Tuan

**Question 1:**

a)

object Square{

/\*\* Calcualtes the remainder on dividing an integer by 3

\* Post: returns n \* n \*/

def square(n : Int) : Int = {

n \* n

}

def main(args : Array[String]) = {

val s = args.size

if(s == 1){

val input = args(0).toInt

println(square(input))

}

else{

println("Wrong number of arguments")

}

}

}

b)

object Remainder{

/\*\* Calculates the remainder on dividing an integer by 3

\* Pre: n >= 0

\* Post: returns n % 3 \*/

def remainder(n : Int) : Int = {

require(n >= 0)

var m : Int = n

while(m - 3 >= 0){

m -= 3

}

m

}

def main(args : Array[String]) = {

val s = args.size

if(s == 1){

val input = args(0).toInt

println(remainder(input))

}

else{

println("Wrong number of arguments")

}

}

}

c)

object BiggestSquare{

/\*\* Calculates the largest perfect square smaller than the input

\* Pre: n >= 0

\* Post: returns the largest square smaller than the input \*/

def biggestSquare(n : Int) : Int = {

require(n >= 0)

var i : Int = 0

while((i + 1) \* (i + 1) <= n){

i = i + 1

}

i \* i

}

def main(args : Array[String]) = {

val n = args.size

if(n == 1){

val input : Int = args(0).toInt

if(input >= 0){

println(biggestSquare(input))

}

else println("Please input a non-negative number")

}

else{

println("Wrong number of arguments")

}

}

}

**Question 2:**

object Milk{

/\*\* Calculate sum of a

\* Post: returns sum(a) \*/

def findSum(a : Array[Int]) : Int = {

val n = a.size

var total = 0; var i = n - 1

// Invariant I: total = sum(a[i..n)) && 0<=i<n

// Variant i

while(i >= 0){

// I && i>=0

total += a(i)

// total = sum(a[i+1..n)) && i>=0

i -= 1

// I

}

// I && i=0

// total = sum(a[0..n))

total

}

def main(args : Array[String]) = {

val a = args.map(x => x.toInt)

println(findSum(a))

}

}

**Question 3:**

object Milk{

/\*\* Calculate max of a

\* Post: returns max(a) \*/

def findMax(a : Array[Int]) : Int = {

val n = a.size

var max = 0; var i = 0

// Invariant I: total = max(a[0..i)) && 0<=i<=n

// Variant i

while(i < n){

// I && i < n

if(max < a(i)) max = a(i)

// max = max(a[0..i+1)) && i < n

i += 1

// I

}

// I && i=n

// max = max(a[0..n))

max

}

def main(args : Array[String]) = {

val a = args.map(x => x.toInt)

println(findMax(a))

}

}

**Question 4:**

object Milk{

/\*\* Calculate sum of a

\* Post: returns sum(a) \*/

def findSum(array : Array[Int]) : Int = {

val n = array.size

var total = 0; var i = 0

// Invariant I: total = sum(a[0..i)) && 0<=i<n

// Variant n-i

while(i < n){

// I && i < n

total += array(i)

// total = sum(array[0..i+1)) && i < n

i += 1

// I

}

// I && i = n

// total = sum(array[0..n))

total

}

def main(args : Array[String]) = {

val array = args.map(x => x.toInt)

println(findSum(array))

}

}

**Question 5:**

a)

object Fibonacci{

/\*\* Calculates the n-th Fibonacci number

\* Pre: n >= 0

\* Post: returns the n-th Fibonacci number \*/

def fib(n : Int) : Int = {

if(n == 0) 0

else{

if(n == 1) 1 else (fib(n-1) + fib(n-2))

}

}

def main(args : Array[String]) = {

val n = args.size

if(n == 1){

val input = args(0).toInt

if(input >= 0){

println(fib(input))

}

else println("Please input a non-negative number")

}

else println("Wrong number of arguments")

}

}

b)

object FibonacciTree{

/\*\* Calculates the n-th Fibonacci number

\* Pre: n >= 0

\* Post: returns the n-th Fibonacci number \*/

def fib(n : Int) : Int = {

if(n == 0) 0

else{

if(n == 1) 1 else (fib(n-1) + fib(n-2))

}

}

/\*\* Prints the '|' symbol with spaces between it

\* Pre: n >= 0 \*/

def printSymbol(n : Int) : Unit = {

var iter = n

while(iter > 0){

print("| ")

iter -= 1

}

}

def printTree(n : Int, d : Int) : Unit = {

if(n == 0){

printSymbol(d)

println("fib(0)")

printSymbol(d)

println("= 0")

}

else{

if(n == 1){

printSymbol(d)

println("fib(1)")

printSymbol(d)

println("= 1")

}

else{

printSymbol(d)

println("fib(" + n + ")")

printTree(n-1, d+1)

printTree(n-2, d+1)

printSymbol(d)

println("= " + fib(n))

}

}

}

def main(args : Array[String]) = {

val n = args.size

if(n == 1){

val input = args(0).toInt

if(input >= 0){

printTree(input, 0)

}

else println("Please input a non-negative number")

}

else println("Wrong number of arguments")

}

}

The program cannot be written without the depth parameter because it needs to keep track how many ‘|’ to print.

**Question 6:**

object Fibonacci{

/\*\* Calculates the n-th Fibonacci number

\* Pre: n >= 0

\* Post: returns the n-th Fibonacci number \*/

def fib(n : Int) : Int = {

var a = new Array[Int](n+1)

a(0) = 0

if(n > 0){

a(1) = 1

}

var i = 2

while(i <= n){

// Invariant I : a(i) = a(i-1) + a(i-2) && 0 <= i <= n+1

a(i) = a(i-1) + a(i-2)

i += 1

}

// I and i = n+1, so a(n) = a(n-1) + a(n-2)

return a(n)

}

def main(args : Array[String]) = {

val n = args.size

if(n == 1){

val input = args(0).toInt

if(input >= 0){

println(fib(input))

}

else println("Please input a non-negative number")

}

else println("Wrong number of arguments")

}

}

**Question 7:**

object DivMod{

/\*\* Calculates the divMod of a number

\* Pre: x >= 0, y >= 0

\* Post: returns the divMod of a number \*/

def divMod(x : Int, y : Int) : (Int, Int) = {

//Invariant I: x = q \* y + z && 0 <= z <= x

var q = 0; var z = x

while(z > y){

// I

z -= y

q += 1

// I

}

// I && 0 <= z < y, so q = a `div` b, z = a `mod` b

return (q, z)

}

def main(args : Array[String]) = {

val n = args.size

if(n == 2){

val x = args(0).toInt; val y = args(1).toInt

if(x >= 0 && y >= 0){

println(divMod(x, y))

}

else println("Please enter non-negative numbers")

}

else println("Wrong number of arguments")

}

}

**Question 8:**

a)

object GCD{

/\*\* Calculate the greates common devisor of m and n

\* Pre: m >= 0, n >= 0 \*/

def gcd(m : Int, n : Int) : Int = {

var a = m; var b = n; var r = a; var q = 0

//Invariant I : a = qb + r && 0 <= r < b

while(r != 0){

//I

q = a / b

r = a - q \* b

if(r != 0){

a = b

b = r

}

}

// I and r = 0, so a = qb

b

}

def main(args : Array[String]) = {

val s = args.size

if(s == 2){

val m = args(0).toInt; val n = args(1).toInt

if(m >= 0 && n >= 0){

println(gcd(m, n))

}

else println("Please enter non-negative integers")

}

else println("Wrong number or arguments")

}

}

b)

object gcd{

/\*\* Find x and y such that a = mx + ny

\* Pre: m >= 0, n >= 0 \*/

def gcd(m : Int, n : Int) : (Int, Int, Int) = {

var s = 0; var s1 = 1

var t = 1; var t1 = 0

var r = m; var r1 = n

//Invariant I : r1 = m \* t1 + n \* s1 and 0 <= r <= m

while(r != 0){

//I

var q = r1 / r

var swap = r

r = r1 - q \* swap

r1 = swap

swap = s

s = s1 - q \* swap

s1 = swap

swap = t

t = t1 - q \* swap

t1 = swap

}

//I and r = 0, so r1 = gcd(m, n), therefore r1 = m \* t1 + n \* s1

(t1, s1, r1)

}

def main(args : Array[String]) = {

val s = args.size

if(s == 2){

val m = args(0).toInt; val n = args(1).toInt

if(m >= 0 && n >= 0){

println(gcd(m, n))

}

else println("Please enter non-negative integers")

}

else println("Wrong number or arguments")

}

}

**Question 9:**

object Hit{

/\*\* Calculate the number of hits in an array

\* Pre: n >= 0 \*/

def hit(a : Array[Int]) : Int = {

var number = 0; var currentHit = a(0); var j = 1 // The first element cannot be a hit

var n = a.size

//Invariant I : a(j) is a hit,

// whenever it is larger than the previous hit, 0 <= j < n,

// currentHit = max(a[0..j]), 0 <= j < n

while(j < n){

//I

if(a(j) > currentHit){

number += 1

currentHit = a(j)

}

j += 1

}

//I and j = n,

//so number contains the number of all hits in a[0..n)

number

}

def main(args : Array[String]) = {

val n = args.size

var array = new Array[Int](n)

array = args.map(\_.toInt)

println(hit(array))

}

}