*Functional Specification Template*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | David Zavala López | **Program #** | 5 |

**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 |

**FindXBySimpsonsRule**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| p: float | Valor decimal que representa la probabildiad objetivo que se busca obtener a través de los calculos |
| x: float | Valor decimal 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. |
| d: float | Valor decimal que representa el incremento delta que se le dará a x en cada iteración. |
| E: float | Valor decimal que representa el márgen de error aceptable. |
| lastResult: float | Valor decimal que representa el último valor obtenido en el resultado de la iteración anterior a la actual. |
| posOfLastP | Valor de cadena de caracteres que se representa un indicador para saber si el resultado de la iteración anterior fue menor o mayor al resultado objetivo. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| FindXBySimspsonsRule(p:float, dof:int, n:int, x:float, d:float, E:float, lastResult:float, posOfLastP:str) | Realiza la integración numérica utilizando la regla de Simpson. Esto para encontrar valores que, progresivamente, se acerquen al valor objetivo, dado por el usuario. Itera tomando en cuenta si el resultado fue menor o mayor a lo esperado para saber si aumentar o reducir el valor de x.  pFromCalculation := SimpsonsRuleIntegration(TDistributionPDF, n, x/n, x, dof)  if abs (probability2 – probability1) > E ->  if pFromCalculation < p ->  if posOfLastP != “Lower” && posOfLastP != “” ->  d := d/2  x := x+d  return FindXBySimspsonsRule(p,dof,n,x,d,E,pFromCalculation,"Lower")  if pFromCalculation > p ->  if posOfLastP != “Higher” && posOfLastP != “” ->  d := d/2  x := x+d  return FindXBySimspsonsRule(p,dof,n,x,d,E,pFromCalculation,"Higher")  return x |

**CalculateX**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| p: float | Valor decimal que representa la probabildiad objetivo que se busca obtener a través de los calculos |
| x\_initial: float | Valor decimal 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. |
| E: float | Valor decimal que representa el márgen de error aceptable. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| CalculateIntegrateBySimpsonsRule(x: float, dof: int, n: 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.  x := FindXBySimspsonsRule(p,dof,n,x\_initial,x\_initial/2,E)  result := f" p = {round(p,5):.5f}\ndof = {round(dof,5):.0f}\n x = {round(x,5):.5f}"  imprimir result  return x |

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

**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  p := askForFloatInput("Ingrese valor de p: ",0,True,0.5,True)  dof := askForIntInput("Ingrese valor de dof (grados de libertad): ",0)  CalculateX(p,dof) |

**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  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 |

**askForFloatInput**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| message: string | Valor de cadena de caracteres que representa el mensaje a imprimir para indicar al usuario qué debe ingresar como input. |
| min: float | Valor decimal que representa el valor minimo que debe tener el input si aplica |
| minAccepted: boolean | Valor booleano que indica si el valor minimo es aceptado o no dentro del rango de aceptación de valores para ese input. |
| max: float | Valor decimal que representa el valor maximo que debe tener el input si aplica |
| maxAccepted: boolean | Valor booleano que indica si el valor maximo es aceptado o no dentro del rango de aceptación de valores para ese input. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| askForFloatInput(message:str,  min:float,minAccepted:bool,  max:float,maxAccepted:bool) | Solicita al usuario que ingrese un valor de punto flotante.  Valida que se ingrese un valor no vacío, numérico y decimal.  Retorna el valor ingresado como un número de punto flotante.  if message != null ->  imprimir mensaje  inputValue := input  else ->  imprimir “Ingrese valor decimal: “  inputValue := input  if isBlank(inputValue) -> repeat;  if ! isNumber(inputValue) -> repeat;  inputValue := removeCommas(inputValue)  inputAsNumber := convertToFloat(inputValue)  if inputAsNumber == null -> repeat;  if min != null ->  if minAccepted == True ->  if inputAsNumber <= min ->  imprimir advertencia  repeat  if minAccepted == False->  if inputAsNumber < min ->  imprimir advertencia  repeat  if max != null ->  if maxAccepted == True ->  if inputAsNumber >= max ->  imprimir advertencia  repeat  if maxAccepted == False->  if inputAsNumber > max ->  imprimir advertencia  repeat  return := inputAsNumber |

**askForIntInput**

|  |  |
| --- | --- |
| **Atributos** | |
| **Declaración** | **Descripción** |
| message: string | Valor de cadena de caracteres que representa el mensaje a imprimir para indicar al usuario qué debe ingresar como input. |
| min: float | Valor decimal que representa el valor minimo que debe tener el input si aplica |
| minAccepted: boolean | Valor booleano que indica si el valor minimo es aceptado o no dentro del rango de aceptación de valores para ese input. |
| max: float | Valor decimal que representa el valor maximo que debe tener el input si aplica |
| maxAccepted: boolean | Valor booleano que indica si el valor maximo es aceptado o no dentro del rango de aceptación de valores para ese input. |

|  |  |
| --- | --- |
| **Ítems** | |
| **Declaración** | **Descripción** |
| askForIntInput(message:str,  min:float,minAccepted:bool,  max:float,maxAccepted:bool) | Solicita al usuario que ingrese un valor entero.  Valida que se ingrese un valor no vacío, numérico y entero.  Retorna el valor ingresado como un número entero.  if message != null ->  imprimir mensaje  inputValue := input  else ->  imprimir “Ingrese valor entero: “  inputValue := input  if isBlank(inputValue) -> repeat;  if ! isNumber(inputValue) -> repeat;  inputValue := removeCommas(inputValue)  inputAsNumber := convertToInt(inputValue)  if inputAsNumber == null -> repeat;  if min != null ->  if minAccepted == True ->  if inputAsNumber <= min ->  imprimir advertencia  repeat  if minAccepted == False->  if inputAsNumber < min ->  imprimir advertencia  repeat  if max != null ->  if maxAccepted == True ->  if inputAsNumber >= max ->  imprimir advertencia  repeat  if maxAccepted == False->  if inputAsNumber > max ->  imprimir advertencia  repeat  return := inputAsNumber |