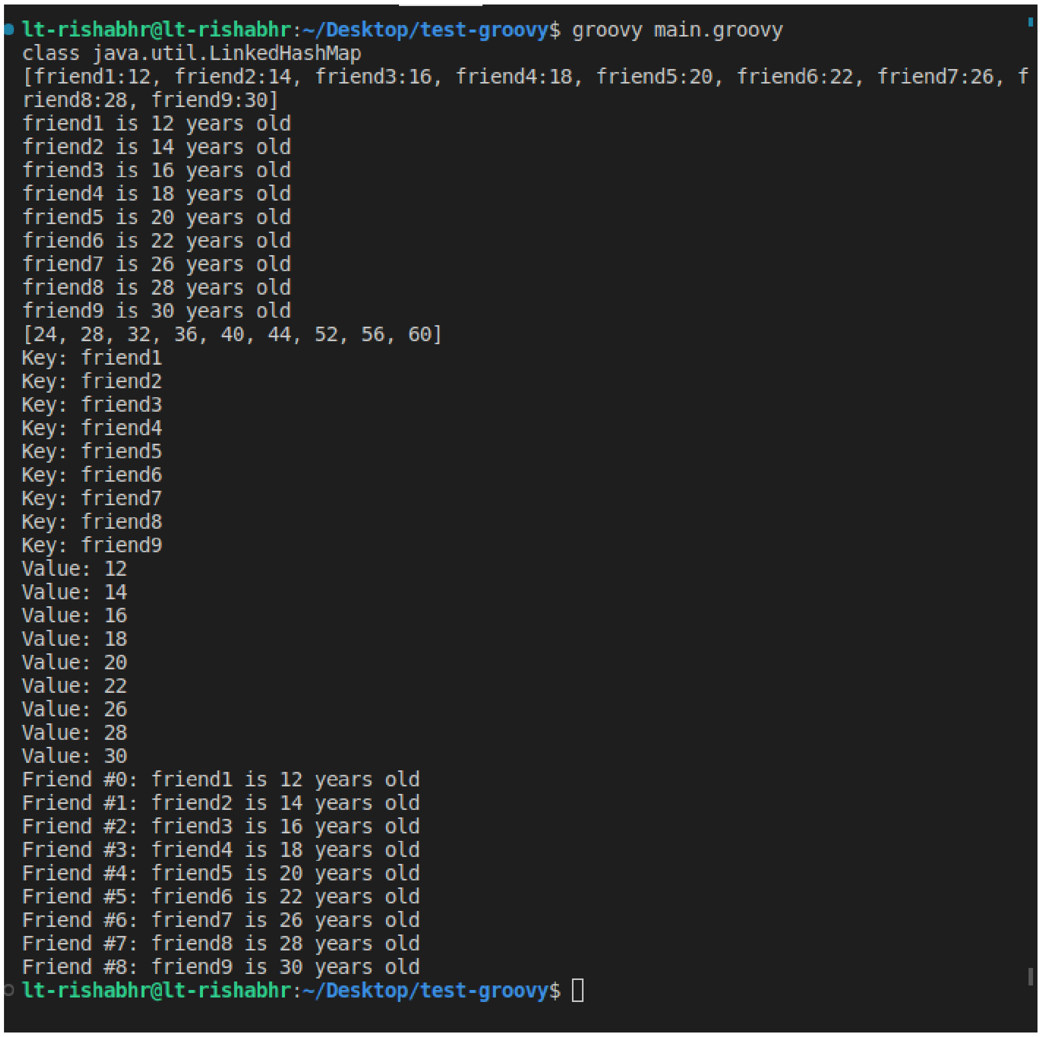


```
● lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
  class java.util.LinkedHashMap
  [friend1:12, friend2:14, friend3:16, friend4:18, friend5:20, friend6:22, friend7:26,
  riend8:28, friend9:30]
○ lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ 
```

A17.

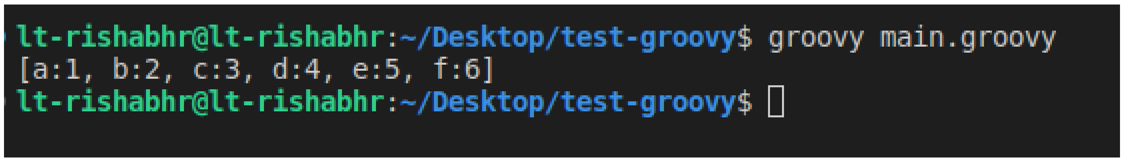
```
def friends =
['friend1':12,'friend2':14,'friend3':16,'friend4':18,'friend5':20,'friend6':22,'friend7':24,'friend7':26,'friend
8':28,'friend9':30]
println friends.getClass()
println friends
```

```
friends.each { key, value ->
    println "$key is $value years old"
}
def doubledAges = friends.collect { key, value ->
    value * 2
}
println doubledAges
for (key in friends.keySet()) {
    println "Key: $key"
}

for (value in friends.values()) {
    println "Value: $value"
}

friends.eachWithIndex { key, value, index ->
    println "Friend #$index: $key is $value years old"
}
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
class java.util.LinkedHashMap
[friend1:12, friend2:14, friend3:16, friend4:18, friend5:20, friend6:22, friend7:26, f
riend8:28, friend9:30]
friend1 is 12 years old
friend2 is 14 years old
friend3 is 16 years old
friend4 is 18 years old
friend5 is 20 years old
friend6 is 22 years old
friend7 is 26 years old
friend8 is 28 years old
friend9 is 30 years old
[24, 28, 32, 36, 40, 44, 52, 56, 60]
Key: friend1
Key: friend2
Key: friend3
Key: friend4
Key: friend5
Key: friend6
Key: friend7
Key: friend8
Key: friend9
Value: 12
Value: 14
Value: 16
Value: 18
Value: 20
Value: 22
Value: 26
Value: 28
Value: 30
Friend #0: friend1 is 12 years old
Friend #1: friend2 is 14 years old
Friend #2: friend3 is 16 years old
Friend #3: friend4 is 18 years old
Friend #4: friend5 is 20 years old
Friend #5: friend6 is 22 years old
Friend #6: friend7 is 26 years old
Friend #7: friend8 is 28 years old
Friend #8: friend9 is 30 years old
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

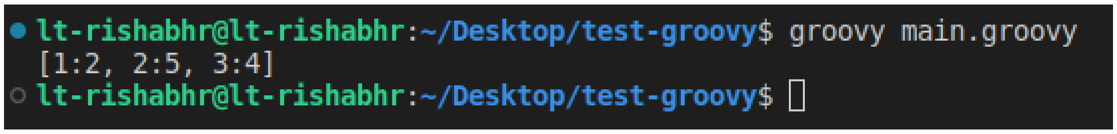
A18.
def map1 = [a: 1, b: 2, c: 3]
def map2 = [d: 4, e: 5, f: 6]
def combinedMap = map1 + map2
println combinedMap



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
[a:1, b:2, c:3, d:4, e:5, f:6]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

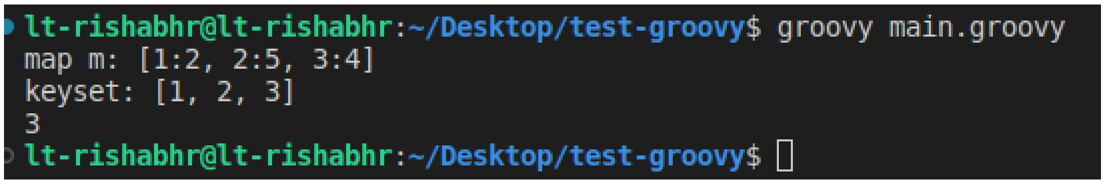
A19.
Map.class shows null
whereas map.getClass() returns Java.util.LinkedHashMap

A20.
This is valid but since map only stores unique keys the value of existing key gets updated to last entry.
def m = ['1':2,'2':3,'3':4,'2':5]
println m



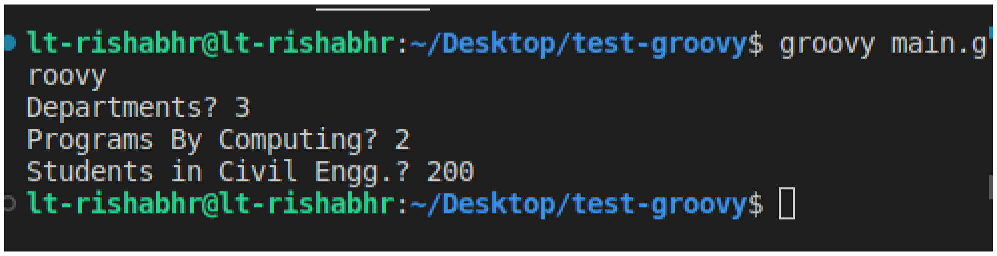
```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
[1:2, 2:5, 3:4]
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A21.
def m = ['1':2,'2':3,'3':4,'2':5]
println "map m: ${m}"
def mkeys = m.keySet()
println "keyset: ${mkeys}"
def result = mkeys.find{it->it=="3"}
println result



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
map m: [1:2, 2:5, 3:4]
keyset: [1, 2, 3]
3
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$
```

A22.
Map m = ['Computing':['Computing':600,'Information Systems':300],'Engineering':
['Civil':200,'Mechanical':100], 'Management':['Management':800]]
println "Departments? ${m.keySet().size()}"
println "Programs By Computing? ${m['Computing'].keySet().size()}"
println "Students in Civil Engg.? ${m['Engineering']['Civil']}"

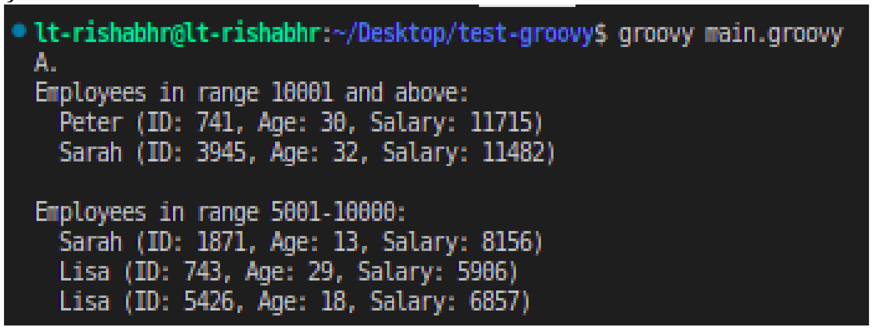


```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.g
roovy
Departments? 3
Programs By Computing? 2
Students in Civil Engg.? 200
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ 
```

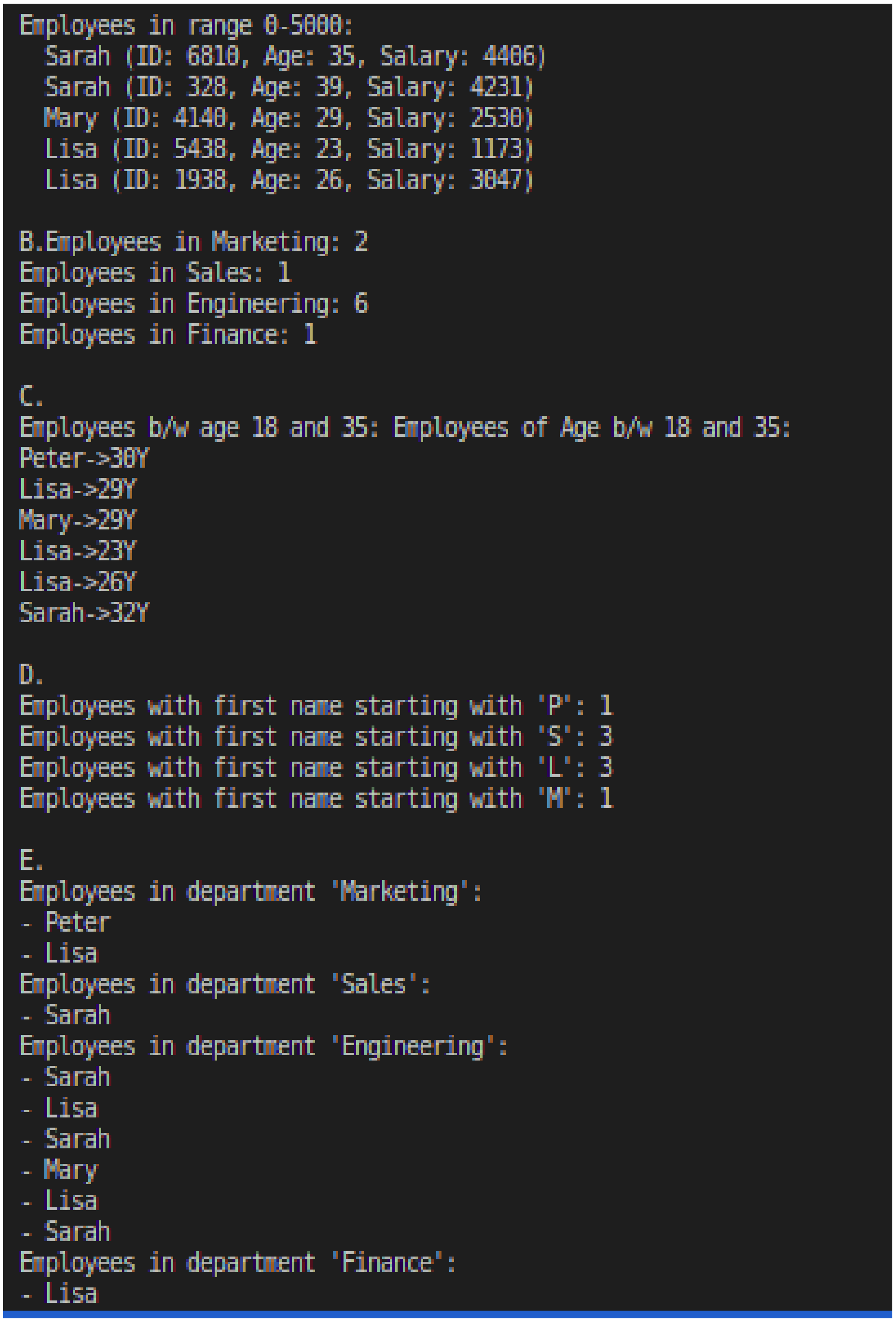
A23.

```
class Employee {
   String name
   int age
   String deptName
   int empID
   int salary

   static Employee createRandomEmployee() {
      String[] names = ["John", "Mary", "David", "Lisa", "Peter", "Sarah"]
      String[] deptNames = ["Sales", "Marketing", "Engineering", "Finance"]
      Random rand = new Random()

      Employee emp = new Employee()
      emp.name = names[rand.nextInt(names.length)]
      emp.age = rand.nextInt(30) + 10 // Random age between 20 and 59
      emp.deptName = deptNames[rand.nextInt(deptNames.length)]
      emp.empID = rand.nextInt(10000)
      emp.salary = rand.nextInt(12000) // Random salary between 50000 and 99999

      return emp
   }

   public String toString(){
      return "${this.name}->${this.age}Y"
   }
}

List<Employee> employees = (1..10).collect { Employee.createRandomEmployee() }

def SalaryRanges = [
   "0-5000":{it.salary<=5000},
   "5001-10000":{it.salary>5000 && it.salary <=10000},
   "10001 and above":{it.salary>10000}
]

def groupedEmployeesBySalary = employees.groupBy{emp->
SalaryRanges.findResult {
   range->range.value(emp)?range.key:null
}}
```

```
print "A."
groupedEmployeesBySalary.each { range, emps ->
   println "\nEmployees in range $range:"
   emps.each { emp ->
      println "  $emp.name (ID: ${emp.empID}, Age: ${emp.age}, Salary: ${emp.salary})"
   }
}
print "\nB."
def groupedEmployeesByDept = employees.groupBy{it.deptName}
groupedEmployeesByDept.each { key,value ->
   println "Employees in ${key}: ${value.size()}"
}
print "\nC.\nEmployees b/w age 18 and 35: "
def EmployeesBw18and35 = employees.findAll{
   it.age>18 && it.age<35
}
println "Employees of Age b/w 18 and 35: "
EmployeesBw18and35.each{emp->println emp}

println "\nD."
def groupedEmployeesByFirstNameLetter20Above = employees.groupBy { emp ->
emp.name[0] }.collectEntries { k, v -> [k, v.count { it.age > 20 }] }

groupedEmployeesByFirstNameLetter20Above.each { letter, count ->
   println "Employees with first name starting with '$letter': $count"
}

println "\nE."
def groupedEmployeesbyDepartment = employees.groupBy { emp -> emp.deptName }

groupedEmployeesbyDepartment.each { dept, empList ->
   println "Employees in department '$dept':"
   empList.each { emp ->
      println "- ${emp.name}"
   }
}
```



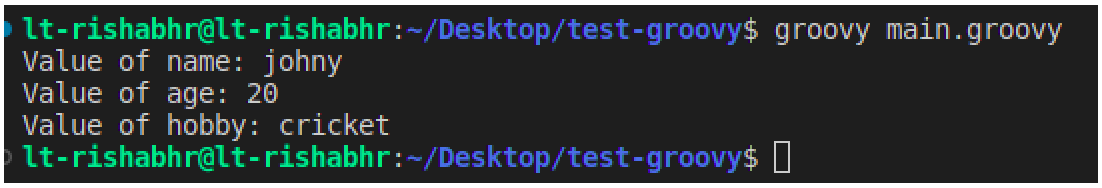
```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
A.
Employees in range 10001 and above:
  Peter (ID: 741, Age: 30, Salary: 11715)
  Sarah (ID: 3945, Age: 32, Salary: 11482)

Employees in range 5001-10000:
  Sarah (ID: 1871, Age: 13, Salary: 8156)
  Lisa (ID: 743, Age: 29, Salary: 5906)
  Lisa (ID: 5426, Age: 18, Salary: 6857)
```



```
Employees in range 0-5000:
  Sarah (ID: 6810, Age: 35, Salary: 4406)
  Sarah (ID: 328, Age: 39, Salary: 4231)
  Mary (ID: 4140, Age: 29, Salary: 2530)
  Lisa (ID: 5438, Age: 23, Salary: 1173)
  Lisa (ID: 1938, Age: 26, Salary: 3047)

B.Employees in Marketing: 2
Employees in Sales: 1
Employees in Engineering: 6
Employees in Finance: 1

C.
Employees b/w age 18 and 35: Employees of Age b/w 18 and 35:
Peter->30Y
Lisa->29Y
Mary->29Y
Lisa->23Y
Lisa->26Y
Sarah->32Y

D.
Employees with first name starting with 'P': 1
Employees with first name starting with 'S': 3
Employees with first name starting with 'L': 3
Employees with first name starting with 'M': 1

E.
Employees in department 'Marketing':
- Peter
- Lisa
Employees in department 'Sales':
- Sarah
Employees in department 'Engineering':
- Sarah
- Lisa
- Sarah
- Mary
- Lisa
- Sarah
Employees in department 'Finance':
- Lisa
```

A24.

```
def searchString = "https://www.google.com?name=johny&age=20&hobby=cricket"

Closure findKey = {String keyName->
    def key = keyName
    def startIndex = searchString.indexOf("${key}=")
    def value = ""
    if (startIndex != -1) {
        def endIndex = searchString.indexOf("&", startIndex)
        if (endIndex == -1) {
            endIndex = searchString.length()
        }
        value = searchString.substring(startIndex + key.length() + 1, endIndex)
    }
    println "Value of ${key}: ${value}"
}

findKey("name")
findKey("age")
findKey("hobby")
```



```
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ groovy main.groovy
Value of name: johny
Value of age: 20
Value of hobby: cricket
lt-rishabhr@lt-rishabhr:~/Desktop/test-groovy$ 
```