Trees

1. Inputs: price per pound, number of pounds

Output: Cost

Error: negative pounds, negative price, $0 price per pound,

Algorithm:

```
    READ PricePound
    READ Pound
IF PricePound <= 0
    DISPLAY error – positive integers only
ELSE IF Pound < 0
    DISPLAY error – positive integers only
ELSE
    COMPUTE Cost as (PricePound * Pound)
    DISPLAY Cost
ENDIF
```

Minimum operations: 4

Maximum: 6

Test cases: [-1, 5][error], [0,5][error], [2,-3][error]

2. Inputs: Price per gallon, number of gallons

Output: Cost

Error: negative Price per gallon, negative number of gallons, $0 per gallon, 0 number of gallons

Algorithm:

```
    READ PriceGallon
    READ Gallons
    READ “Cash” or “Card”?
IF PriceGallon <= 0
```

```
        DISPLAY error – positive integers only
ELSE IF Gallons <= 0
        DISPLAY error – positive integers only
ELSE IF Card
        COMPUTE cost as (PriceGallon * Gallons)*(1.1)
        DISPLAY cost
ELSE Cash
        COMPUTE cost as (PriceGallon * Gallons)
        DISPLAY cost
ENDIF
```

Minimum operations: 5

Maximum operations: 8

Test Cases: [0,5,Card][Error], [5,0,Card][Error], [5,0, Cash][Error], [0,5, Cash][Error], [3,3, cash][9], [3,3,card][9.9]

3. Inputs: Type of student, number of credits

Output: cost

Error: two types for one student, negative credits

Algorithm:

```
        READ studentType
        READ credits
IF studentType = university employee
        Display $6.87
ELSE IF studentType = state employee
        COMPUTE cost as 6.87 + (credits * 20 * 0.10)
        DISPLAY cost
ELSE studentType
        COMPUTE cost as (6.87 + (Credits * 20) ) * 1.03
        DISPLAY cost
ENDIF
```

Minimum: 4

Maximum: 6

Test cases: [university, state][error], [university, -1][error], [university, 5][6.87], [state, 1][27.47], [state, -1][error], [normal/other, -1][error], [normal/other, 0][7.08]

4. input: number of zoos, number of giraffes, lions, and snakes in each zoo

Output: total number of giraffes, lions, and snakes

Error: none, assumed all inputs are positive

Algorithm:

```
        READ numbersZoo
        SET numZoo = 1
        SET prevGiraffes = 0
        SET prevLions = 0
        SET prevSnakes = 0
While (numZoo <= numbersZoo)
        DISPLAY numZoo
        READ giraffes
        READ lions
        READ snakes
        COMPUE prevGiraffes equal to giraffes + prevGiraffes
        COMPUE prevLions equal to giraffes + prevLions
        COMPUTE prevSnakes equal to snakes + prevSnakes
        COMPUTE numZoo + 1
ENDWHILE
DISPLAY prevGiraffes, prevLions, prevSnakes
```

Minimum: 16

Maximum: infinite

Test Cases: [1, 2,2,2] [2, 2, 2], [2, 1,0,1, 1,1,1] [2,1,2]