CS118 Problem Solving
Applying to Computers

#1: Regarding the microwave analogy from an earlier exercise:

Analogy
When you press buttons on a microwave oven, you are providing instructions to the device ("Turn on for 60 seconds") and then you execute those instructions by pressing another button ("Start"). Some microwave ovens permit you to provide longer "programs" ("Cook for 5 minutes at 100% power, then cook for 10 minutes at 50% power"). Those instructions are not saved, so you get to enter them every time you want to cook something. The analogy applies to computer programs except that we'll save our "cooking instructions" for re-use. And they'll be much more complex in what they can accomplish.

a. What are possible INPUTS to that "program"?

a. Examples of inputs:
- Time to cook
- Power level to cook
- Start the oven
- Some ovens have temperature probes
- etc.

b. What are possible OUTPUTS from that "program"?
(Remember: Outputs do NOT have to go to the user.)

b. Examples of outputs:
- Time remaining
- Turn on/off magnatron
- Turn on/off light
- Beep when done
- etc.

For each of the following exercises, prepare a simple solution that uses any or all of the four tools specified above – INPUT, OUTPUT, DEFINE. Use only one tool per task. Follow the format of the example.

2. Have a computer program compute the horizontal distance a projectile travels and display it to the user. Use the Internet to determine the information that will be needed from the user to compute the desired value.

```
T01: INPUT from user: "speed"
T02: INPUT from user: "angle"
T03: DEFINE: "horiz_distance" using speed, angle
T04: OUTPUT to user: horiz_distance
```

3. Have a computer program find and display to the user the real roots of a quadratic equation. Assume the user provides only information which will result in real roots (i.e. do not attempt to determine if the root will be real, and do not attempt to have the user correct mistakes).

```
T01: INPUT from user: "coeff_a"
T02: INPUT from user: "coeff_b"
T03: INPUT from user: "coeff_c"
T04: DEFINE: "root1" using coeff_a, coeff_b, coeff_c
T05: DEFINE: "root2" using coeff_a, coeff_b, coeff_c
T06: OUTPUT to user: root1, root2
```

4. Have a computer program determine the GPA of the user for the most recent completed semester and save that information on disk. Assume the user took exactly four classes (all equal credits) and received a grade of 4, 3, 2, 1, or 0 in each class..

```
T01: INPUT from user: "grade1"
T02: INPUT from user: "grade2"
T03: INPUT from user: "grade3"
T04: INPUT from user: "grade4"
T05: DEFINE: "GPA" using grade1, grade2, grade3, grade4
T06: OUTPUT to disk: GPA
```

Note: When the wording can be made unambiguous, listing of variables can be abbreviated:
e.g. `DEFINE: "GPA" using all variables for grades`

For each of the following exercises, prepare a simple solution that uses any or all of the tools specified above, except for `REPEAT` and `EXECUTE`. Use only one tool per task. Follow the format of the example – note that the `TEST` and `REPEAT` tools require you to specify which tasks they are controlling, so you will need to number all of your tasks.

5. Have a computer program compute the horizontal distance a projectile travels and display it to the user but only if the angle with the horizontal is greater than 0 – otherwise, display an error message to the user and stop execution. [HINT: There is no tool to "stop execution". To stop executing means to have no more tasks to process.]

```
T01: INPUT from user: "speed"
T02: INPUT from user: "angle"
T03: TEST: if angle>0, T04-T05
   T04: DEFINE: "horiz_distance" using speed, angle
   T05: OUTPUT to user: horiz_distance
T06: TEST: if angle<=0, T07
   T07: OUTPUT to user: error message
```

6. Have a computer program find and display to the user the real roots of a quadratic equation. Determine if the root will be real and display an error message to the user if they will not be and stop execution. Otherwise, compute the real roots. Do not attempt to have the user correct mistakes.

```
T01: INPUT from user: "coeff_a"
T02: INPUT from user: "coeff_b"
T03: INPUT from user: "coeff_c"
T04: DEFINE: "discriminant" using coeff_a, coeff_b, coeff_c
T05: TEST: if discriminant>0, T06-T08
   T06: DEFINE: "root1" using coeff_a, coeff_b, coeff_c
   T07: DEFINE: "root2" using coeff_a, coeff_b, coeff_c
   T08: OUTPUT to user: root1, root2
T09: TEST: if discriminant<=0, T10
   T10: OUTPUT to user: error message
```

For each of the following exercises, prepare a simple solution that uses any or all of the tools specified above, except for `EXECUTE`. Use only one tool per task. Follow the format of the example – note that the `REPEAT` tool requires you to specify which tasks they are controlling, so you will need to number all of your tasks.

7. Have a computer program determine and display to the user the number of days the user ate breakfast over a year. Have it ask the user every day for 365 days if s/he ate breakfast and keep count of the number of days that answer was "Yes". Assume the user will answer the question only once per day. Do not use more than ten tasks to do this.

```
T01: DEFINE: "days_ate" as 0

    T02: INPUT from user: "answer_to_question"
    T03: TEST if answer_to_question is Yes, T04
    T04: DEFINE: increment "days_ate"
T05: REPEAT 364 times, T02-T04

T06: OUTPUT days_ate


Note: Since the word “repeat” means to “do again” and
since we’re determining if we will repeat after
having performed T02-T04 at least once, it is
technically correct to repeat 364 times (not 365
times). However, many people will understandably use
365.
```

Option 1

```
T01: DEFINE: "days_ate" as 0

T02: REPEAT 365 times, T04-T05
    T03: INPUT from user: "answer_to_question"
    T04: TEST if answer_to_question is Yes, T05
        T05: DEFINE: increment "days_ate"

T06: OUTPUT days_ate


An alternative approach with the REPEAT above the
repeating tasks. With the REPEAT occurring before
the tasks, it's correct to think of this REPEAT as
"Execute and Consider Again" so 365 is correct.
```

Option 2

```
An alternative way to repeat:
   T01: DEFINE: "count" as 0
   T02: DEFINE: "days_ate" as 0

      T03: INPUT from user: "answer_to_question"
      T04: TEST if answer_to_question is Yes, T05
         T05: DEFINE: increment "days_ate"
      T06: DEFINE: increment "count"
   T07: REPEAT until count is 365, T03-T06

   T08: OUTPUT days_ate
```

The versions above use a counting mechanism which is simplified, ignoring the variable needed for counting (since it's not needed inside the REPEAT).

The version to the left is slightly more complicated in that it maintains a counter variable which is incremented by the tasks in the REPEAT. (Because of how the counter is initially defined and the counter is used by the condition, we know that its limit is correctly 365.)

8. Have a computer program ask the user to provide both the credit hours and grade value (e.g. A=4, B=3, etc) for every class taken in school. Do not ask the user how many classes there are – if the user provides a 0 for credit hours for a class, do not ask for the grade value and stop collecting information. After information is done being collected, report to the user the total number of credit hours and the average GPA. Assume the user will provide at least one class worth more than 0 credit hours. Do not use more than ten tasks to do this exercise.

```
   T01: DEFINE "grade_points" as 0
   T02: DEFINE "total_hours" as 0
   T03: INPUT from user: "credit_hours"

   T04: REPEAT while credit_hours>0, T05-T08
      T05: INPUT from user: "grade_value"
      T06: DEFINE: add credit_hours to "total_hours"
      T07: DEFINE: "grade_points" using grade_points, credit_hours, grade_value
      T08: INPUT from user: "credit_hours"

   T09: DEFINE: "GPA" using grade_points, total_hours
   T10: OUTPUT to user: total_hours, GPA
```

$$GPA = \frac{grade\ points}{total\ hours} = \frac{\sum_{k=1}^{N} CH_k \cdot G_k}{\sum_{k=1}^{N} CH_k}$$

where :
CH_k is Credit hours for class k
G_k is Grade for class k

Note in T07 that since we are keeping a running total, "grade_points" must be used as well as defined. There was no need for this in T06 because the wording showed that "total_hours" was being both defined and used.