*Functional Specification Template*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | David Zavala López | **Program #** | 4 |

**Gamma**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| x: float | Valor decimal al que se le calculará su valor gamma |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| Gamma(x: float) | Calcula el valor de la función Gamma para el número real (x).  If x = 1 -> return := 1  If x = 0.5 -> return := sqrt(pi)  Return := (x-1) \* Gamma(x-1) |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TDistributionPDF**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| x: int | Valor entero que en este caso representa el index por W. |
| dof: int | Valor entero que representa los grados de libertad usados en el cálculo de la integración a través de la regla de Simpson |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| TDistributionPDF(x: float ,dof: int) | Calcula el valor de la función de densidad de probabilidad de la distribución t de Student para el valor x y el grado de libertad dof. Utiliza la siguiente fórmula:  numerator := Gamma((dof + 1) / 2)  denominator := (Gamma(dof / 2) \* ((dof \* pi) \*\* 0.5)))  return := (nominator / denominator) \* (1 + (x \*\* 2 / dof)) \*\* (-(dof + 1) / 2)  Donde Gamma es la función Gamma, dof es el grado de libertad y pi es el valor de pi. |

**SimpsonsRuleIntegration**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| x: int | Valor entero que en este caso representa el index por W. |
| dof: int | Valor entero que representa los grados de libertad usados en el cálculo de la integración a través de la regla de Simpson |
| W: float | Valor decimal que representa el ancho de cada subintervalo para la aproximación integral. |
| n: int | Valor entero que representa la cantidad de segmentos utilizados en el cálculo de la aproximación integral. |
| func: method | Función utilizada en la formula de integración, en este caso representa la función con la fórmula para obtener T-Distibution. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| SimpsonsRuleIntegration(  func: method, n: int, W: float, x: float, dof: int)  func -> en este caso representa la función TDistributionPDF | Calcula la integral de la función utilizando la regla de Simpson.  sumOdds := GetSumOdds()  sumEvens := GetSumEvens()  integral := (W / 3) \* (func(0,dof) + sumOdds + sumEvens + func(x,dof))  return := integral  Donde sumOdds es la suma de las evaluaciones de la función en las posiciones impares dentro del intervalo de integración, sumEvens es la suma de las evaluaciones de la función en las posiciones pares dentro del intervalo de integración, y func es la función que se está integrando. |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**GetSumOdds**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| func: method | Función utilizada en la formula de integración, en este caso representa la función con la fórmula para obtener T-Distibution. |
| dof: int | Valor entero que representa los grados de libertad usados en el cálculo de la integración a través de la regla de Simpson |
| W: float | Valor decimal que representa el ancho de cada subintervalo para la aproximación integral. |
| n: int | Valor entero que representa la cantidad de segmentos utilizados en el cálculo de la aproximación integral. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| GetSumOdds(func: method, n: int, W: float, dof: int)  func -> en este caso representa la función TDistributionPDF | Calcula la suma de las evaluaciones de la función en las posiciones impares dentro del intervalo de integración.  sumOdds := 0  ∀ i ∈ oddNumbers : 0 .. n – 1 | sumOdds := sumOdds + (4 \* simpson(i \* W, dof))  return := sumOdds |

**GetSumEvens**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| func: method | Función utilizada en la formula de integración, en este caso representa la función con la fórmula para obtener T-Distibution. |
| dof: int | Valor entero que representa los grados de libertad usados en el cálculo de la integración a través de la regla de Simpson |
| W: float | Valor decimal que representa el ancho de cada subintervalo para la aproximación integral. |
| n: int | Valor entero que representa la cantidad de segmentos utilizados en el cálculo de la aproximación integral. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| GetSumEvens(func: method, n: int, W: float, dof: int)  func -> en este caso representa la función TDistributionPDF | Calcula la suma de las evaluaciones de la función en las posiciones pares dentro del intervalo de integración. Utiliza la siguiente fórmula:  sumEvens := 0  ∀ i ∈ evenNumbers : 0 .. n – 1 | sumEvens := sumEvens + (2 \* simpson(i \* W, dof))  return := sumEvens |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**IntegrateWithSimpsonsRule**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| x: float | Valor entero que representa el punto dentro del intervalo de integración en el que se está evaluando la función que se está integrando. |
| dof: int | Valor entero que representa los grados de libertad usados en el cálculo de la integración a través de la regla de Simpson |
| n: int | Valor entero que representa la cantidad de segmentos utilizados en el cálculo de la aproximación integral. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| IntegrateWithSimpsonsRule(x: float, dof: int, n=10: int) | Realiza la integración numérica utilizando la regla de Simpson. Itera duplicando el número de segmentos hasta que la diferencia entre dos estimaciones consecutivas de la integral sea menor que una tolerancia predefinida.  E := .0000001  probability1 := SimpsonsRuleIntegration(TDistributionPDF, n, x/n, x, dof)  n := n\*2  probability2 := SimpsonsRuleIntegration(TDistributionPDF, n, x/n, x, dof)  if abs (probability2 – probability1) > E ->  probability1 = probability2;  n \* 2;  repeat;  return := probability2 |

**CalculateIntegrateBySimpsonsRule**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| x: float | Valor entero que representa el punto dentro del intervalo de integración en el que se está evaluando la función que se está integrando. |
| dof: int | Valor entero que representa los grados de libertad usados en el cálculo de la integración a través de la regla de Simpson |
| n: int | Valor entero que representa la cantidad de segmentos utilizados en el cálculo de la aproximación integral. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| CalculateIntegrateBySimpsonsRule(x: float, dof: int, n=10: int) | Calcula e imprime el resultado de la integración numérica utilizando la regla de Simpson. Guarda el resultado en un archivo de caché si no existe previamente.  if fileExists(“Cache/”+str(x)+’-‘+str(dof)+’.txt’) -> return := file  else ->  p := IntegrateWithSimpsonsRule(x, dof, n)  return := p  saveResultInCache(x,dof,p) |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Main**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
|  |  |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| main() | Método principal ejecutable.  Se obtienen los valores para x y dof y se llama al método encargado de usar Simpsons Rule con esos valores  x := askForX()  dof := askForDof()  CalculateIntegrateBySimpsonsRule(x, dof) |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**fileExists**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| path: string | String que representa la ruta al archivo al que se intentará acceder |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| fileExists(path: string) | Verifica si el archivo con el nombre especificado existe en el sistema de archivos. Retorna True si el archivo existe, de lo contrario retorna False.  if fileExists = True -> return := True  if !fileExists = False -> return := False |

**saveResultsInCache**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| content: string | Cadena de texto que representa el contenido a guardar en el archivo |
| name: string | Cadena de texto que representa el nombre del archivo donde se guardara el contenido |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| saveResultsInCache(content,name) | Guarda el contenido en un archivo de caché con el nombre especificado.  path := “Cache”+name  writeInFile(path,content) |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**hasCharIn**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| string: string | Cadena de texto en la que se haran las pruebas o procedimientos |
| character: string | Cadena de texto, o carácter dependiendo el lenguaje, que representa el carácter que se buscara. |
| position: int | Número entero que representa la posición en la que se buscara el carácter |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| hasCharIn(string, position, character) | Verifica si un carácter específico está presente en una posición dada dentro de una cadena de texto.  Retorna True si el carácter está presente en la posición especificada, de lo contrario retorna False.  if charInPosition = True -> return := True  if ! charInPosition = False -> return := False |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**isBlank**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| string: string | Cadena de texto en la que se haran las pruebas o procedimientos |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| isBlank(string: string) | Determina si una cadena de texto dada está en blanco, es decir, si solo contiene espacios en blanco o si está vacía.  Retorna True si la cadena está en blanco, de lo contrario retorna False.  if isBlank = True -> return := True  if ! isBlank = False -> return := False |

**isNumber**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| string: string | Cadena de texto en la que se haran las pruebas o procedimientos |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| isNumber(string: string) | Verifica si una cadena de texto representa un número válido, permitiendo solo caracteres numéricos, coma (,) y punto (.).  Retorna True si la cadena representa un número válido, de lo contrario retorna False.  if isNumber = True -> return := True  if ! isNumber = False -> return := False |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**extractCant**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| string: string | Cadena de texto en la que se haran las pruebas o procedimientos |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| extractCant(string: string) | Extrae un entero que representa la cantidad indicada en una marca de comentario especial dentro de la cadena de texto proporcionada.  Retorna el entero que representa la cantidad extraída.  string := string.strip()  string := string.strip[5:]  return := string |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**removeCommas**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| string: string | Cadena de texto en la que se haran las pruebas o procedimientos |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| removeCommas(string: string) | Elimina todas las comas de una cadena de texto dada.  Retorna la cadena de texto sin comas.  ∀ char ∈ realNumbers: 0 .. string.length | if char != ‘,’ | newValue := newValue + char  Return := newValue |

**convertToFloat**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| string: string | Cadena de texto en la que se haran las pruebas o procedimientos |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| convertToFloat(string: string) | Intenta convertir una cadena de texto en un número de punto flotante (float), manejando excepciones en caso de fallos.  Retorna el número de punto flotante si la conversión es exitosa, de lo contrario retorna None.  f := float(string)  return := f |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**convertToInt**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| string: string | Cadena de texto en la que se haran las pruebas o procedimientos |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| convertToInt(string: string) | Intenta convertir una cadena de texto en un número entero (int), manejando excepciones en caso de fallos.  Retorna el número entero si la conversión es exitosa, de lo contrario retorna None.  i := float(string)  return := i |

**askForX**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
|  |  |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| askForX() | Solicita al usuario que ingrese el valor de x.  Valida que se ingrese un valor no vacío y numérico.  Retorna el valor de x ingresado como un número de punto flotante.  inputValue := inputX  if isBlank(inputValue) -> repeat;  if ! isNumber(inputValue) -> repeat;  inputValue := removeCommas(inputValue)  inputAsNumber := convertToFloat(inputValue)  if inputAsNumber == null -> repeat;  if inputAsNumber < 0 -> repeat;  return := inputAsNumber |

**askForX**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
|  |  |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| askForDof() | Solicita al usuario que ingrese el valor de dof.  Valida que se ingrese un valor no vacío y numérico.  Retorna el valor de dof ingresado como un número entero.  inputValue := inputDof  if isBlank(inputValue) -> repeat;  if ! isNumber(inputValue) -> repeat;  inputValue := removeCommas(inputValue)  inputAsNumber := convertToInt(inputValue)  if inputAsNumber == null -> repeat;  if inputAsNumber < 0 -> repeat;  return := inputAsNumber |