

1. Consider the integer factorial function  $m = n!$ , defined by the conditional formula:  
if  $n = 0$ ,  $m = 1$ ; and if  $n > 0$ ,  $m = n \cdot (n-1)!$

This is a partial function \_\_T/F.

Counting the number of combinations of  $k$  objects chosen from  $n$  objects is denoted  $C(n,k)$  and spoken as  $n$  choose  $k$ . Now consider this second integer function  $C(n,k)$ :  $\binom{n}{k}$  and has the formula:

$$C(n,k) = c = n! / (k! (n-k)!)$$

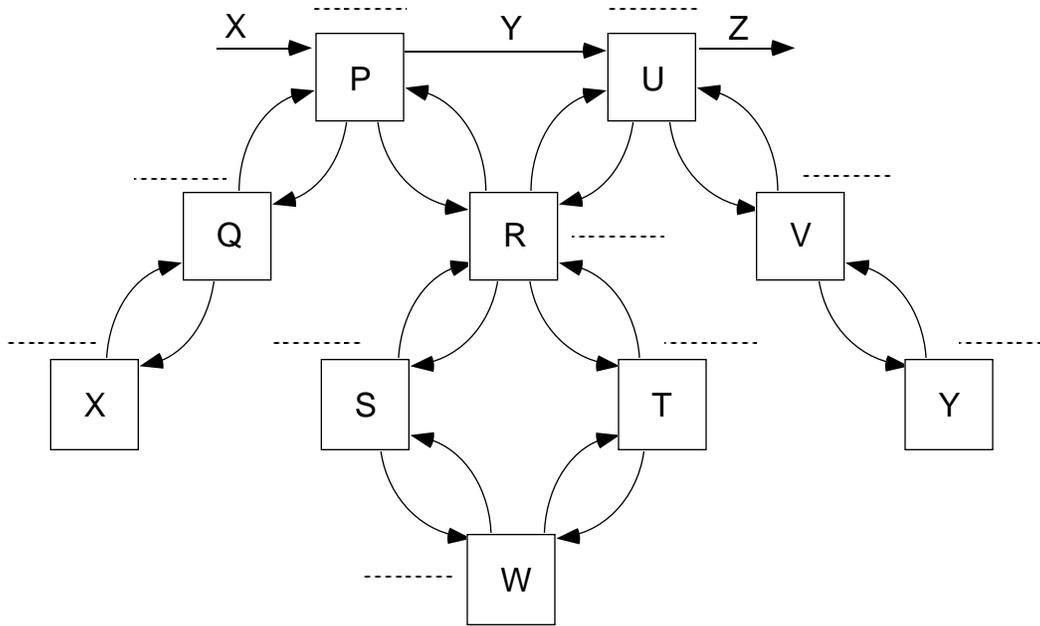
What are the function's

- Domain?
- Range?
- Rule?
- This is a total function \_\_T/F.

Suppose a procedure is to implement this function. What are/is its:

- Data types?
- Variables?
- Data space?
- Data states (give four samples)? { ... }  
(be sure to indicate the ordering)

2. In the context model shown below identify the relationships by naming the relationships (fill in the blank next to each box) and give an equivalent representation which is expressive of the embedding (or nesting) described in the context model.



3. Answer the following two part problem. (a) What is a “wicked” problem?

(b) List three of the four distinguishing properties of a wicked problem.

4. Define [based on Witts discussion] and be specific:

(a) Abstraction

(b) Process abstraction

(c) Data abstraction

5. Give an examples (at least X) of each of the four types of design representation:

a. Structural (X=1):

b. Behavioral (X=1):

c. Functional (X=2):

d. Data-modeling (X=2):

6. Describe the three components of a design method.

(a)

(b)

(c)

7. Describe the 4 different broad categories of design methods we discussed in class.

**Hint:** Describe how the problem is partitioned (i.e., functionally, structurally, data, etc., and be sure to discuss how stepwise refinement / iterative techniques fit the picture). Other concepts which should be discussed include templates, design experience, design patterns, data-directed and function-directed.

(a) Decompositional methods:

(b) Compositional methods:

(c) Organizational methods:

(d) Template-based methods:

8. Describe the five steps of the Jackson Structured Programming design method.

- a. Draw a ...
- b. Merge these to form ...
- c. List the program ...
- d. Convert the program to ...
- e. Add the conditions ...

9. Answer the following two part question:

**a.** Describe 3 non-trivial differences between Jackson Structured Programming and SSA/SD by filling the blanks (the size and number of blanks is a hint to the wordings).

JSP is a \_\_\_\_\_ method and SSA/SD is a \_\_\_\_\_ method.

JSP is suited for designing \_\_\_\_\_ with a single sequential process, while SSA/SD can be used to design \_\_\_\_\_.

JSP is \_\_\_\_\_-directed, reflected by the use of the \_\_\_\_\_ Diagram to represent the data in early stages of design. SSA/SD is \_\_\_\_\_-directed, reflected by the use of the Data Flow Diagram to show the flow of data between \_\_\_\_\_.

JSP is a \_\_\_\_\_ method, while SSA/SD incorporates both \_\_\_\_\_ and \_\_\_\_\_.

**b.** Describe the *design representations* (not including data dictionary) used in SSA/SD by filling in the blanks.

Data Flow Diagrams are used in SSA to define the \_\_\_\_\_.

P-Spec - \_\_\_\_\_ representation used in SSA (and also used in SD) to describe the \_\_\_\_\_ in the system represented by \_\_\_\_\_ in the DFD.

Entity-Relationship Diagram - used in SSA to model the \_\_\_\_\_ between \_\_\_\_\_ elements.

Structure Chart - used in SD to represent the " \_\_\_\_\_ " design of the system.

10. The four design representations described in the text are structural, behavioral, functional, and data-modeling. For each part of this question, fill in the blank with the form that it best fits with (one of the four just mentioned).

- a. \_\_\_\_\_ concerned with the time aspects of a system.
- b. \_\_\_\_\_ critical when considering detailed design, and has to do with relationships between data items.
- c. \_\_\_\_\_ concerned with the transitions that occur between different states of a system.
- d. \_\_\_\_\_ to define what the system does.
- e. \_\_\_\_\_ concerned with the physical components of a system.
- f. \_\_\_\_\_ at the abstract level, models objects that generally closely correspond to objects in the real-world problem.

11. Describe the difference between the following sets of terms.

(a) Representation part and process part (of a software design method) by filling in the blanks:

The representation part provides a set of \_\_\_\_\_ forms that the designer can use for building \_\_\_\_\_ of the \_\_\_\_\_ and their ideas for its \_\_\_\_\_, and for describing the \_\_\_\_\_ features of the \_\_\_\_\_ to the eventual implementers.

The process part is concerned with describing \_\_\_\_\_ the necessary \_\_\_\_\_ between the \_\_\_\_\_ forms are to be organized, as well as with any \_\_\_\_\_ of their detail that might be required.

The representation part describes the \_\_\_\_\_ and the \_\_\_\_\_ part describes its construction.

(b) Perfective maintenance, adaptive maintenance, corrective maintenance

Perfective maintenance is concerned with \_\_\_ and system once it is operational, typically by providing new ...\_\_\_\_\_.

Adaptive maintenance is performed in order to meet ...\_\_\_\_\_

Corrective maintenance is performed to ....\_\_\_\_\_

(c) Coupling and Cohesion

Coupling is a measure of \_\_\_\_\_ and is concerned with identifying the forms of \_\_\_\_\_ and the "strength" of these \_\_\_\_\_.

Cohesion provides a measure of the extent to which the \_\_\_\_\_ within a \_\_\_\_\_ can be considered "functionally related."

*Coupling* deals with relations \_\_\_\_\_ modules while *cohesion* deals with \_\_\_\_\_ module.

12. Given the system requirements (below) for a vending machine, describe the system using a state transition diagram (hint: some details will need to be abstracted away). The vending machine (VMC) system requirements are as follows:

The VMC dispenses goods: (1) very large candy (VC) at 15¢, (2) large candy (LC) at 10¢, and (3) a small candy (SC) at 5¢. *The vending machine only deals in nickels, dimes and quarters.*

The VMC gives the proper change after a customer selection is made. The VMC must tabulate the amount being deposited. The following rule applies to the VMC's behavior:

The VMC remains idle until a customer or owner begins to interact with the VMC. When a selection button is depressed the VMC indicates the required amount needed:

If the full amount needed has been deposited then dispense the proper candy and display: Thank You!.

If an insufficient amount (possibly zero) has been deposited then display: remaining amount needed.

If an over amount has been deposited then dispense the proper candy and change and display: Thank You!.

Once the customer *deposits coins* makes a *selection* which matches the amount deposited (or more) the candy is dispensed and he/she will *retrieve the candy and change* (if applicable). The VMC owner *inserts a key* to stock the candy and the change and retrieves the profits from the money bin. Once completed these maintenance activities the owner will lock the VMC and *extract the key*. Hint: Use six different states and at least ten transitions. Selection: a button is selected / deselected.

13. Given the system requirements (below) for the elevator, provide the artifacts for both the JSP (the 2 different Structure diagrams of Steps 1 I/O and 2 Program Structure) and the SSA/SD (three items: context diagram, top-level DFD and P-spec) methods:

**Elevator system requirements:**

There are call buttons (up/down) which generate pickup calls on each of the  $n$ -floors. In each of the  $m$  cars, there are  $n$  destination call buttons. The elevator cars: go up / go down / stop / open door / close door. The cars themselves control the opening and closing of the doors according to the following rules:

- Open the door when the desired (destination/pickup) floor is stopped at.

- When opened the door stays open for ten time units.

- Do not squeeze any object / person in the door.

There is a controller that arbitrates the call / destination signals using the following rules:

- Do not send more than one car to a pickup call.

- Always stop at the destination calls that have not been already passed.

- Pickup calls are only serviced if they are on the way.

- Dispatch an idle car for an unassigned pickup call if no other car can get it (on its way).

14. True/False When assessing the extent of modular structuring in software:

- T/F Common environment coupling is moderately desirable.
- T/F Logical cohesion, where elements perform operations that are logically similar and which involve very different actions internally is moderately desirable.
- T/F One of the fundamental design principles described by Witt is: *Portability can be achieved by employing concrete context interfaces.*

Fill in the blanks:

- Verification: Are we building \_\_\_\_\_?
- Validation: Are we building \_\_\_\_\_?

15. Software Architecture includes a high level partitioning of software into subsystems and their specification as well as SEE/CRS (languages, standards, configuration management, traceability, etc.) True / False

16. Name three of the six problems (root causes) denoting the software crisis.

17. The three main views of the system include *structure*, *function* and \_\_\_\_\_.

18. Give the five principles of design (discussed in my slides) and a brief definition of each.

19. Give a diagram of SDUs that exhibits the idea of a public specification and hidden designs. Utilize the following characteristics:

- a) Behavior - a process abstraction summarizing the data transformations that will be performed.
- b) Interface - the name of the process and how it may be utilized (IN and OUT parameters)
- c) The design part of an SDU encapsulates all the details of implementation required to fulfill the specification.

20. Cohesion is a measure of how closely the parts of a component relate to each other. Match the term with the definition:

- 1) Coincidental cohesion \_\_\_\_\_ (a-g)
- 2) Logical association \_\_\_\_\_ (a-g)
- 3) Temporal cohesion \_\_\_\_\_ (a-g)
- 4) Procedural cohesion \_\_\_\_\_ (a-g)
- 5) Communication cohesion \_\_\_\_\_ (a-g)
- 6) Sequential cohesion \_\_\_\_\_ (a-g)
- 7) Functional cohesion \_\_\_\_\_ (a-g)

- a) All elements are activated at a single time.
- b) All elements of a component operate on the same input -> output.
- c) Each component part is necessary for the execution of a single function.
- d) Elements in a component make up a single control sequence.
- e) Output from one element in the component serves as input for another element.
- f) Related components bundled together (e.g., input and error handling).
- g) Unrelated parts bundled together!

21. Claim: models assist in finding design solutions. Give three ways that support this claim:

22. In assessing design quality, what should be the ultimate goal?

23. Identify the correct axiom:

Design correctness is unaffected by movement between equivalent contexts.

Axiom of \_\_\_\_\_.

Design correctness is unaffected by replacement of equivalent components.

Axiom of \_\_\_\_\_.

A complex problem can best be solved by initially devising an intermediate solution expressed in terms of simpler independent problems.

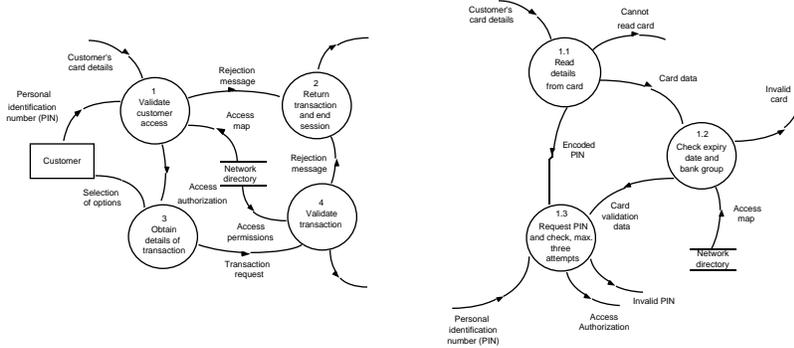
Axiom of \_\_\_\_\_.

The mind cannot easily manipulate more than about seven things at a time.

Axiom of \_\_\_\_\_.

24. The following set of pictures gives some example design representations: In each case, (1) identify (name) the representation form and its requisite method name (if any), (2) the viewpoint(s) expressible and, (3) the design attributes. Also, try to (4) roughly identify the next step in the process that would be necessary given that you were completed with that particular stage in the method.

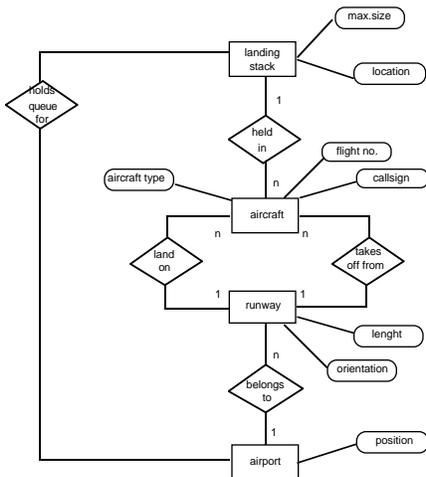
Number 1:



- (1)
- (2)
- (3)
- (4)

Also name the different representation form(s):

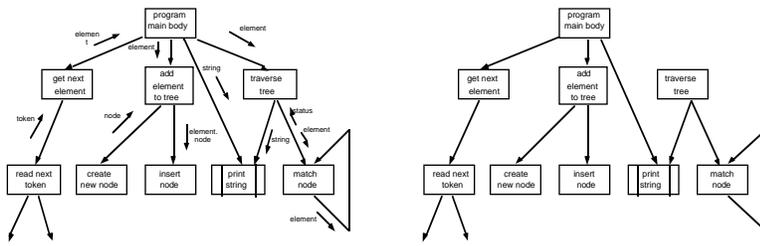
Number 2:



- (1)
- (2)
- (3)
- (4)

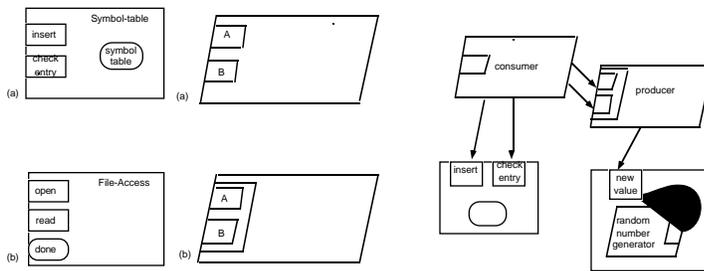
Also name the different representation form(s):

Number 3:



- (1)
- (2)
- (3)
- (4)

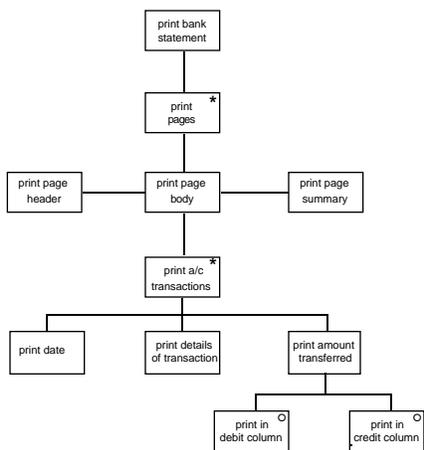
Number 4:



- (1)
- (2)
- (3)
- (4)

Also name the different representation forms:

Number 5:



- (1)
- (2)
- (3)
- (4)

Also name the different representation forms:

Number 6:

---

```

boil water;
pour some water into teapot;
empty teapot;
REPEAT
    place spoonful of tea in pot
UNTIL enough tea for no. of drinkers;
REPEAT
    pour water
UNTIL enough water for no. of drinkers;
    
```

---



---

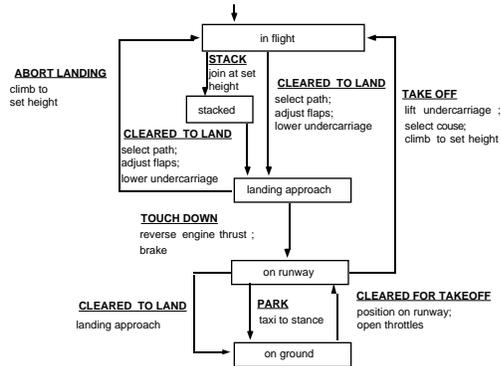
```

INITIALIZE line buffer;
READ first character from keyboard;
WHILE not the end of line DO
    IF character is terminator of a word
    THEN
        mark end of word in buffer;
        SKIP any trailing word separators
    ELSE
        copy character to buffer
    END IF ;
    READ next character;
END WHILE;
    
```

---

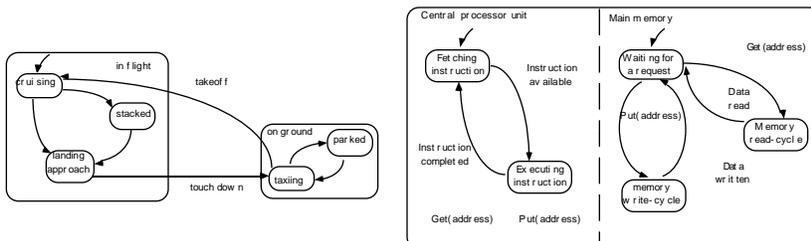
- (1)
- (2)
- (3)
- (4)

Number 7:



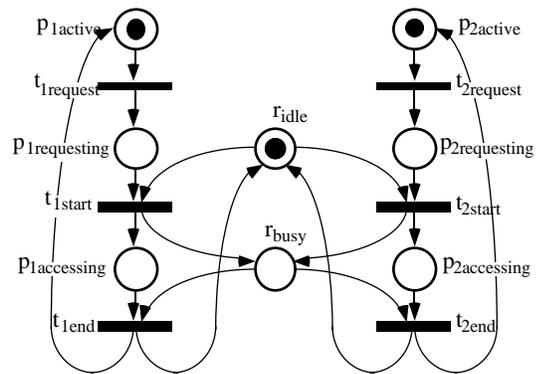
- (1)
- (2)
- (3)
- (4)

Number 8:



- (1)
- (2)
- (3)
- (4)

Number 9:



- (1)
- (2)
- (3)
- (4)

Also name the different representation forms: