1. **Write a Module Interface Specification**
   Write the Module Interface Specification (MIS) for a module that stores circuits made up of simple components. A component is assumed to have the following type:
   `type comp = String*String*String*int`
   where the first string is the name of the component, the second string is the name of the positive terminal of the component, the third string is the name of the negative terminal of the component, and the integer is a scaled, unitless value associated with the component. The names of all the components of a circuit should be unique. The module provides the following functions:

   - `init` takes an integer $n > 0$ and initializes the module such that at most n entries can be stored.
   - `put` takes a component as its only input and inserts it into the circuit. Re-inserting a component is not allowed.
   - `get_value` takes a String as input and returns the integer value associated with the component with that name. Does not change the circuit.
   - `get_reversed_nodes` takes a String as input and returns the String*String tuple corresponding to the terminal names associated with the component, but in reverse order. Does not change the circuit.

   Please pick your exceptions from the following list: notfound, notinit, invalid, full, duplicate, empty, shortcircuit. Note: not all of those exceptions are needed.

2. **Inspection**
Find (circle) and fix 6 errors in the following MID which corresponds to the MIS of question 1. Any correct portion circled will result in 2 marks being lost. Notes: the declaration and initialization of "components" is deemed to be correct pseudo-code, as well as the typedef associated to comp. "fi" is the terminator for an "if". There are more than 6 errors.

```
typedef struct {
    nam:String, pos:String, neg:String, value:integer
    } comp;
Variables:
    isinit:boolean := false;
    components:comp[];  /* Array declaration */
    max:integer;
    size:integer;

init(n:integer)
    components := new comp[0..n-1];
    max := n-1;
    size := 0;
    isinit := true;
bool isempty()
    if not(isinit) then ERROR(notinit) fi;
    RETURN(size = 0);
put(c:comp)
    local i:integer;
    if not(isinit) then ERROR(notinit) fi;
    if (size>max) then ERROR(full) fi;
    for (i := 0; i < size; i++) {
        if (components[i].nam = c.nam) then ERROR( duplicate ) fi;
    }
    components[size] = c;
    size := size + 1;
float get_value(nam:String)
    local i:integer;
    if not(isinit) then ERROR(notinit) fi;
    for (i := 0; i < size; i++) {
        if (components[i].nam = nam) then RETURN( components[i].value ) fi;
    }
    ERROR(notfound)
String*String get_reversed_nodes(nam:String)
    local i:integer;
    if not(isinit) then ERROR(notinit) fi;
    for (i := 0; i < size; i++) {
        if (components[i].nam = nam) then RETURN( components[i].pos, components[i].neg
    }
    ERROR(duplicate);
```

3. **Testing**
Given the MIS of question 1 and the original (unfixed) MID of question 2, give 10 test sequences that illustrate the errors you have found. Do not test the same problem over and over again, but spread out your test cases evenly over the errors. Start each test case from an uninitialized state. Each "Test Sequence" box should contain a **sequence** of calls. Make sure each test sequence clearly finds an inconsistency.

Test Sequence	Final MIS state	MIS Output or Exception	Final MID state	MID Output or Exception

4. What is the difference between an expression and a formula?
5. Name 3 different aspects to verify when doing inspection of a Module Interface Specification (MIS).
6. What is the difference between verification and validation?
7. Compare readthroughs and active reviews. Which is better?