*Logic Specification Template*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Ivan Escalante | **Program #** | 7 |

|  |  |
| --- | --- |
| **Class Name** | T\_dist.h |

|  |  |
| --- | --- |
| **Method Name** | Double gammaFunc |

|  |  |
| --- | --- |
| **Parameters** | Double alfa |

|  |
| --- |
| If alfa == 1 -> return 1 |
| If alfa == .5 return power of pi^.5 |
| Else return (alfa-1) \* recursive(alfa-1) |
|  |
|  |

|  |  |
| --- | --- |
| **Method Name** | Double evalDistT |

|  |  |
| --- | --- |
| **Parameters** | Double x |
|  | Int dof |

|  |
| --- |
| Crea una nueva variable llamada expn |
| Expn = funcionGamma( (dof+1)/2); //aquí estarás llamando a la funcion de gamma arriba, con Alpha |
| expn = expn/((pow((dof\*PI),0.5))\*gammaFunc(dof/2.0)); |
| expn = expn\*pow((1+(pow(x,2)/dof)),((dof+1)/-2.0)); |
|  |

|  |  |
| --- | --- |
| **Class Name** | Punto.h |

|  |  |
| --- | --- |
| **Method Name** | Ninguno |

|  |  |
| --- | --- |
| **Parameters** | Ninguno |

|  |
| --- |
| Ninguno |

|  |  |
| --- | --- |
| **Class Name** | Simpson.h |

|  |  |
| --- | --- |
| **Method Name** | Double evalP |

|  |  |
| --- | --- |
| **Parameters** | Double x |
|  | Int dof |
|  | Int numSeg |

|  |
| --- |
| Double W = x/(double) numSeg; //no pierdas el formato |
| Double suma= 0 ; //vas a ocupar un sumador y en dobles |
| For contador = 0 -> numSeg |
| Funx = evalDistT((w\*contador),dof) ///aquí estas llamando a las funciones de T\_dist |
| If contador = 0, or contador == numSeg -> suma += funx |
| If cont%2 == 0 -> suma += 2\*funx //si el residuo es 0 sumalo dos veces |
| Else -> suma += funx\*4 |
|  |
| Return (w/3)\*suma //al final |

|  |  |
| --- | --- |
| **Method Name** | Double evalSimpson |

|  |  |
| --- | --- |
| **Parameters** | Double x |
|  | Int dof |
|  |  |

|  |
| --- |
| Int numSeg = 10 //para el primero nadamas, ahorita lo tienes que multiplicar por 2 |
| //el error que estamos buscando es de .000000001 |
| Double error = .000000001; |
| //valor de p con Simpson, |
| Double pOne = evalP(x,dof,numSeg); |
|  |
| //valor d p con doble numero de segmentos  numSeg \*=2 ; |
| pNext = evalP(x,dof,numSeg); |
|  |
|  |
| //mientras la dif entre las p sea mas grande que el error, sigue doblando los segmentos hasta que no cumpla |
| While ( abs(pNext-pOne)> error |
| pOne = pNext; |
| numSeg \*= 2; |
| pNext = evalP(x,dof,numSeg); |
|  |
| //regresa pNext cuando ya no , ósea afuera del while |
| Return pNext |