CS 330 Exam #3 Study Guide
Chapters 9-12

The questions at the back of the chapters are also an excellent source for studying for this exam.

1. Describe the contents of a typical activation record.

2. What’s the difference between a static link pointer and a dynamic link point, as related to subprogram control?

3. What are the differences between an activation stack’s static and dynamic links?

4. Explain what a thunk is and what it does.

5. What are the possible modes of parameter passing? What modes do the following languages support? Ada, Java, C++

6. What are the five models of parameter passing? What are the disadvantages of each?

7. C++ includes a constant reference parameter type. What does that mean, and why is it included along with C++’s other two parameter types?

8. What are the difficulties in type checking parameters?

9. For the following function, consider that the parameters are passed by name. What problem can arise with this example?
   \[
   \text{swap}(x, y) \{ t=x; \ x=y; \ y=t; \}
   \]

10. When local referencing environments are deleted between subprogram activations, using a central stack as in C/C++, it sometimes appears as if values are retained. For example, if procedure Sub has a local variable X and Sub assigns the value 5 to X on the first call, then on a second call, if X is (inadvertently) referenced before it is assigned a new value, sometimes X still has its old value 5. However, in the same program, a third call on Sub may find X has not retained its old value from the second call. Explain this apparent anomaly. In what circumstances could an activation that references an uninitialized variable find that that variable still had a value assigned on a previous call? In what circumstances would it not have its previously assigned value (but may have some other value entirely unrelated to that variable)?

11. Explain why it is impossible in a language with only parameters transmitted by value to write a subprogram Swap of two parameters that simply swaps the values of its two parameters (which must be simple or subscripted variables). For example, Swap(X,Y) should return with X having the original value of Y and Y having the original value of X. For the purposes of this question, consider that the parameter types are integers. Answer the same question for the pass by name parameter passing method.
12. Give an example in code (either Java or C++) of causing the program stack to run out of space.

13. Give an example in code (either Java or C++) of causing the program heap to run out of space.

14. Given a program in a language that allows recursion and uses static scope, show the activation record at a given point in the program’s execution.

15. Given a program in a language that allows recursion and uses dynamic scope, show the activation record at a given point in the program’s execution.

16. Given a program in a language that allows nested subprograms, show the activation record at a given point in the program’s execution.

17. What are the two most important principles in the construction of abstract data types? Describe each.

18. What are the differences between the support for abstract data types in C++ and Java?

19. What are the information hiding mechanisms of Java? How are they different from similar ones in C++?

20. What are the fundamental design issues for object-oriented languages?

21. Give an example of the inheritance mechanism in C++ and Java.

22. Give an example of a hardware generated exception, and an example of a software generated exception.

23. What are the two sources of exceptions?

24. Write Java code that will generate an appropriate exception when attempting to execute the following statement without relying on Java’s automatic exception detection:
   \[ z = x / y; \]

25. What is the main problem with exception range checking subscripts with each array reference?

26. Given the following pseudo-code in some arbitrary language:

```java
Class Aclass {
    method print...
}
Class Bclass is_a A {
```
method print...

A class X;
B class Z;

... X.print;
... X = Z;
X.print;

What are the concerns about the second X.print operation?

27. Consider the following C++ code:

class single_linked_list {
    private:
        class node {
            public:
                node *link;
                int contents;
        };
        node *head;
    public:
        single_linked_list ( ) {head = 0};
        void insert_at_head (int);
        void insert_at_tail (int);
        int remove_at_head( );
        int empty ( );
    }

class stack : public single_linked_list {
    public:
        stack ( ) { }
        void push (int value) {
            insert_at_head(value);
        }
        int pop ( ) {
            return remove_at_head( );
        }
    }

What are the issues involved in how the stack object is dealt with by C++? How is the stack class’s constructor handled? What about the stack class’s need for an empty( ) method? Are there any hazards with this implementation of a stack ADT?