**Section 7.1**

What is a type system?

What is explicit parametric polymorphism?

What the composite types arrays, records, variant records (unions), pointers

**Section 7.2**

What are the differences between structural equivalence and name equivalence?

What is type casting?

What is type coercion?

What are universal reference types?

**Section 7.3**

What is a record field?

What is a record hole?

What is a word or double word alignment?

What is a Union?

How are variants (unions) mapped?

Page 325, Problems 22 and 25.

What are two purposes for using unions (Page 325, Problem 26)?

**Section 7.4**

What is an array index?

Section 7.4.2: static versus dynamic allocation

Page 342, Problem 31.

What are row major order and column major order?

How are array elements addressed in contiguous allocation?

Why do many languages use 0 based lower bounds?

Compare contiguous allocation (as in C) to row-pointer layout (as in Java) (Page 342, Problem 33).

Compare rectangular arrays to ragged arrays.

**Section 7.5**

Statically allocated strings: fixed length, sentinel

Dynamically allocated strings (reference model)

Immutable strings - the string object does not change.  Any operation create a new string object, it does not modify the operand object.

**Section 7.7**

How are pointers and references the same?

How are pointers and references different? Why are references safer? Why are pointers useful?

What are the operators "address of" and "dereference"?

What is pointer arithmetic?  Why is it dangerous? Why is it useful?

**(What is a dangling reference? What causes a dangling reference? (Page 370, Problem 44).**

**What is garbage? How is it created and why is it a problem?**

**How do garbage collection and explicit deallocation differ?**

**Page 370, Problem 49.**

**Page 370, Problem 51.  Consider the problems with having a pointer to an automatic variable such as a local variable.  What can go wrong?**