|  |  |  |
| --- | --- | --- |
| Category | Test Data | Expected Result |
| 1. Enter first if | 1 | timeConversion() |
| 1. Enter first else if | 2 | seasonDetermination() |
| 1. Enter second else if | 3 | Output: “Goodbye!” |
| 1. Enter final else | 22 | Output: “Please enter only choice 1 , 2 or 3” |
| 1. Exception: non numeric input | aa | Caught by catch, output “ Invalid input. Please enter only number 1 , 2 or 3” |
| 1. Loop repeats | 0 | Menu repeats until choice == 3 (exit) |

**White Box Testing (I see inside the code, test conditions, logic)**

**Black Box Testing (I dk how it works, I just test what it does)**

|  |  |  |
| --- | --- | --- |
| Category | Test Data | Expected Result |
| 1. Valid input -> Choice 1 | 1 | timeConversion() |
| 1. Valid input -> Choice 2 | 2 | seasonDetermination() |
| 1. Valid input -> Choice 3 | 3 | Output: “Goodbye!” |
| 1. Invalid menu input | 5 | Output: “Please enter only choice 1 , 2 or 3” |
| 1. Non-numeric input | a | Output: “Invalid input. Please enter only number 1 , 2 or 3” |
| 1. Symbol input | @ | Output: “Invalid input. Please enter only number 1 , 2 or 3” |
| 1. Negative number input | -1 | Output: “Please enter only choice 1 , 2 or 3” |

**White-box testing** is used to verify the internal logic of the program by checking if all possible code paths (e.g., if, else, catch) behave correctly.

**Black-box testing** focuses on validating the program's input and output behavior without knowing the internal code, ensuring it meets user expectations.