Planning sheet

Task 1: Describe the problem  
  
The program will take a recipe with a certain amount of serves, then take a new number of serves and alter the recipe by a scale factor to create an updated recipe.

Task 2: Identify the input information

What information will the user have to enter? Copy and complete this table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable name** | **Scope** | **Data type** | **Purpose of variable** |
| old\_servings | Global? | int | The amount of servings the old recipe made |
| new\_servings | Global? | int | The amount of servings the new recipe made |
| ingredient | global | Str | What the ingredient is (stored in list) |
| measurement | global | Str | How it is measured (stored in list) |
| amount | global | float | How much of that measurement (stored in list??) |

Task 3: Identify other variable information

What information will the program need to store? Also include constants and derived values as they will maximise the flexibility and robustness of your plan. Your chosen scope should match the way the variable will be used in your program.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable name** | **Scope** | **Data type** | **Purpose of variable and where it comes from.** |
| scale\_factor | Local? | float | What to times the amount by, comes from new\_servings/old \_servings |
|  |  |  |  |

Task 4: Identify the indexed data (list) structure

What information will be stored in a list?

|  |  |  |  |
| --- | --- | --- | --- |
| **List name** | **Scope** | **Data type** | **Purpose** |
| ingredients | global | strings | This list holds what the ingredient actually is, e.g. flour, milk |
| units | global | strings | This list holds how the matching ingredient is measured, e.g. cups, teaspoons |
| amounts | global | Floats | This list holds the amount of each unit for each ingredient, and this is the list the calculations will be performed on. |
| new\_amount | Global? | List of floats | Loop through the amount list and times each one by the scale factor, then append these new values to new\_amount |

Task 5: Identify the output information

What information will the program need to print out?

|  |  |  |
| --- | --- | --- |
| **Output** | **When it happens** | **Details** |
| Finished recipe | At the end | Loop through each list, print format like [amount] [measurement] [ingredient] |

Task 6: Write the algorithm

* Print out instructions on how the program works
* Ask for input of the original recipe
  + Menu based?
    - I – ingredient
    - M – measurement
    - A – amount
    - Q – quit
  + Or:
    - Enter the ingredient or press q to quit:
    - Enter how it is measured:
    - Enter how much of each unit there is:
* Input – how many people does the new recipe serve?
* Input – how many people should the new recipe serve?
* Scale factor = new serve amount / old serve amount
* While true:
  + If scale factor > 3:
    - Print upper limit bound. error message
  + Elif scale factor < 0.25:
    - Print lower limit bound. error message
  + Else:
    - Break
* For item in amounts:
  + Times the number by the scale factor
  + Append it to the new amounts list
* Output the new recipe:
  + For loop, using the length of one of the lists (they should all be the same length)
  + Print the ingredient, the unit, and the NEW amount on one line
  + The next one goes on the next line

Task 7: Testing Plan

Include a testing schedule with input cases that ensures that your program will work correctly on all inputs – **expected, boundary, and exceptional input**.

Step through your algorithm showing the outputs that are generated. If you find problems with your algorithm make changes to it and run through your algorithm again until it works.

|  |  |
| --- | --- |
| **TEST** | **RESULT** |
| Ingredient amount input (expected) | |
| 2, 3.5, etc. (floats and ints) | It will store the amount and ask for the next ingredient. |
| Ingredient amount input (unexpected) | |
| “Three”, [enter], anything that isn’t a number | It will come up with an error message and ask you to try again. |
| Serves input – original or new (expected) | |
| 3, 12, etc. (ints) | It will store the amount and then ask for the new serve amount (if it is the original) or move on (if it is the new). |
| Serves input – original or new (unexpected) | |
| 4.5, 3.2, etc. (any float) | It will come up with an error message and ask for a whole number. |
| “hello”, “seven”, [enter], etc. anything that isn’t a number | It will come up with an error message and ask for a whole number. |
| Scale factor calculation (expected) | |
| 2, 0.5, etc. | It will accept this and calculate/output the new recipe. |
| Scale factor calculation (boundary expected) | |
| 3 | It will accept this and calculate/output the new recipe. |
| 0.25 | It will accept this and calculate/output the new recipe. |
| Scale factor calculation (boundary unexpected) | |
| 3.1 | It will print the upper error message and ask the user re-enters the new serves amount. |
| 0.24 | It will print the lower error message and ask the user re-enters the new serves amount. |
| Scale factor calculation (exceptional unexpected) | |
| 7, 23, etc. (anything over 3) | It will print the upper error message and ask the user re-enters the new serves amount. |
| 0.1, -32 (anything below 0.25) | It will print the lower error message and ask the user re-enters the new serves amount. |