Задание №1

def lol(a:List[Int]):Int = {  
 var i: Int = 0  
 var pop = a  
 while (pop.nonEmpty) {  
 pop = pop.tail  
 i += 1  
 }  
 a(i-1)  
}  
  
lol(*List*(32,24,5,7,8,9))

задание №2

def lol(a:List[Int]):Int = {  
 var i: Int = 0  
 var pop = a  
 while (pop.nonEmpty) {  
 pop = pop.tail  
 i += 1  
 }  
 i  
}  
  
lol(*List*(32,24,5,7,8,9))

Задание 3

def lol(a:List[Int]):Boolean = {  
 var i: Int = 0  
 var pop = a  
 while (pop.nonEmpty) {  
 pop = pop.tail  
 i += 1  
 }  
 i -= 1  
 var LastValue = a(i)  
 if(LastValue==a.head) {  
 var b:List[Int] = a.patch(i, *Nil*, 1)  
 b=b.patch(0, *Nil*, 1)  
 if(b.nonEmpty) {  
 lol(b)  
 1==1  
 }  
 else {2==1}  
 }  
 else {2==1}  
}  
  
lol(*List*(1, 2, 3, 2, 1))

Задание №4

def lol(a:List[Symbol],n:Int):(List[Symbol],Symbol) = {  
 var i: Int = 0  
 var pop = a  
 var b:List[Symbol] = *List*()  
 while(pop.nonEmpty) {  
 pop = pop.tail  
 if (i != n) {  
 b = b :+ a(i)  
 i+= 1  
 }  
 else {  
 i+= 1  
 }  
 }  
 (b,a(n))  
}  
  
lol(*List*('a, 'b, 'c, 'd),1)

Задание №5

import scala.util.Random  
  
def lol(a:List[Symbol],n:Int):List[Symbol] = {  
 var num: Int = 0  
 var i: Int = 1  
 var pop = a  
 var b:List[Symbol] = *List*()  
   
 while (pop.nonEmpty) {  
 pop = pop.tail  
 num += 1  
 }  
 num -= 1  
 while(i<=n) {  
 *print*(num)  
 var r = Random.between(0, num)  
 b = b :+ a(r)  
 i+= 1  
 }  
 b  
}  
  
lol(*List*('a, 'b, 'c, 'd),4)

Задача №6

def isPrime2(i :Int) : Boolean = {  
 var b:Boolean = false  
 if(i==1 || i == 2){  
 return true  
 }  
 for (x<-2 until i ) {  
 if(i%x==0) {  
 b= 2==1  
 return false  
  
 }  
 else  
 b = true  
 }  
  
 b  
 }  
isPrime2(5)

Задание №7

import scala.math.{abs}  
  
def find(x: Double, eps: Double) = {  
 var f = .0  
 var df = .0  
 var iter = 0  
 var result:Double = 0  
 *print*("x0="+ x + "\n")  
 do {  
 f = (x\*x\*x) + 18\*x-83  
 df = 3\*(x\*x) + 18  
 result = x - f / df  
 iter += 1  
 } while ( {  
 *abs*(f) > eps && iter < 2000000  
 })  
 *print*("Final: "+ iter + " "+ "iterations" + " " )  
 *print*("Result: "+result)  
}  
  
  
find(2.9, 0.04405)

import scala.annotation.tailrec  
import scala.math.abs  
import scala.util.Random  
  
// задание 1  
@tailrec  
def last(x: List[Int]): Int = {  
 if (x.tail.nonEmpty)  
 last(x.tail)  
 else  
 x.head  
}

val l1 = *List*(1, 1, 2, 3, 5, 8)

*print*(last(l1))  
  
// задание 2  
@tailrec  
def length(x: List[Int], i: Int = 0): Int = {  
 if (x.nonEmpty)  
 length(x.tail, i + 1)  
 else  
 i  
}

val l1 = *List*(1, 1, 2, 3, 5, 8)

*print*(length(l1))  
  
// задание 3  
@tailrec  
def isPalindrome(x: List[Int], i: Int = 0): Boolean = {  
 //узнаем длину списка  
 var len = 0  
 var a = x  
 while (a.nonEmpty) {  
 a = a.tail  
 len += 1  
 }  
 //сравниваем элементы  
 if (i <= len / 2)  
 if (x(i) != x(len - i - 1))  
 false  
 else  
 isPalindrome(x, i + 1)  
 else  
 true  
}

val l2 = *List*(1, 2, 3, 2, 1)

*print*(isPalindrome(l2))  
  
//задание 4  
@tailrec  
def removeAt(n: Int, x: List[Char], b: List[Char] = *List*(), i:Int=0): (List[Char], Char) = {  
 //узнаем длину списка  
 var len = 0  
 var s = x  
 while (s.nonEmpty) {  
 s = s.tail  
 len += 1  
 }  
 //удаляем элемент  
 if (i < len)  
 if (i != n)  
 removeAt(n, x, b:+x(i), i+1)  
 else  
 removeAt(n, x, b, i+1)  
 else  
 (b, x(n))  
}

val l3 = *List*('a', 'b', 'c', 'd')

*print*(removeAt(1, l3))

//задание 5  
@tailrec  
def randomSelect(n:Int, x:List[Char], b:List[Char]=*List*(), i:Int=0):List[Char] = {  
 //узнаем длину списка  
 var len = 0  
 var a = x  
 while (a.nonEmpty) {  
 a = a.tail  
 len += 1  
 }  
 //берем случайные элементы  
 if (i<n)  
 randomSelect(n, x, b:+x(Random.between(0,len)), i+1)  
 else  
 b  
  
}

val l4 = *List*('a', 'b', 'c', 'd', 'f', 'g', 'h')

*print*(randomSelect(3, l4))

//задание 6  
@tailrec  
def isPrime(x :Int, a:Int=2) : Boolean = {  
 if(x == 1 || x == 2)  
 return true  
 if (a < x - 1)  
 if(x % a == 0)  
 false  
 else  
 isPrime(x,a+1)  
 else  
 true  
}

*print*(isPrime(7))  
  
//задание 7  
@tailrec  
def Newton(x: Double, eps: Double, iter:Int=0): Double = {  
 var f = .0  
 var df = .0  
 var result = .0  
 if (iter < 20000) {  
 f = (x\*x\*x) + 18\*x-83  
 df = 3\*(x\*x) + 18  
 result = x - f / df  
 if (*abs*(f) > eps) {  
 result  
 } else {  
 Newton(x, eps, iter+1)  
 }  
 } else {  
 result  
 }  
}

*print*(Newton(83, 0.00001))