CMSC 21

Fundamentals of Programming

OVERVIEW

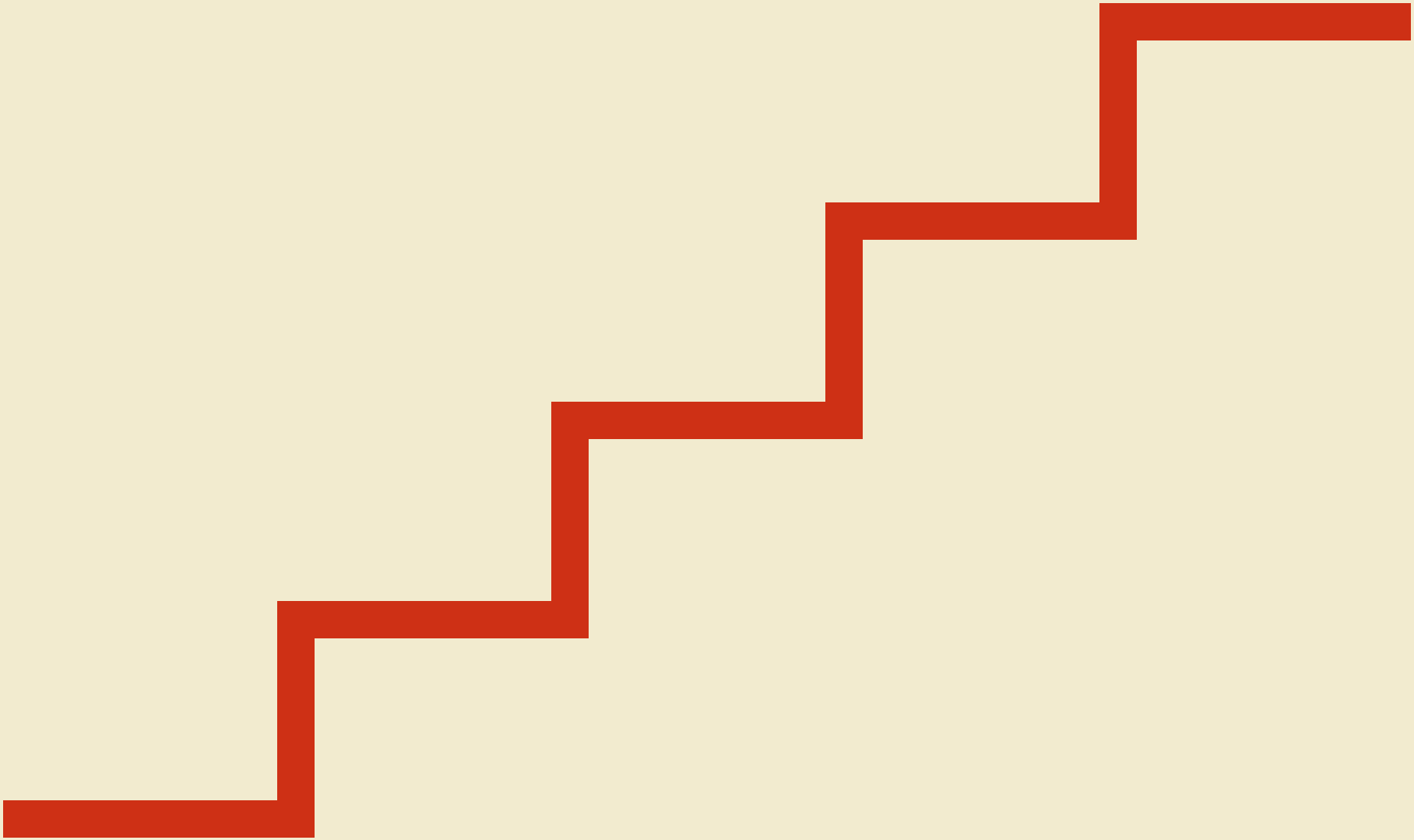
To learn the Program Development Process

Via Structured Programming

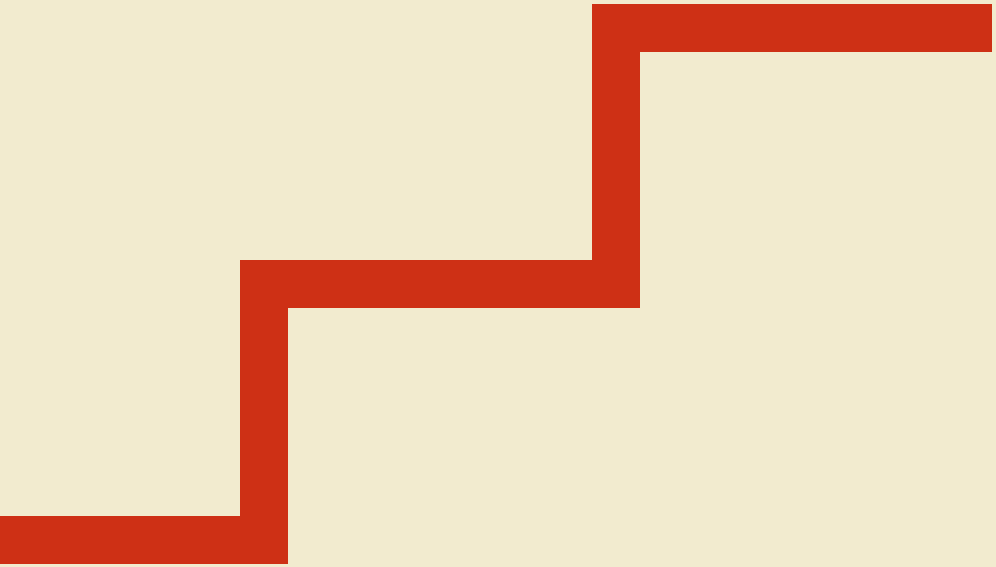
"Resist the temptation to code."

- Old programming proverb

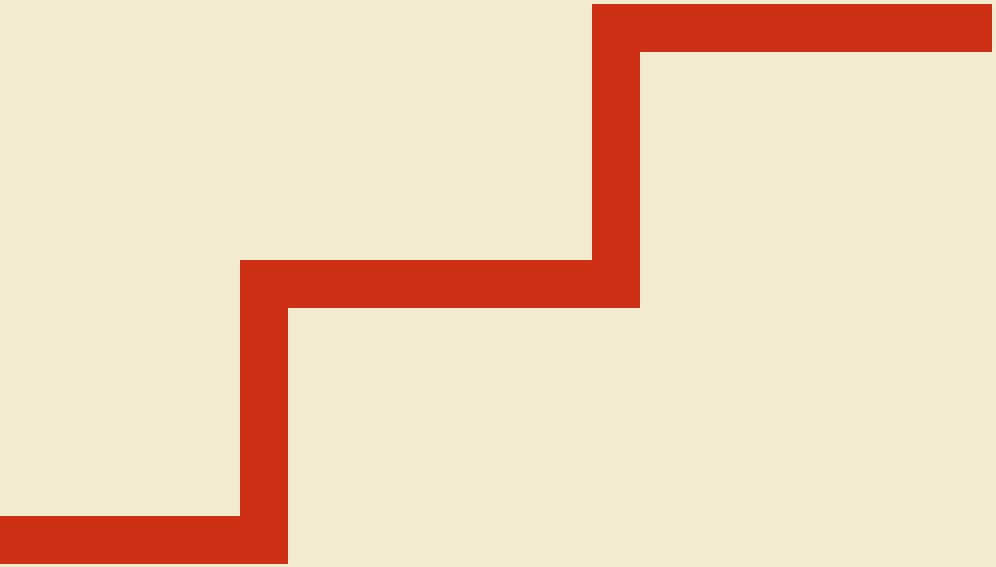
THE PROGRAM DEVELOPMENT PROCESS



BETTER PROGRAMMING



A PROPER SOLUTION



STEPS IN THE PROGRAM DEVELOPMENT PROCESS

STRUCTURED WALKTHROUGH

STEPWISE REFINEMENT

MODULAR DESIGN

BOTTOM-UP CODING

TESTING

DOCUMENTATION

1

STRUCTURED WALKTHROUGH

1

aka

REQUIREMENT ANALYSIS

1

Main ideas:

1

- *Understanding the problem through its*

SPECIFICATIONS

1

- *Involves*

ASKING QUESTIONS

to the client

1

- *May require several*

DISCUSSIONS

to come up with clear requirements

1

•ENUMERATE

I/O specifications, processes, and constraints

1

- *In the end, problem specifications must be:*

VERY PRECISE

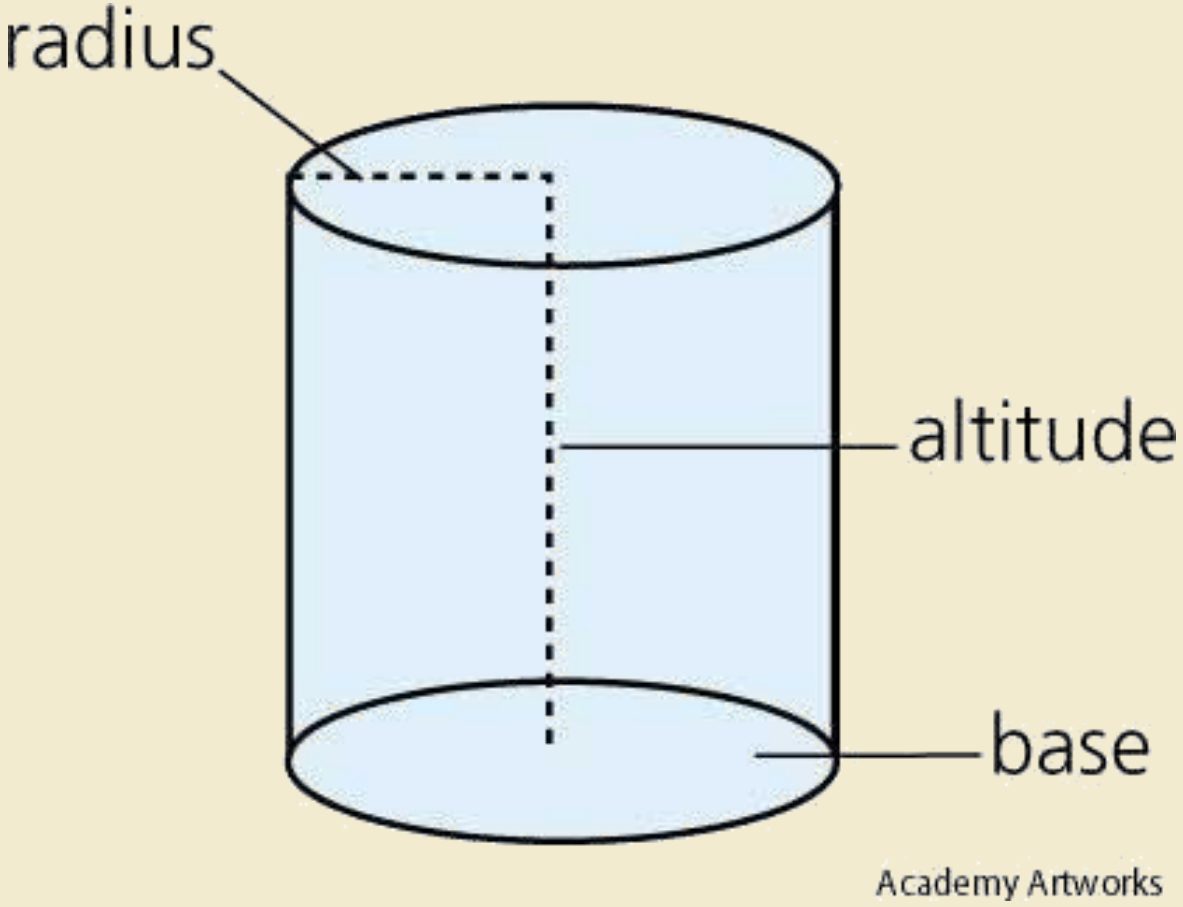
1

and

UNDERSTANDABLE BY THE PROGRAMMER

(in programming terms)

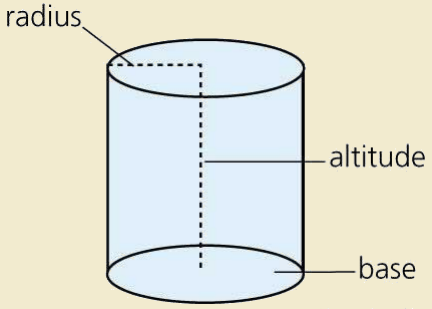
1



Ex. Get the surface area of a cylinder.

1

Suppose that we don't know the formula for computing the surface area of a cylinder.

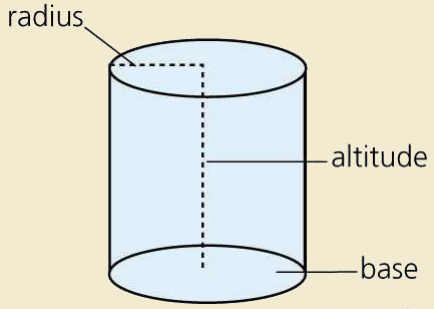


Academy Artworks

1

And that all we know are the following:

- We can compute for the area of a circle.
- We can compute for the area of a rectangle.
- The radius and the height.



2

STEPWISE REFINEMENT

2

aka

TOP-DOWN DESIGN

2

aka

DIVIDE AND CONQUER

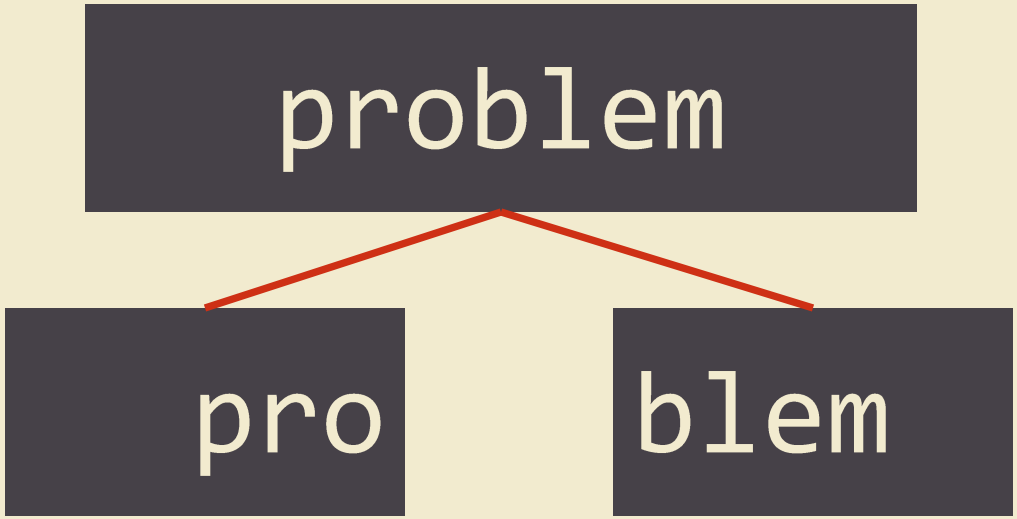
2

BREAK THE PROBLEM

```
problem
```

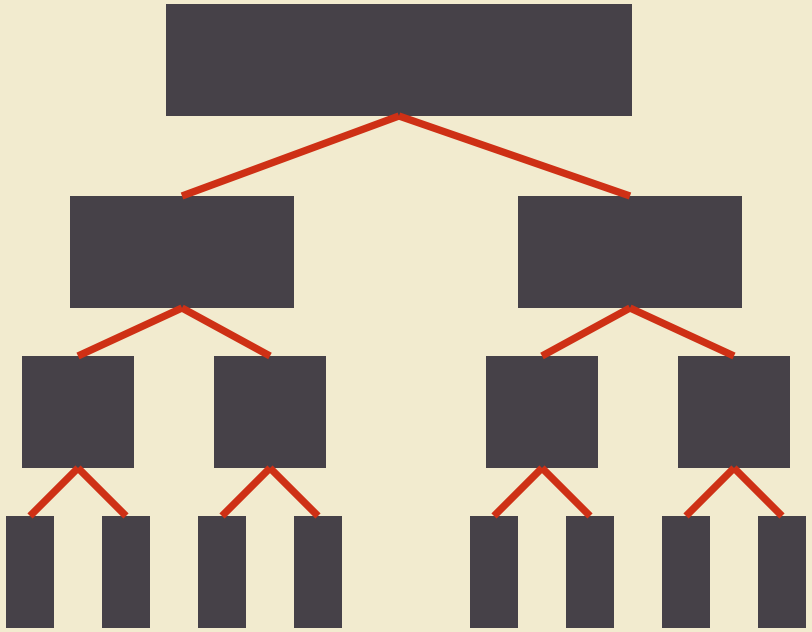
2

INTO SMALLER PARTS



2

UNTIL EACH PART CAN BE EASILY DONE



2

"Inside every large problem is a small problem struggling to get out."

HOARE'S LAW OF LARGE PROBLEMS

2

- *Generally, divide the job into three parts:*

INPUT
PROCESS
OUTPUT

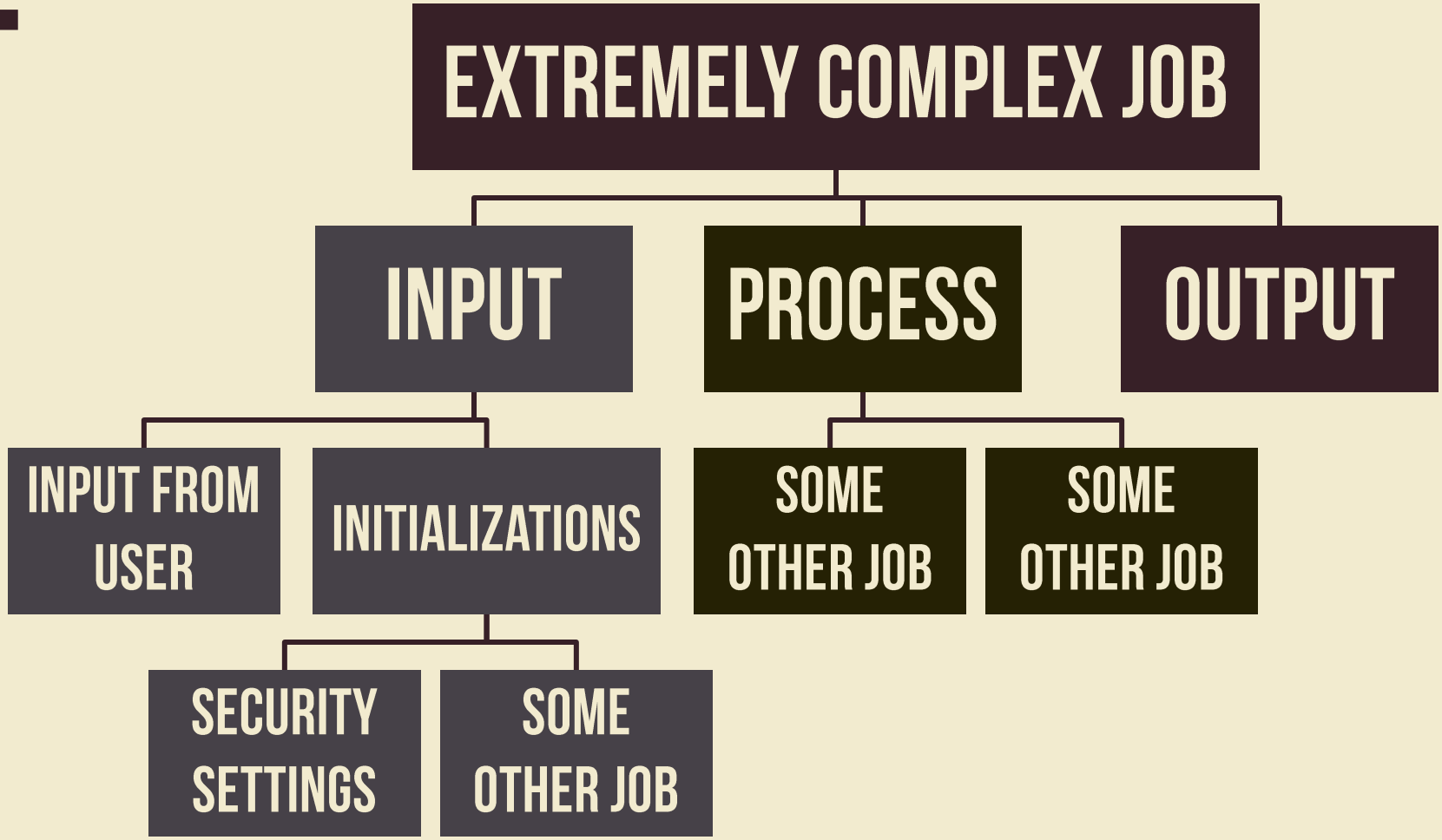
2

- *Draw a tree and refine each job level by level.*
- *Use pseudo-code to describe each job*

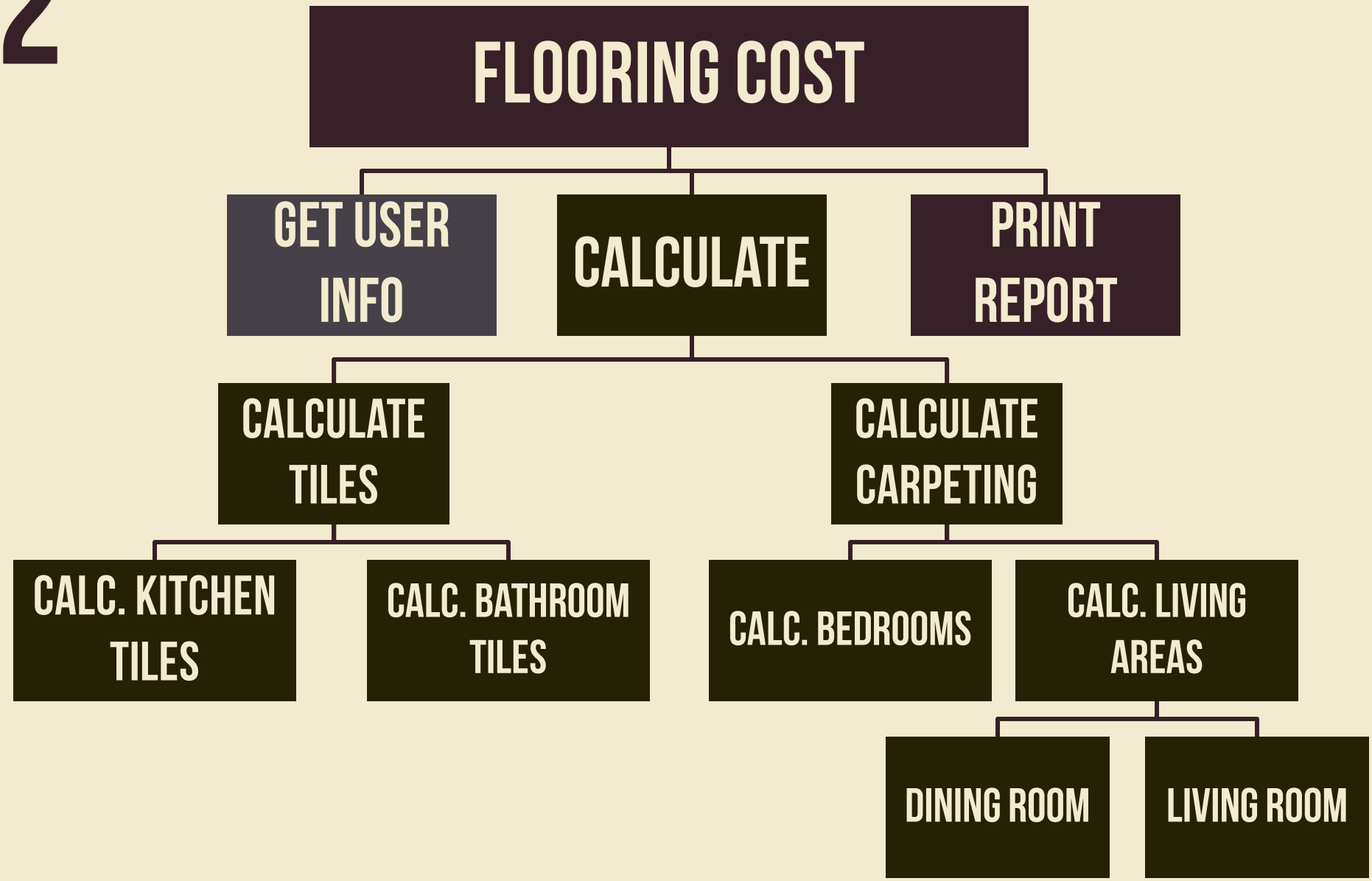
2

- *Decision on data structure is delayed as much as possible*

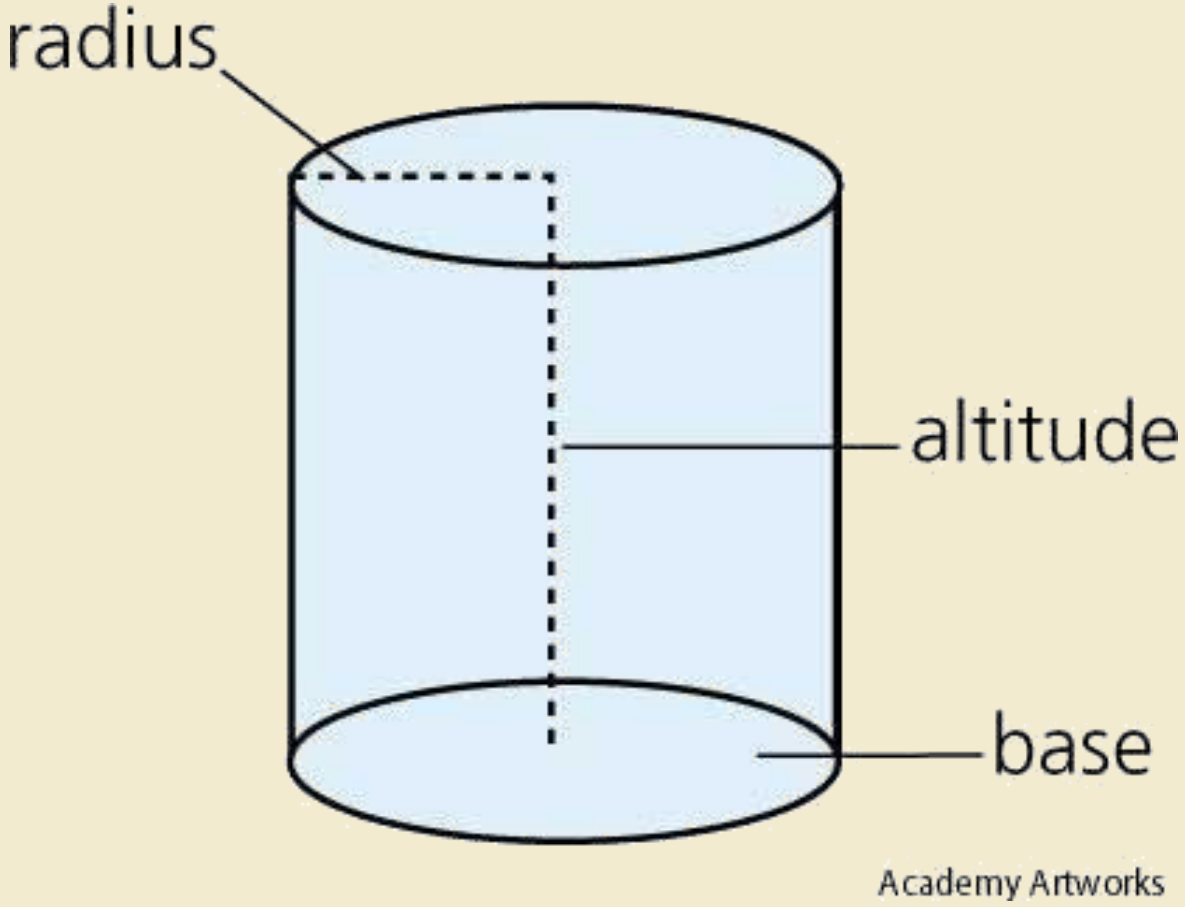
2


EXTREMELY COMPLEX JOB
INPUT
PROCESS
OUTPUT
INPUT FROM USER
INITIALIZATIONS
SOME OTHER JOB
SOME OTHER JOB
SECURITY SETTINGS
SOME OTHER JOB

2

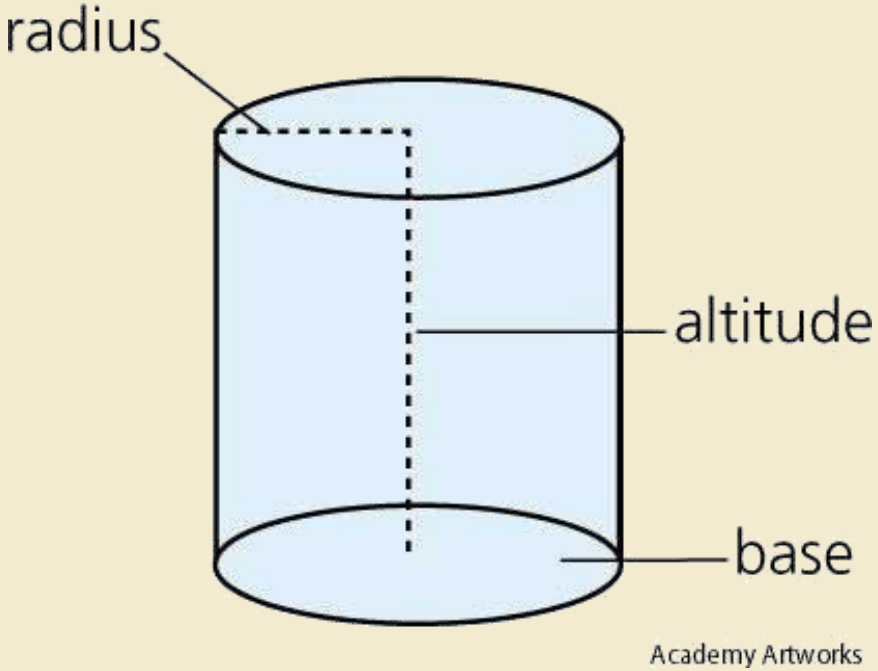

FLOORING COST
GET USER INFO
CALCULATE
PRINT REPORT
CALCULATE TILES
CALCULATE CARPETING
CALC. KITCHEN TILES
CALC. BATHROOM TILES
CALC. BEDROOMS
CALC. LIVING AREAS
DINING ROOM
LIVING ROOM

2



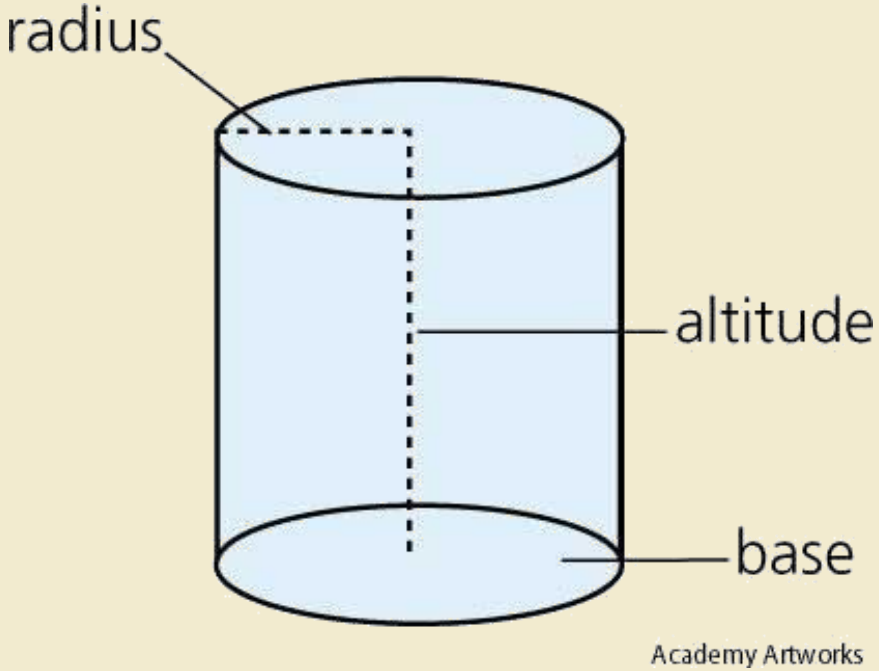
Let’s divide the problem into three parts:

2



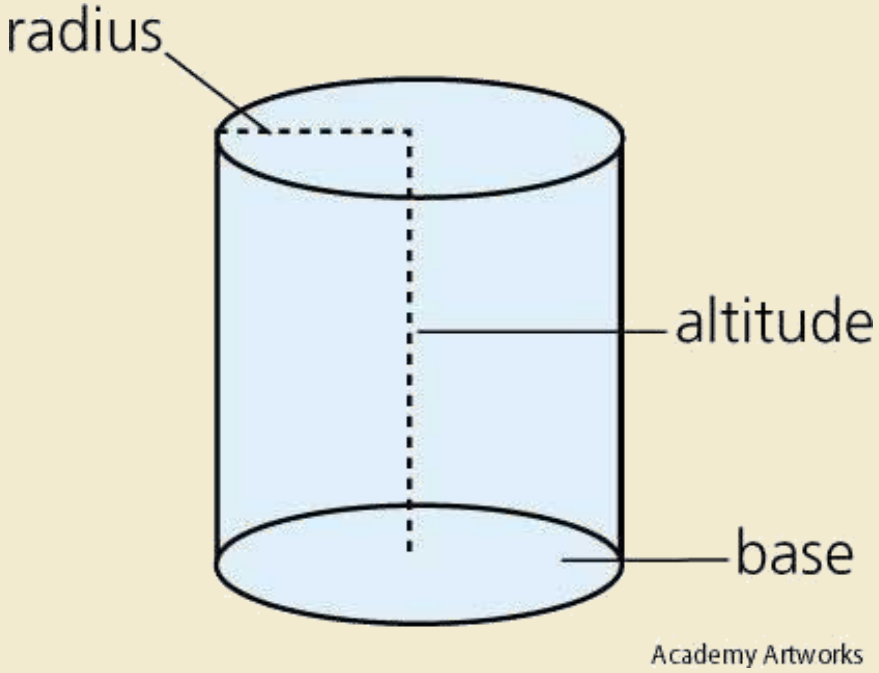
Input: Radius and Height

2



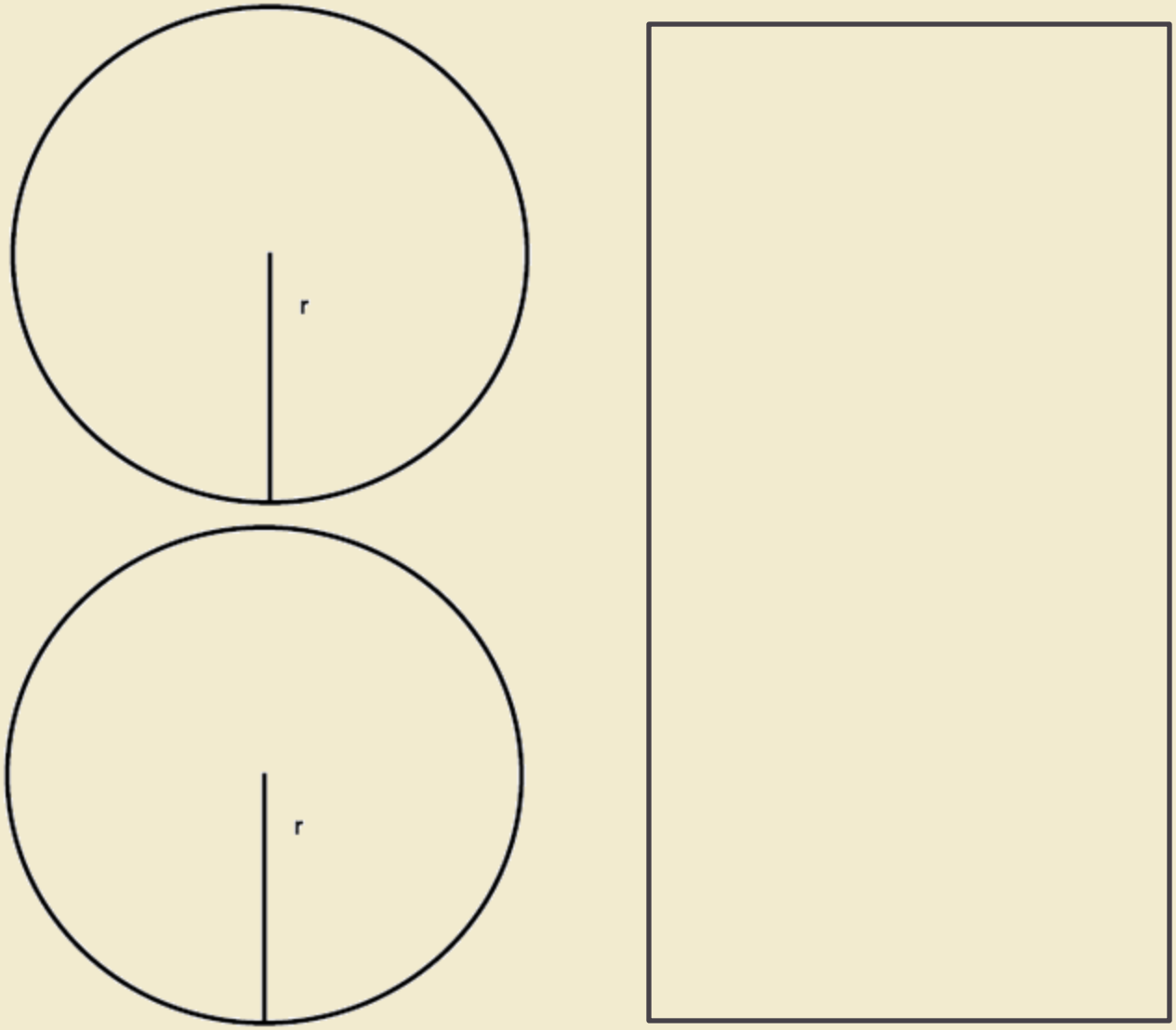
Process: Compute for the area of two circles and the area of a rectangle

2



Output: The surface area of the cylinder

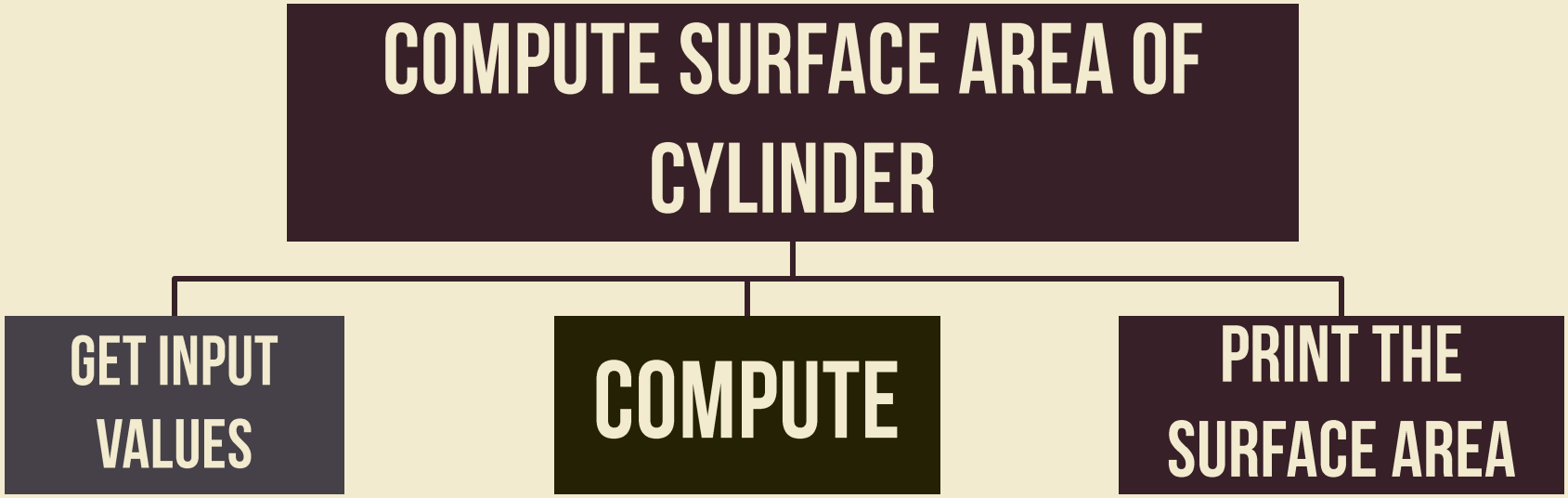
2



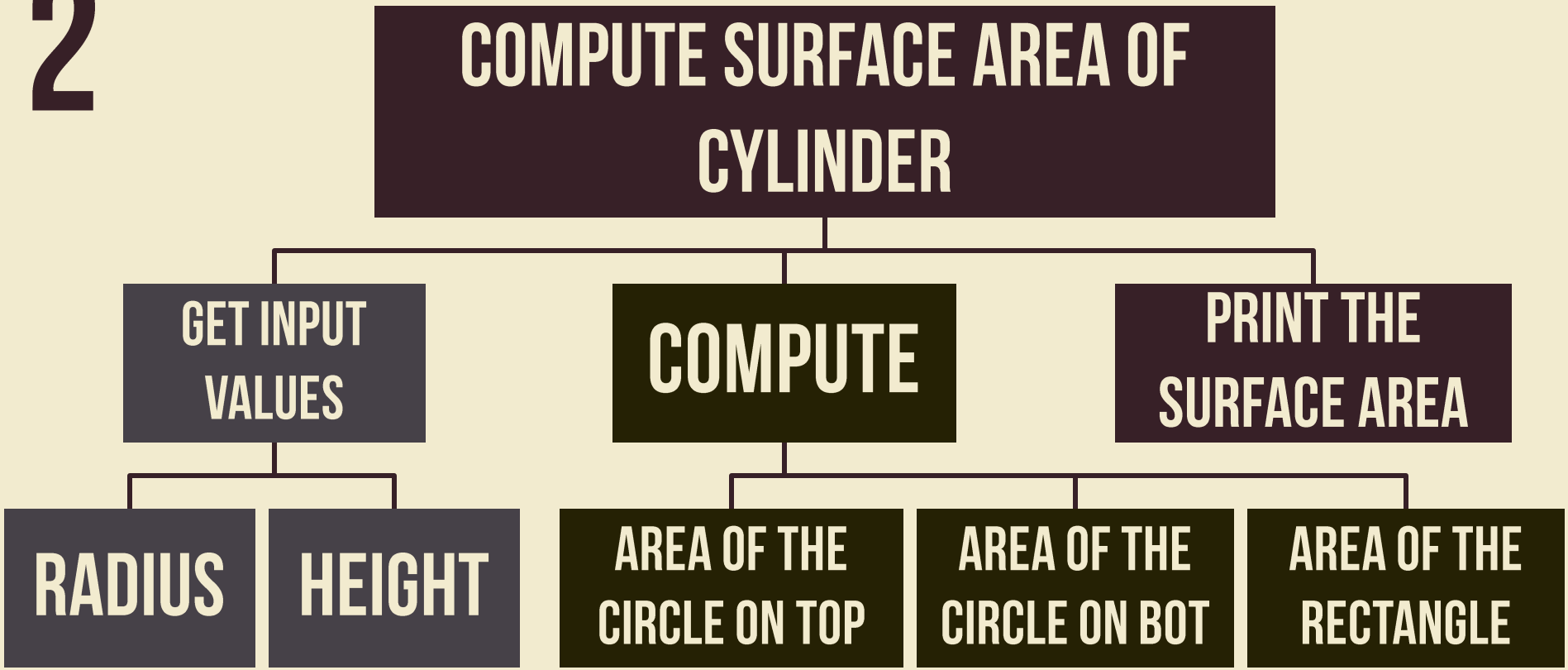
2

COMPUTE SURFACE AREA OF CYLINDER

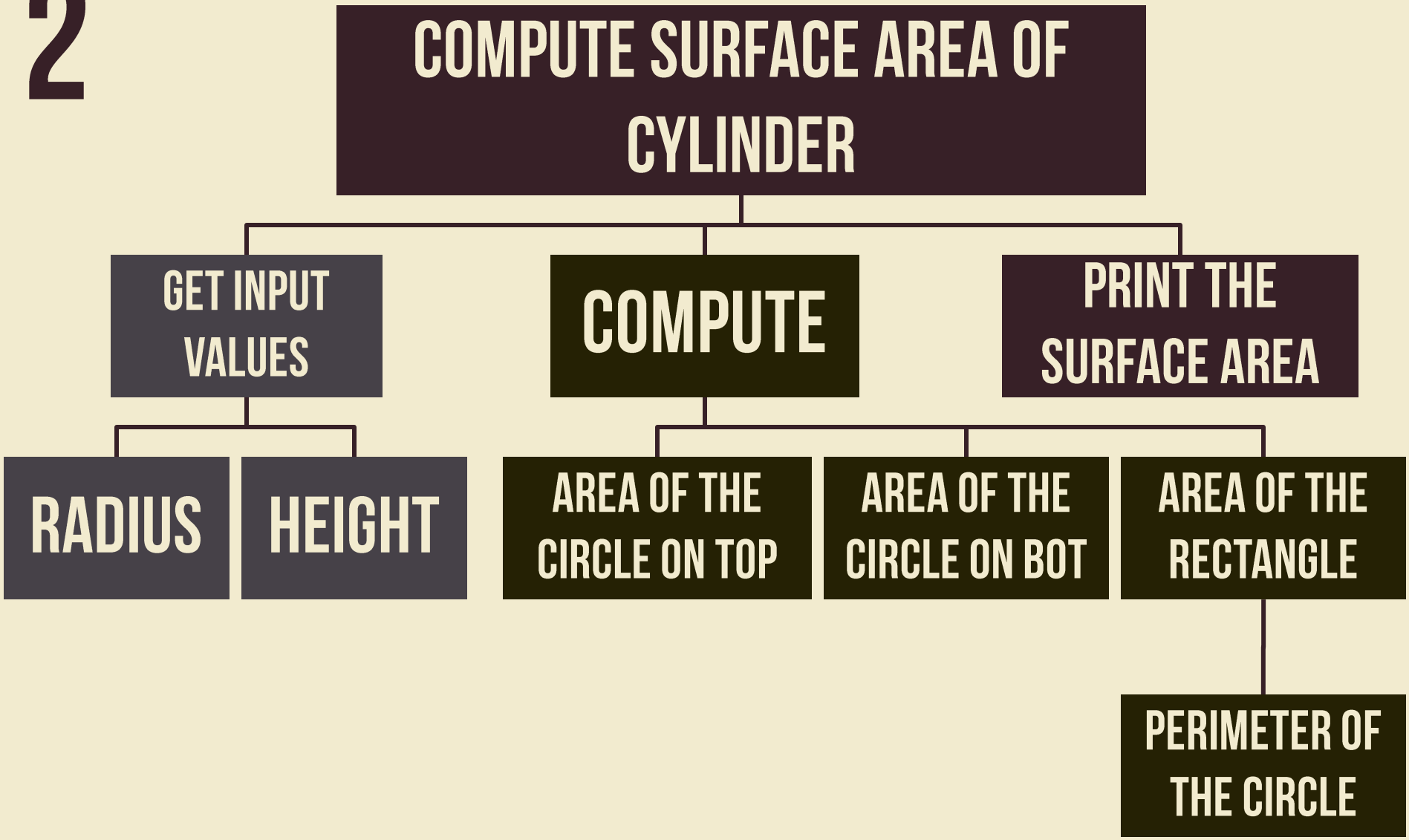
2


COMPUTE SURFACE AREA OF CYLINDER
GET INPUT VALUES
COMPUTE
PRINT THE SURFACE AREA

2


COMPUTE SURFACE AREA OF CYLINDER
GET INPUT VALUES
COMPUTE
PRINT THE SURFACE AREA
RADIUS
HEIGHT
AREA OF THE CIRCLE ON TOP
AREA OF THE CIRCLE ON BOT
AREA OF THE RECTANGLE

2


COMPUTE SURFACE AREA OF CYLINDER
GET INPUT VALUES
COMPUTE
PRINT THE SURFACE AREA
RADIUS
HEIGHT
AREA OF THE CIRCLE ON TOP
AREA OF THE CIRCLE ON BOT
AREA OF THE RECTANGLE
PERIMETER OF THE CIRCLE

3

MODULAR DESIGN

3

- *Define the*
  INPUT AND OUTPUT SPECIFICATIONS
  of each module

3

- DESIGN AN ALGORITHM FOR EACH MODULE

using flowcharts and pseudocodes

3

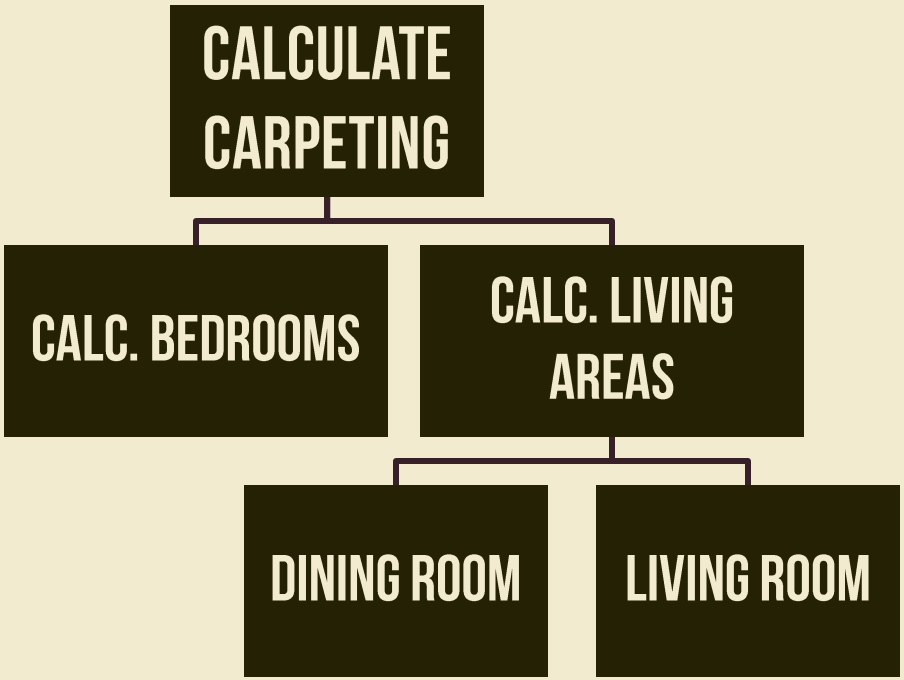
- *Each module is*
  A SELF-CONTAINED BLOCK

3

- *Codes and data structures ought to be*

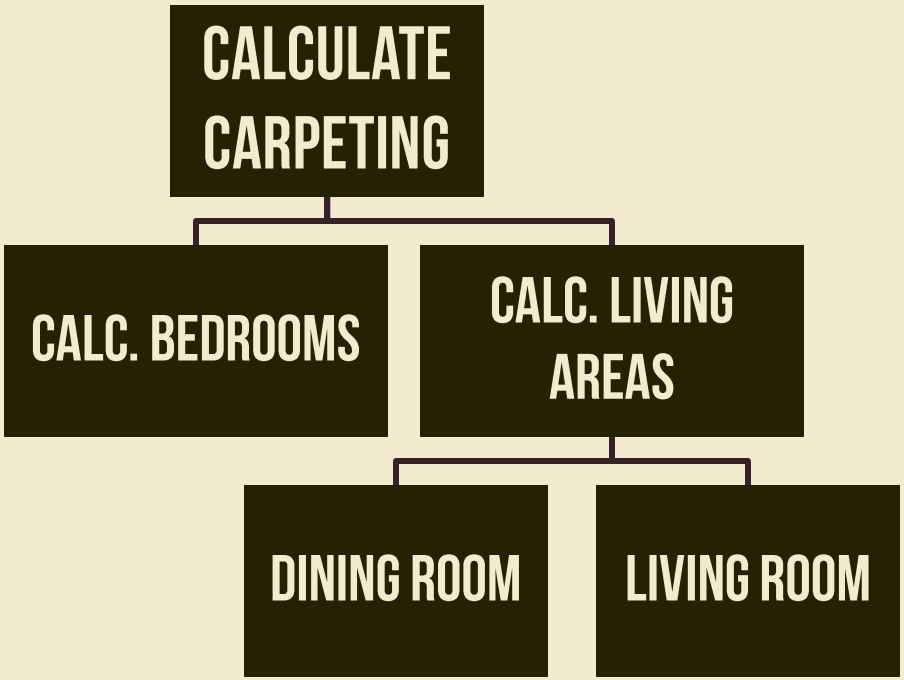
REUSABLE

3



“Calculate Living Areas” is the immediate ancestor of “Dining Room” and “Living Room”.

3




```
Flooring cost for Living Areas =
Flooring cost for Dining Rm +
Flooring cost for Family Rm
```

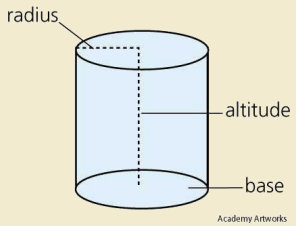
3

- *Each node in the hierarchy is a*
  MODULE

3

GetRadius – acquires the radius

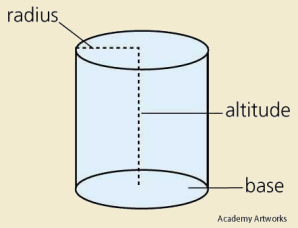
GetHeight – acquires the height



3

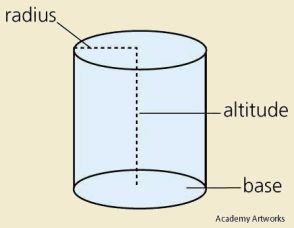
`ComputeAreaOfCircle` – calculates the area of a circle given the radius

`ComputeAreaOfRectangle` – calculates the area of a rectangle given the height and width



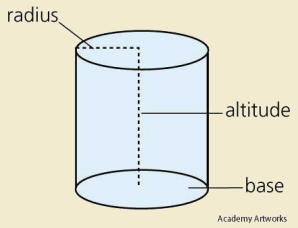
3

`ComputePerimeterOfCircle` – calculates the perimeter of a circle



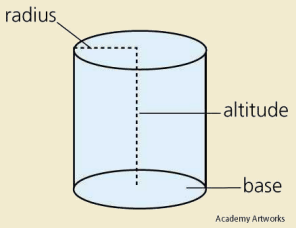
3

`ComputeSurfaceAreaOfCylinder` – calculates the surface area given the area of the circles and the area of the rectangle



3

`PrintSurfaceArea` – outputs the result



4

BOTTOM-UP CODING

4

Code the

SIMPLEST FUNCTIONS FIRST

4

Simple functions
CAN BE TESTED INDEPENDENTLY

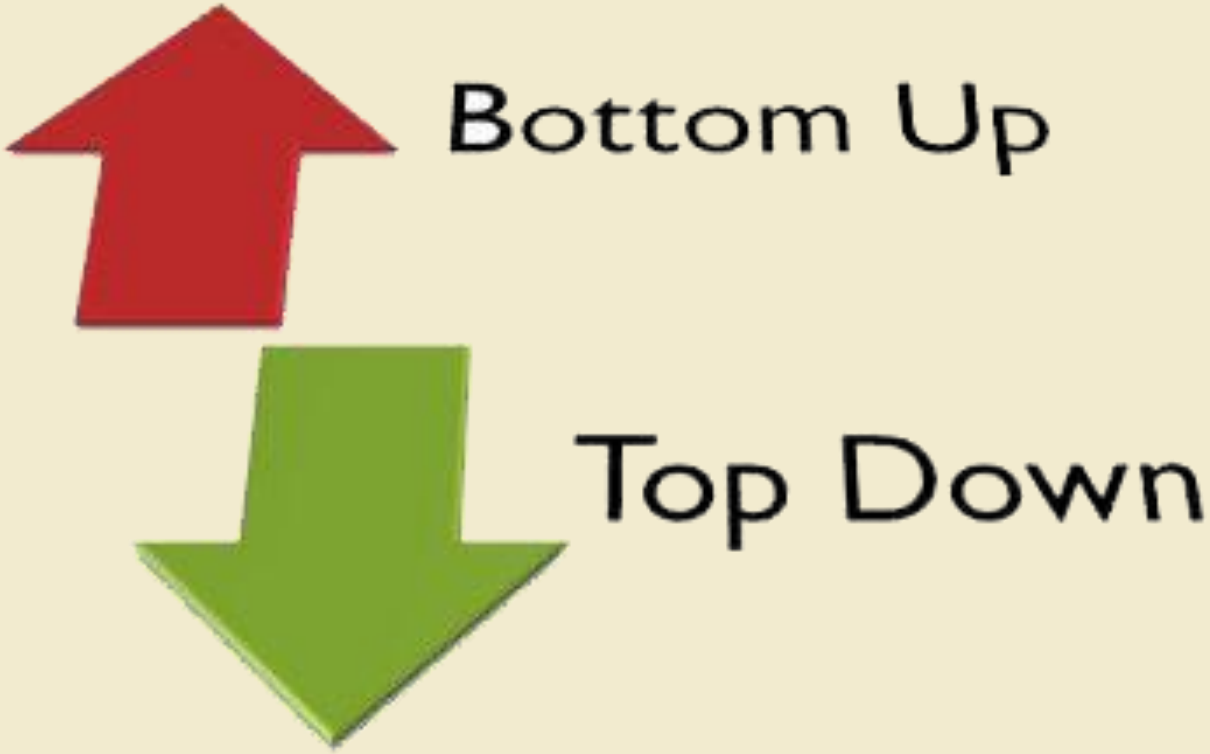
4

Modules can also be

DISTRIBUTED

among team members

4



5

TESTING

5

Types:

TESTING USING STUBS

WHITE-BOX TESTING

BLACK-BOX TESTING

5

TESTING USING STUBS

WHITE-BOX TESTING

BLACK-BOX TESTING

5 TESTING USING STUBS

- Field testing before deployment.

5 TESTING USING STUBS

- Stubs

  - A human tester to emulate functions of unfinished modules

5 TESTING USING STUBS

- Stubs

  - Tester acts on inputs and enters the correct output for the finished modules of the program

5 TESTING USING STUBS

- Programmers can test their work even when other programmers are not yet finished.

5

TESTING USING STUBS

WHITE-BOX TESTING

BLACK-BOX TESTING

5 WHITE-BOX TESTING

- Assumption: The tester knows *everything* about the program.

5 WHITE-BOX TESTING

- Given a set of inputs, the tester must know what output to expect.

5 WHITE-BOX TESTING

- Practical for small systems only.

5

TESTING USING STUBS

WHITE-BOX TESTING

BLACK-BOX TESTING

5 BLACK-BOX TESTING

- Assumption: The tester knows *nothing* about the program.

5 BLACK-BOX TESTING

- The result for a set of inputs should not be known beforehand.

5 BLACK-BOX TESTING

- Does the program give a reasonable output given a set of test inputs?

6

DOCUMENTATION

6

GENERATED

along with the program

6

DESCRIBES

the whole program and each module

6

HELPS IN IDENTIFYING

which modules are needed to be modified when the requirements change

6

USEFUL

since a programmer forgets his code after some time

PROBLEM 1.

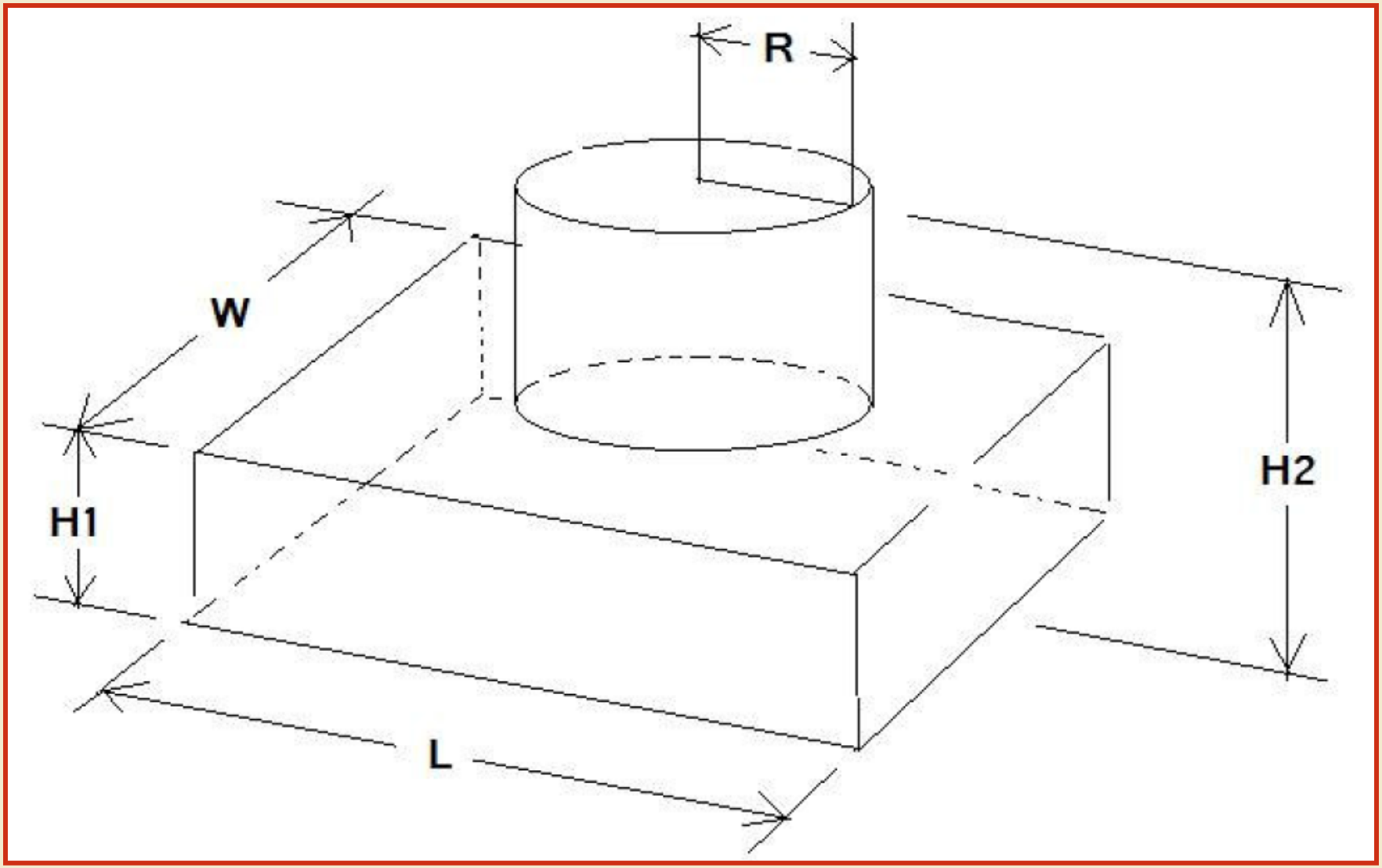
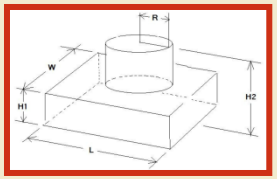


Fig. 1

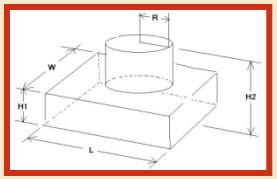
PROBLEM 1.



Create a program that computes for the **surface area** and **volume** of the 3D geometric figure (in Fig. 1).

"Known" formulas: Area of Circle, Area of Rectangle, and Circumference of Circle

PROBLEM 1.



Quiz: Solve the problem (surface area only) up to the Modular Design step.