*Logic Specification Template*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Roberto Cantu Reyes | **Program #** | 2 |

|  |  |
| --- | --- |
| **Class Name** | PrintResults |

|  |  |
| --- | --- |
| **Method Name** | print |

|  |  |
| --- | --- |
| **Parameters** | none |
| **Pseudocode:** |

print x with 5 decimals

print dof

print p with 5 decimals

|  |  |
| --- | --- |
| **Class Name** | Main |

|  |  |
| --- | --- |
| **Method Name** | main |

|  |  |
| --- | --- |
| **Parameters** | nonw |
| **Pseudocode:** |

Create StatisticalAnalysis object

Create PrintResults object

Ask user for x and dof

if x or dof are not valid input

Endprogram

Call simponsRule method from StaticalAnalysisobject and store results in a double variable

Call print method from PrintResults object and print results in console.

End program

|  |  |
| --- | --- |
| **Class Name** | StatisticalAnalysis |

|  |  |
| --- | --- |
| **Method Name** | gammaFunction |

|  |  |
| --- | --- |
| **Parameters** | Double num |
| **Pseudocode:** |

if (num == 1)

return 1;

}

else if(num == 0.5){

return sqrt of pi;

}

else {

return (num - 1) \* gammaFunction(num - 1);

}

|  |  |
| --- | --- |
| **Class Name** | StatisticalAnalysis |

|  |  |
| --- | --- |
| **Method Name** | tDistributionFunction |

|  |  |
| --- | --- |
| **Parameters** | Double x , int dof |
| **Pseudocode:** |

[gamma((dof + 1) / 2) / [(dof \* pi) ^0.5 \* GammaFunction(dof / 2)]] \* (1 + (x^2) / dof) ^ -(dof + 1)/ 2

|  |  |
| --- | --- |
| **Class Name** | StatisticalAnalysis |

|  |  |
| --- | --- |
| **Method Name** | simpsonsRule |

|  |  |
| --- | --- |
| **Parameters** | Double x , int dof |
| **Pseudocode:** |

**int I = 1**

**double w = x / numseg**

**int sumImpair = 0**

**int sumPair = 0**

**int tempResult = 0**

**for (I ; I < numSeg; I ++)**

**if I % 2 == 0 then**

**sumPair += tDistributionFunction(dof, I \* W)**

**else**

**sumImpair += tDistributionFunction(dof, I \* w)**

**endofFor**

**tempResult = TDistribution(dof,0) + 4\*sumImpair + 2 \* sumPair + tDistributionFunction(dof,(i)\*w)) \* W / 3**

**//Recursion**

**if abs oldResult – newResult < acceptableError**

**return newResult**

**elseif newResult > 0 and abs oldResult – newResult > acceptableError**

**oldResult = newResult**

**newResult = tempResult**

**numSeg \*= 2**

**return simponsRule(x,dof)**

**if oldResult == 0 and newResult == 0 then**

**oldResult = tempResult;**

**newSeg\*=2**

**return simponsRule(x,dof)**

**if oldResult > 0 and newResult == 0 then**

**newResult = tempResult**

**newSeg \*=2**

**return simpsonsRule(x,dof)**

**Include the image of the activity diagram created for the program:**