Fundamentals of Parallel Systems

Lab Book

**C2**

**Consumer.groovy:**

//insert a modified println statement

*println* "Your input multiplied by the factor is: " + i

**CreateSetsOfEight.groovy:**

// put v into outList and read next input

outList.add(v)

**Multiplier.groovy:**

// write i \* factor to outChannel

outChannel.write(i \* factor)

// read in the next value of i

i = inChannel.read()

Run Multiplier.groovy:

//insert here an instance of multiplier with a multiplication factor of 4

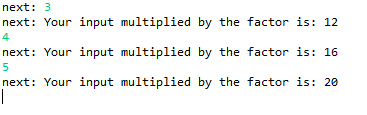
**new** Multiplier ( inChannel: connect1.**in**(),

outChannel: connect2.out(), factor:4 ),

**Diagram:**



**Working Screenshot:**



**ListToStream.groovy:**

// hint: output list elements as single integers

**for**(**int** i=0; i<3; i++)

{

outChannel.write(inList[i])

}

inList = inChannel.read()

**GenerateSetsOfThree.groovy:**

//write the terminating List as per exercise definition

outChannel.write([-1, -1, -1])

**CreateSetsOfEight.groovy:**

// put v into outList and read next input

outList[i] = v

v = inChannel.read()

**Questions:**

What change is required to output objects containing six integers?

Change the for loop in ‘CreateSetsOfEight.groovy’ from 0..7 to 0..5

How could you parameterise this in the system to output objects that contain any number of integers (e.g. 2, 4, 8, 12)?

###############################################################

What happens if the number of integers required in the output stream is not a factor of the total number of integers in the input stream (e.g. 5 or 7) ?

Any remainders are put into a list, but the list is not printed until it becomes full

**C3**

**Differentiate.groovy:**

// insert a constructor for Minus

**new** Minus ( inChannel0: a.in(),

inChannel1: c.in(),

outChannel: outChannel)

**Minus.groovy:**

// output one value subtracted from the other

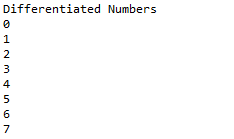
// be certain you know which way round you are doing the subtraction!!

outChannel.write(read0.value - read1.value)

**Diagram:**



**Working Screenshot:**



**DifferentiateNeg.groovy:**

//insert a constructor for Negator

**new** Negator ( inChannel: c.in(),

outChannel: d.out()),

**Negator.groovy:**

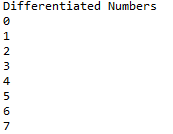
//output the negative of the input value

outChannel.write(inChannel.read() \* -1)

**Diagram:**



**Working Screenshot:**

****

**Questions:**

Which is the more pleasing solution? Why?

The minus approach is more pleasing as it is simpler to follow and requires less steps, whereas using the negator creates the need for another process, and as a result of that things become a little more complex. However, this could simplify problems that are larger and more complex overall, but in terms of this task the simpler solution is better.