Documentation Template

|  |
| --- |
| ***You MUST provide evidence showing how the problem has been decomposed, how the components have been developed and trialled, and of how they have been assembled and tested to create a final, working outcome.*** |

### **Outline / Decomposition**

*Please write down your task decomposition here (a numbered list is a good idea)*

1. User is asked if they have played before, if no show instructions, if yes continue.
2. User is asked how many rounds they want to play or if they want to play infinitely.
3. User is asked to enter either rock / paper / scissors.
4. User’s input is compared with the computer’s random answer, and they are told if they win / lose / draw, their score is also displayed.
5. Once the user quits or runs out of rounds, they are shown the game stats / history with all their wins and losses.

Components I will need:

* Instruction component (ask user if they have played before, show instructions or continue program) i.e., yes/no checker
* Round component (ask user how many rounds they want to play, number checker component invalid input: negative numbers, zero, or decimals)
* Continuous component (if user says they want they want to play endless rounds, then play infinite rounds until user quits)
* User choice component (ask user for either rock / paper / scissors and generate a random choice from the computer, then compare the choices to see who won)
* Score component (store the user’s score with wins, losses and draws)
* Summary component (show the score sheet with each win, loss and draw, along with the choices made by the player and computer in each round)

### **Flowchart**

*Please show a developed flowchart of your program below (you may use draw.io to create your flowchart)*

*Diagram, engineering drawing

Description automatically generated*

### **Version Log**

*Your version log should go here. Annotated screenshots are a good idea at this point*

**User Choice Component**

Text

Description automatically generated

Text

Description automatically generated

**Version #1 for user choice component (02\_user\_choice\_v1):**

Checks user choice using four if / elif statements. Works for lowercase and uppercase responses and returns full word (eg: ‘r’ returns ‘rock).

Text

Description automatically generated

Text

Description automatically generated

**Version #2 for user choice component (02\_user\_choice\_v2):**

Almost identical to version #1 but error messages can be customized (ie: error message a parameter of the function)

Text

Description automatically generated

Text

Description automatically generated

**Version #3 for user choice component (02\_user\_choice\_v3):**

Checks if response is in a given list (or is the first letter of an item in the list). Can be used for r/p/s or yes/no responses. Only works if first letter of options is different. Does allow users to use ‘q’ or ‘quit’ as the exit code. I will use this function in assembled program.

Text

Description automatically generated**Round Mechanics Component**

Text

Description automatically generated

**Version 1 (03\_RPS\_round\_mechanics\_v1):**

Asks for rounds in the main routine and uses an integer checking function to allow users to press an integer or <enter>. The code is quite inefficient and clunky.

Text

Description automatically generated

Shape

Description automatically generated with medium confidence

**Version 2 (03\_RPS\_round\_mechanics\_v2):**

Similar to version #1 but much more efficient removing repeating lines of code by incorporating continuous mode and rounds mode together.

**Computer Choice Component**

Graphical user interface, text, application

Description automatically generated

**Version 1 (04\_RPS\_comp\_choice\_v1):**

Similar to the lucky unicorn token generator, but with a list using rock, paper and scissors (no exit code.

Text

Description automatically generated with medium confidence

**Version 2 (04\_RPS\_comp\_choice\_v2):**

I have added the exit code to the list but to exclude it from the computer choice, I have excluded the last item from the list, so that I do not have to create two different lists. I will add this component to by base component.

**Compare Choice Component**

Text

Description automatically generated

**Version 1 (05\_RPS\_compare\_v1):**

The code works by having a for loop in another for loop, but there are too many if/elif/else statements that are long and inefficient.

Text

Description automatically generated

**Version 1 (05\_RPS\_compare\_v2):**

The code has been made more efficient through the use of three if/elif/else statements. I will use this code in my base component.

### **Component Testing**

*Show that you have tested each component here. You should have a test plan and then screenshot proof for each component. You should also include notes that justify the major decisions you made.*

**User Choice Component**

**User Choice Test Plan**

|  |  |
| --- | --- |
| **Test Data** | **Expected Outcomes** |
| Rock | Accepted, changed to ‘rock’ (note lowercase) |
| PAPER | Accepted, changed to ‘paper’ |
| scissors | Accepted, stays as ‘scissors’ |
| R | Accepted, changed to ‘rock’ |
| p | Accepted, changed to ‘paper’ |
| S | Accepted, changed to ‘scissors’ |
| <enter> | Error message ‘please enter rock, paper or scissors |
| QUIT | Accepted, changed to ‘quit,’ loop ends |
| q | Accepted, changed to ‘quit,’ loop ends |

**User Choice Testing**

**Text

Description automatically generated**

Results are as expected

**Round Mechanics Component**

**Round Mechanics Test Plan**

|  |  |
| --- | --- |
| **Test Data** | **Expected Outcomes** |
| xlii | <error, please enter an integer more than 0> |
| 1.5 | <error, please enter an integer more than 0> |
| <enter>  3 rounds  quit | Program goes for 3 rounds and ends when exit code is entered |
| 0 | <error, please enter an integer more than 0> |
| 1 | Program goes for 1 round |

**Round Mechanics Testing**

Text

Description automatically generated

**Computer Choice Component**

**Computer Choice Test Plan**

|  |  |
| --- | --- |
| **Test Data** | **Expected Outcomes** |
| Randomly generate 20 items (rock / paper / scissors) | Each of rock / paper / scissors should appear in the list several times. The order should be random. List should not include exit code. |

**Computer Choice Testing**



Each item appears several times as expected.

**Compare Choices Component**

**Compare Choices Plan**

* Three ways to win
* If it’s the same, it’s a tie
* Otherwise, it’s a loss

**Compare Choices Test Plan**

|  |  |
| --- | --- |
| Test Data | Expected Outcomes |
| Paper vs rock | User wins |
| Rock vs scissors | User Wins |
| Scissors vs paper | User Wins |
| Paper vs paper | Draw |
| Rock vs rock | Draw |
| Scissors vs scissors | Draw |
| Paper vs scissors | User loses |
| Rock vs paper | User loses |
| Scissors vs rock | User loses |

**Compare Choices Testing**

Text

Description automatically generated

All cases worked as expected

### Assembled Outcome Testing

*Please show testing for your assembled outcome below. This should include a test plan followed by screenshot proof*

### Usability Testing

*Write a list of things improvements which need to be made based on your usability testing. Then write down what you changed.*

### Post Usability Test…

*Show that your post usability testing program works correctly*

### Social and End User Considerations…

**How did you ensure that your task was suitable for your chosen audience?**

*Answer here*

**How have you honoured copyright?**

*Answer here*

**How did you make your quiz easy to use?**

*Answer here*

\*When you actually do it, have plan for component, then test plan and testing.