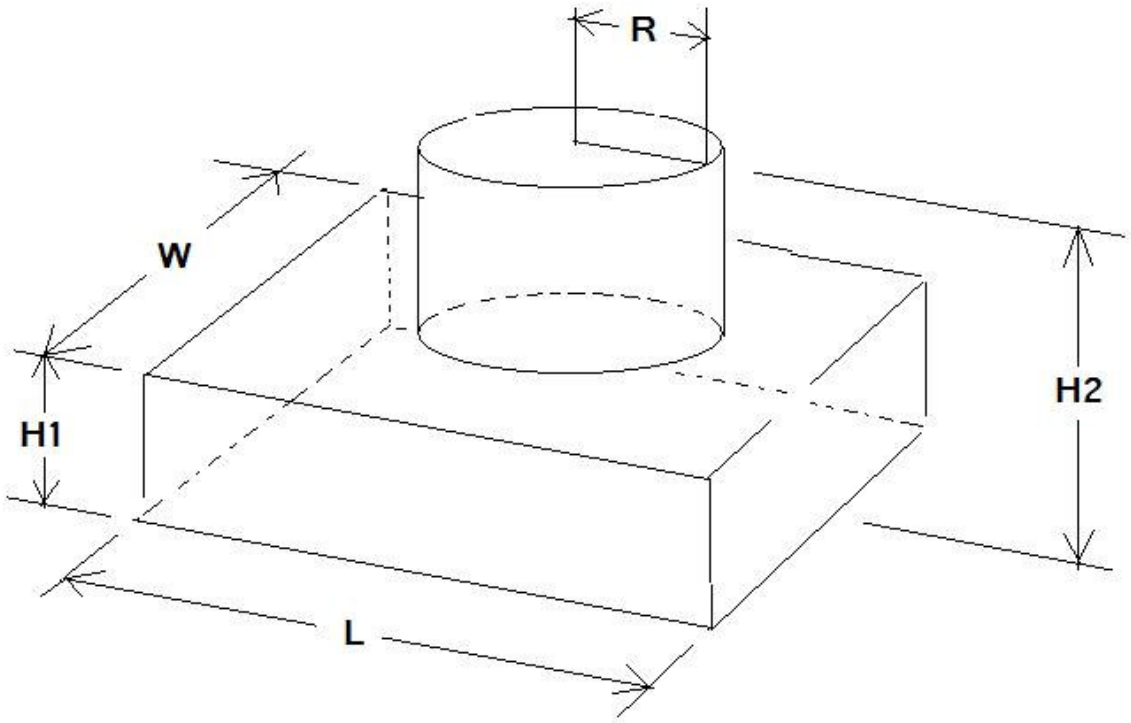
CMSC 21: Laboratory Exercise on Structured Programming

Outline of Steps:

1. Structured Walk-through (Requirement Analysis)
2. Step-wise refinement (Top-down Design)
3. Modular Design
4. Bottom-up Coding (Implementation)
5. Testing using Stubs
6. White Box & Black Box Testing
7. Structured Documentation

A. Structured Walk-through (Req't Analysis)

Problem: Create a program that computes for the surface area and volume of the 3D geometric figure below:



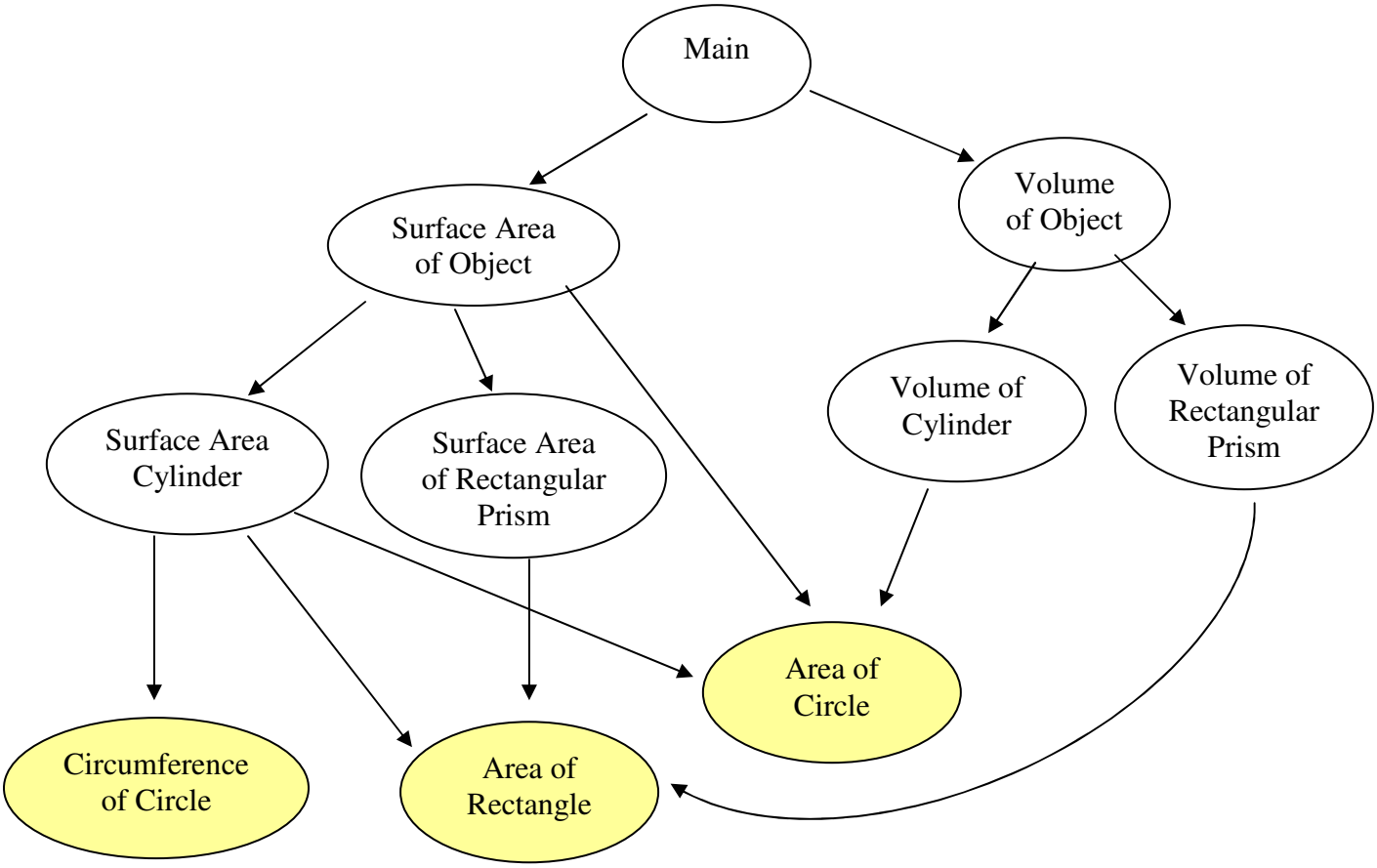
Input: R, L, W, H1, H2
Output: Surface Area, Volume

Constraints:

1. Ready-made formula that should NOT be used (should be derived instead)
    a. Surface Area of Cylinder
    b. Volume of Cylinder
    c. Surface Area of Rectangular Prism
    d. Volume of Rectangular Prism

2. Ready-made formula that are allowed to be used
    a. Area of Circle
    b. Area of Rectangle
    c. Circumference of Circle (to be guessed by the designer)

B. Step-wise refinement (Top-down Design; Interaction Diagram)



C. Modular Design (I/O Specification, Description, Algo)

1. Module: Main
    - Description: Given R, L, W, H1, H2 values, display on the screen the surface area & volume of the 3D object
    - Input: -
    - Output: -
    - Algo: read values from the keyboard, call Modules 2 & 3

2. Module: Surface Area of Object
    - Description: (implied)
    - Input: R, L, W, H1, H2
    - Output: Surface Area

3. Module: Volume of Object
    - Description: (implied)
    - Input: R, L, W, H1, H2
    - Output: Volume

4. Module: Surface Area of Cylinder
    - Description: (implied)
    - Input: radius, height
    - Output: Surface Area

5. Module: Surface Area of Rectangular Prism
    - Description: (implied)
    - Input: length, width, height
    - Output: Surface Area

6. Module: Volume of Cylinder
    - Description: (implied)
    - Input: radius, height
    - Output: volume

7. Module: Volume of Rectangular Prism
    - Description: (implied)
    - Input: length, width, height
    - Output: volume

8. Module: Circumference of Circle
    - Description: (implied)
    - Input: radius
    - Output: circumference

9. Module: Area of Rectangle
    - Description: (implied)
    - Input: length, width
    - Output: area

10. Module: Area of Circle
    - Description: (implied)
    - Input: radius
    - Output: area

D. Bottom-up Coding

Constraint: No global variable

See source code:

1. write prototypes (bottom-up)
2. define modules (bottom-up)

E. Testing using Stubs (hands-on)
F. White Box/Black Box Testing (hands-on)

G. Structured Documentation
- finalize interaction diagram, module description, source code document

----

Comments: improvement
- reusability of Area of Rectangle Module (note: called more than once, w/ different parameters)
- (drawback) Area of Circle Module, called more than once, yet w/ same parameter (hence, not a manifestation of code reuse): resolution – call once, then pass result to needing module (advance topic, due to the use of pointers when a module needs to return more than one value; use of global variable isn't a preferred solution)
- Implementation may be combination of top-down & bottom up strategy (stubbing is crucial here)
- The steps above may not represent the standard (if any) SoftEng'g approach