1. In the Scala REPL, type 3. followed by the Tab key. What methods can be applied?

% & \* + - / > >=

>> >>> ^ asInstanceOf isInstanceOf toByte toChar toDouble

toFloat toInt toLong toShort toString unary\_+ unary\_- unary\_~

|

1. In the Scala REPL, compute the square root of 3, and then square that value. By how much does the result differ from 3? (Hint: The res variables are your friend.)

s : Double = 1.7320508075688772

1. Are the res variables val or var?

Val coz res<number> holds the information till the end and cannot be assigned with a new value coz they are constant

1. Scala lets you multiply a string with a number— try out "crazy" \* 3 in the REPL. What does this operation do? Where can you find it in Scaladoc?

Crazycrazycrazy and the reverse doesn’t work in math multiply function

1. What does 10 max 2 mean? In which class is the max method defined?

10

1. Using BigInt, compute 2^1024.

Infinity

1. What do you need to import so that you can get a random prime as probablePrime(100, Random), without any qualifiers before probablePrime and Random?

**val** ss=scala.math.BigInt.probablePrime(100, scala.util.Random)

//> ss : scala.math.BigInt = 913290511074301403824724771987

1. One way to create random file or directory names is to produce a random BigInt and convert it to base 36, yielding a string such as "qsnvbevtomcj38o06kul". Poke around Scaladoc to find a way of doing this in Scala.
2. How do you get the first character of a string in Scala? The last character?

Val ss=”Hello”

ss.head

ss(0)

tail

**val** cs=sss(sss.length()-1) //> cs : Char = o

1. What do the take, drop, takeRight, and dropRight string functions do? What advantage or disadvantage do they have over using substring?

Take helps in return of elements in the start of string

Drop returns the elements left in the string after dropping

Takeright helps in return of elemts from the tail end

Dropright helps in return of the elemens from the start after dropping the sequence length

1. The signum of a number is 1 if the number is positive, –1 if it is negative, and 0 if it is zero. Write a function that computes this value.
2. **def** signum(x:BigInt) :BigInt={
3. **if**(x ==0) 0;
4. **else** **if** (x <0) -1;
5. **else** 1;
6. } //> signum: (x: scala.math.BigInt)scala.math.BigInt
7. signum(-4) //> res3: scala.math.BigInt = -1
8. What is the value of an empty block expression {}? What is its type?

Val sa={ } provides a null array

Come up with one situation where the assignment x = y = 1 is valid in Scala. (Hint: Pick a suitable type for x.)

1. Write a Scala equivalent for the Java loop for (int i = 10; i > = 0; i--) System.out.println( i);
2. **for** (i <- 10 to (0,-1))**yield**{i} //> res4: scala.collection.immutable.IndexedSeq[Int] = Vector(10, 9, 8, 7, 6, 5,
3. //| 4, 3, 2, 1, 0)
4. **for** (i<- 10 to 0 by -1) println(i) //> 10
5. //| 9
6. //| 8
7. //| 7
8. //| 6
9. //| 5
10. //| 4
11. //| 3
12. //| 2
13. //| 1
14. //| 0
15. Write a procedure countdown( n: Int) that prints the numbers from n to 0.

//| 4, 3, 2, 1, 0)

**def** countdown(n:BigInt)={

**for** (i<- n to 0 by -1) println(i)

} //> countdown: (n: scala.math.BigInt)Unit

countdown(5) //> 5

//| 4

//| 3

//| 2

//| 1

//| 0

1. }Write a for loop for computing the product of the Unicode codes of all letters in a string. For example, the product of the characters in "Hello" is 9415087488L.
2. **val** s2="string" //> s2 : String = string
3. s2.length //> res5: Int = 6
4. (**for**(i <- s2)**yield**(i.toInt)).product //> res6: Int = 990902384
5. Solve the preceding exercise without writing a loop. (Hint: Look at the StringOps Scaladoc.)
6. Write a function product(s : String) that computes the product, as described in the preceding exercises.
7. **def** product(x:String) :BigInt={
8. (**for**(i<-x)**yield**(i.toInt)).product
9. } //> product: (x: String)scala.math.BigInt
10. product("Hello") //> res7: scala.math.BigInt = 825152896
11. Make the function of the preceding exercise a recursive function.