Project Name: CS003C Project 1

Group Members: Adit Gautam (solo)

Project Description: Basic I/O

Date: 1/27/25

**Part One: Algorithms and flowcharts**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Exercise 1.1**

In the following diagram write a question in the blank box that best fits the statements that have appeared after the box. Assume that just before the blank box you have read the values for *x* and *y*.

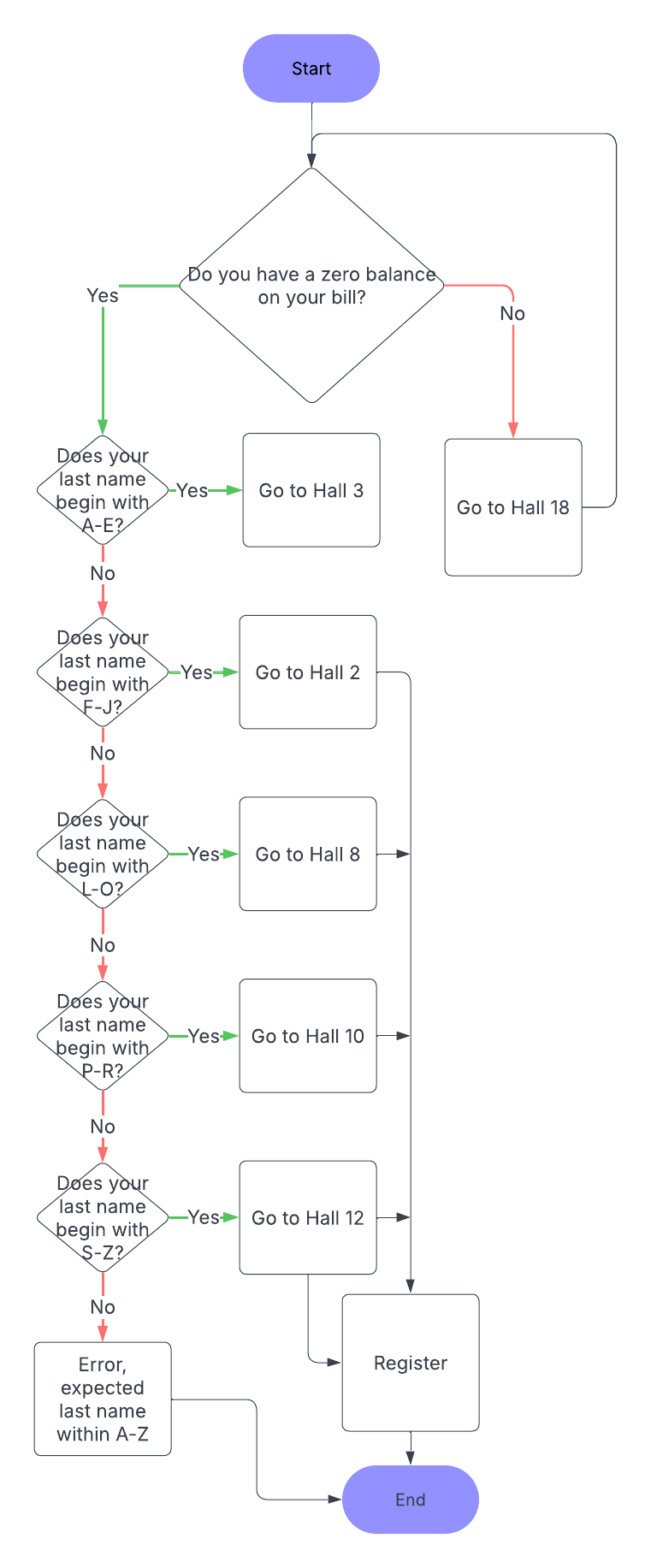


In the following diagram write an English statement in the blank box that best fits the statement that has appeared before the box.

**Exercise 1.2**   
Suppose you are helping the university registrar’s office with the registration process. You are to send students to six different halls depending on the first letter of their last names and the balance that has appeared on their bills.  Here are the criteria you will use to separate them:   
Students with a balance of zero, Letters:   
A-E in Hall 3, F-J in Hall 2, L-O in Hall 8, P-R in Hall 10, and S-Z in Hall 12.

Students with a non-zero balance go to Hall 18. These students can go back to register once they have a zero balance on their bills.

**Draw a flowchart to show the steps that will take to solve this problem.**

**Exercise 1.3**

Write an algorithm in Pseudo Code that demonstrates the method we use to solve a long division problem. Refer to the algorithm examples from Ch. 1 slides.

* Long Division Algorithm
* **Start**
* Result = “” #Start with an empty string to concatenate
* Is the divisor zero?
  + **Terminate the algorithm, cannot divide by 0**
* Divisor length = number of digits in the divisor
* Dividend length = number of digits in the dividend
* Is divisor length > dividend length?
  + Append the difference to trailing zeroes in the dividend
* Remainder = dividend #So we know when to terminate the while loop
* Parse the dividend to get the leading digits that match the divisor length
  + Declare as temp\_dividend
* While remainder != 0 :
  + Temp\_result = temp\_dividend// divisor #Get the integer portion
  + Remainder = remainder - temp\_result
  + Result = result + temp\_result #Concatenate
  + temp\_dividend = temp\_dividend + next digit of the dividend
* Once remainder == 0, terminate the loop
* Print the result
* **End**

**Part Two: A Simple Python Program**

**1) Program Design - Algorithm**   
Before we attempt to write the program, let's develop an algorithm for solving the problem. Make sure to include all 4 steps.

**Exercise 2.1**  
Design the **algorithm** for this problem written in Pseudo Code (plain English).  Remember your algorithm must be precise. This algorithm must be translated to Python to obtain the program.

**Step 1:**

* Read inputs for height, width and cost per gallon
* Wall area = wall height \* wall width

**Step 2:**

* number of gallons = wall area / gallon coverage (350)
* Print wall area rounded to 1 decimal

**Step 3:**

* Cans needed = ceil(number of gallons)
* Print cans needed

**Step 4:**

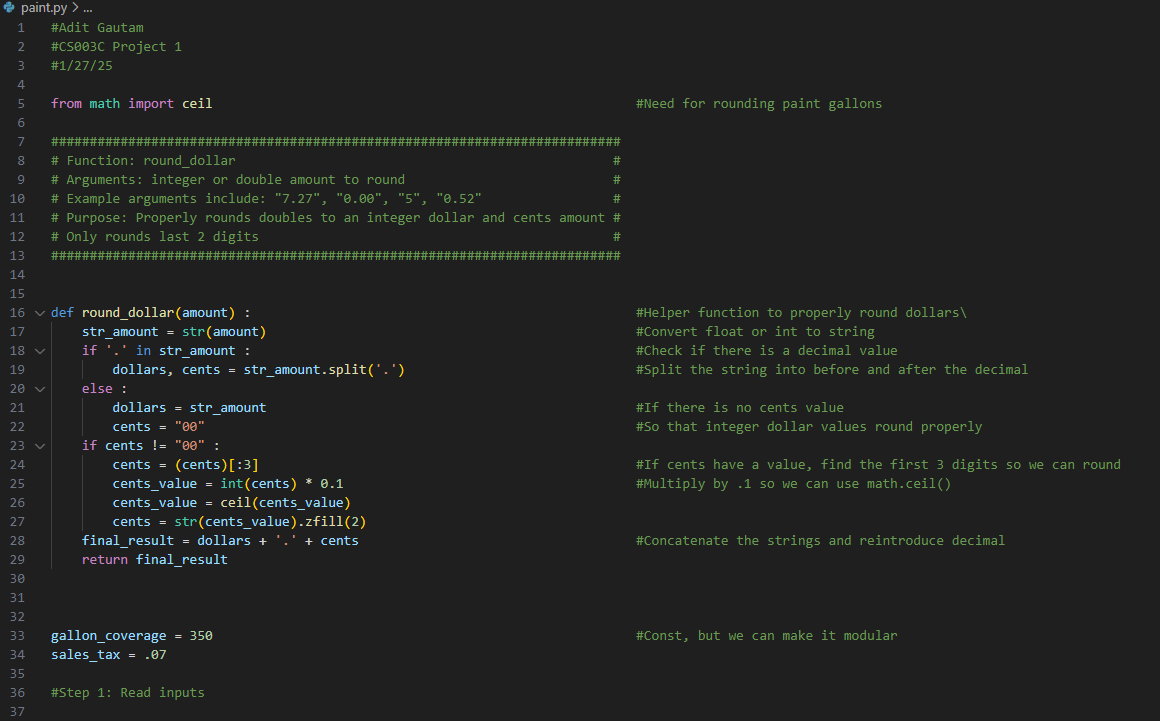
* Paint cost = cans needed \* cost per gallon
* Print paint cost
* Tax = paint cost \* tax (.07)
* Total cost = paint cost + tax

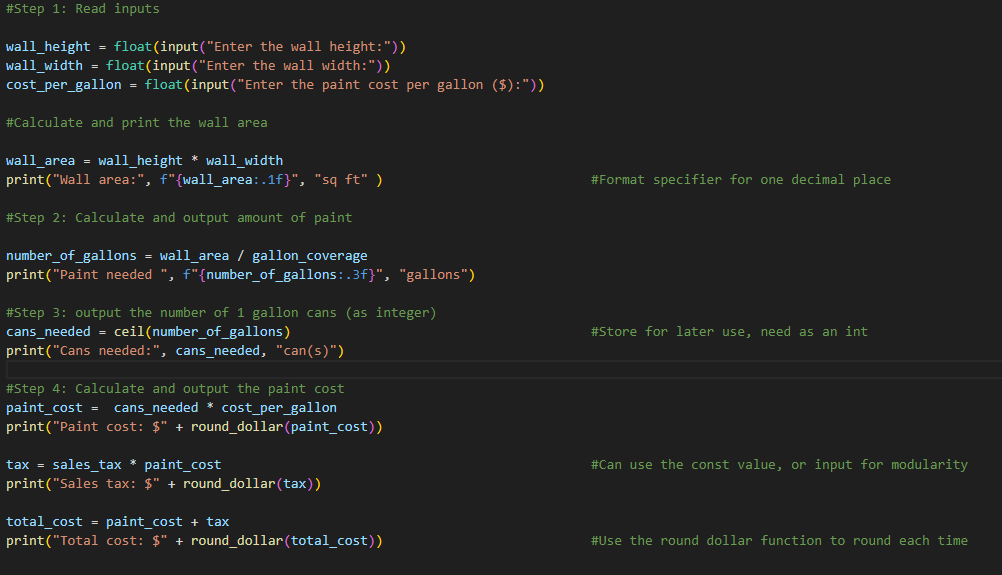
**2) Desktop Testing**- **Trace Table**  
Now that you have the algorithm, create a **Trace Table** to test your algorithm and see if it works on paper (refer to the trace table video in our module). Remember, your table should have columns for the variables and the expected output.

| **Line** | Height | Width | Cost Per Gallon | Area | # of Gallons | Cans needed | Paint cost | Tax | Total Cost | Sales Tax (const) | Sq ft per gallon (const) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 33 | - | - | - | - | - | - | - | - | - | - | 350 |
| 34 | - | - | - | - | - | - | - | - | - | .07 | 350 |
| 38 | 8.0 | - | - | - | - | - | - | - | - | .07 | 350 |
| 39 | 8.0 | 8.0 | - | - | - | - | - | - | - | .07 | 350 |
| 40 | 8.0 | 8.0 | 49.20 | - | - | - | - | - | - | .07 | 350 |
| 44 | 8.0 | 8.0 | 49.20 | 64.0 |  |  |  |  |  | .07 | 350 |
| 49 | 8.0 | 8.0 | 49.20 | 64.0 | .18285714 | - | - | - | - | .07 | 350 |
| 50 | 8.0 | 8.0 | 49.20 | 64.0 | .1828 | 1 | - | - | - | .07 | 350 |
| 57 | 8.0 | 8.0 | 49.20 | 64.0 | .1828 | 1 | 49.20 | - | - | .07 | 350 |
| 60 | 8.0 | 8.0 | 49.20 | 64.0 | .1828 | 1 | 49.20 | 3.444 | - | .07 | 350 |
| 63 | 8.0 | 8.0 | 49.20 | 64.0 | .1828 | 1 | 49.20 | 3.444 | 52.644 | .07 | 350 |

**3) Implementing the Algorithm in Python**

**Exercise 2.2**  
This is where you will translate the algorithm to Python.  Write a program called Paint.cpp that is designed based on the algorithm that is given in the previous part. Make sure to write descriptive comments for each step.

****

****

**4) Test the Program**   
Test each step and provide the output in this step. Use the test data provided in each step.

