Operational Specification Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Pisit Pisuttipunpong | **Date** | 5/1/2025 |
| **Program** | Program 6 | **Program #** | 6 |
| **Instructor** | Sakasit Ramingwong | **Language** | JavaScript |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario Number** | **1** | **User Objective** | Get x from binary search for finding given expected p | |
| **Scenario Objective** | | Perform Binary Search to find x given expected p | | |
| **Source** | **Step** | **Action** | | **Comments** |
| User | 1 | Input p, degree of freedom | |  |
| User | 2 | Run Program 6 | |  |
| System | 3 | Perform Binary Search in Simpson Logic | |  |
| System | 4 | Output X (which gives p) | |  |
| User | 5 | Read program output | |  |
| User | 6 | Exit Program | | Thank you Program 6 😊 |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |

Functional Specification Design

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Student** | | Pisit Pisuttipunpong | | **Date** | 5/1/2025 |
| **Program** | | Program 6 | | **Program #** | 6 |
| **Instructor** | | Sakasit Ramingwong | | **Language** | JavaScript |
|  | | | | | |
|  | | | | | |
| **Variables** | | | | | |
|  | **Declaration** | | **Description** | | |
|  | Let num\_seg = 10 | | The initial number of segments | | |
|  | Let W = x/num\_seg | | The width of each segment | | |
|  | Let E = 0.00001 | | The minimum error for the Simpson value | | |
|  | Let dof | | Degree of freedom | | |
|  | Let x | | The x value which gives expected p by binary search | | |
|  | Let F(x) | | T distribution function | | |
|  | Let p | | The expected P to find x | | |
|  | Let D | | The modifying difference in Binary Search | | |
|  | Let prev\_state | | The previous state of binary search operation | | |
|  | | | | | |
| **Functions** | | | | | |
|  | **Declaration** | | **Description** | | |
|  | Function gamma\_function | | Calculate gamma function it like factorial | | |
|  | Function t\_distribution\_function | | Calculate T distribution F(x) | | |
|  | Function Simpson\_function | | The Simpson function p | | |
|  | Function Find\_p\_function | | The function to find p based on predicted value x | | |

Logic Specification Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Pisit Pisuttipunpong | **Date** | 5/1/2025 |
| **Program** | Program 6 | **Program #** | 6 |
| **Instructor** | Sakasit Ramingwong | **Language** | JavaScript |

|  |
| --- |
| Pseudo Code |
| Setting Up the Simpson Variables |
| Setting Up the Binary Search Variable |
| Perform Binary Search |
| Set initial value of X = 1 |
| Set modifying distance D = 0/5 |
| Output p less than expected p |
| If not the same as the previous direction, then decrease D by half |
| Increase x by D |
| Output p greater than expected p |
| If not the same as the previous direction, then decrease D by half |
| Decrease x by D |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

State Specification Template

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Student** | | Pisit Pisuttipunpong | | | **Date** | | 5/1/2024 |
| **Program** | | Program 6 | | | **Program #** | | 6 |
| **Instructor** | | Sakasit Ramingwong | | | **Language** | | JavaScript |
|  | | |  | | | | |
| **State Name** | | | | **Description** | | | |
| Start | | | | Set x=1, D=x/2 | | | |
| End | | | | P(x) = Pex | | | |
| Too High | | | | P(x) > Pex | | | |
| Too Low | | | | P(x) < Pex | | | |
|  | | | |  | | | |
| **Function/Parameter** | | | | **Description** | | | |
| X | | | |  | | | |
| P | | | |  | | | |
| Dof | | | |  | | | |
| D | | | |  | | | |
|  | | | |  | | | |
|  | | | |  | | | |
| **States/Next States** | | | | **Transition Condition** | | **Action** | |
| Start | | | |  | |  | |
| End | | | P(x) = Pex | | Return x, end the program | |
| Too high | | | P(x) > Pex | | Increase value of X by D | |
| Too low | | | P(x) < Pex | | Decrease value of X by D | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |
| Too high | | | |  | |  | |
| End | | | P(x) = Pex | | Return x, end the program | |
| Too low | | | P(x) < Pex | | Decrease value of X by D, halve value of D | |
| Too high | | | P(x) > Pex | | Increase value of X by D | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |
| Too low | | | |  | |  | |
| End | | | P(x) = Pex | | Return x, end the program | |
| Too low | | | P(x) < Pex | | Decrease value of X by D | |
| Too high | | | P(x) > Pex | | Increase value of X by D, halve value of D | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |
|  | | |  | |  | |