Chapter 3 (3.1 – 3.3) Describing Syntax and Semantics
Chapter 4 Lexical and Syntax Analysis
Chapter 5 Names, Binding, Type Checking and Scopes
Chapter 6 Data Types
Chapter 7 Expressions and Assignment Statements
Chapter 8 (8.1 – 8.4) Statement-Level Control Structures
Chapter 9 (9.1 – 9.10) Subprograms
Chapter 10 Implementing Subprograms
Chapter 11 Abstract Data Types and Encapsulation Constructs
Chapter 12 (12.1, 12.2, 12.3, 12.5, 12.6) Support for Object - Oriented Programming
Chapter 14 (14.1, 14.3, 14.4) Exception Handling
Chapter 15 (15.1 – 15.5) Functional Programming Languages

Q1. For the following grammar:
S → a S b | c E
E → d | f E a

(1) Circle the strings that can be generated by the grammar and cross out those that cannot:
cfda (T) acfab (F) aacdbb (T) aacfdab (F)

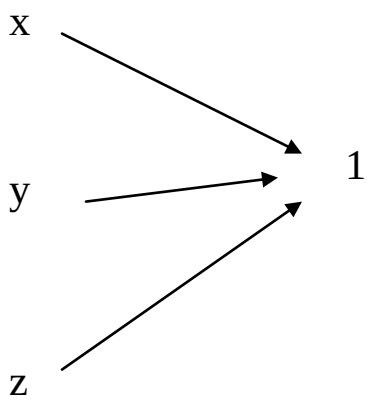
(2) The grammar is ambiguous. True or false (circle one)
false

Q2. const member functions ...

i. are mutator functions
ii. must have a void return type
iii. cannot change its object's data members
iv. can invoke any other member functions in its class

Q3. Consider the following program in C:

```
int x=0;
int p(int y, int z) {
     x=x+1;
     y=y+1;
     z=z+1;
     printf ("%d", x+y+z);
}

void main(){
    int x=1;
    p(x, x);
}
```



- Passing by value: 5
- Passing by reference: 7
  int p(int &y, int &z)

Q4. Write a lambda function in Scheme that takes one argument and returns the value of that argument plus itself.

(lambda (x) (+ x x))

CONS builds a list from its two arguments.

CAR returns the first element of that list.

CDR returns the list after removing its first element

Q5. Assuming that the following definitions are executed in this order:

(define a 6)
(define b '(3 14 27))
(define c (cons a (cdr b)))

What is the result of typing the following into the Scheme interpreter:

c => ??? **(6 14 27)**
(car (cdr c)) => ??? **1**

Q6. Write a Scheme function that delete all a given atom from a given list.

```
(DEFINE (deleteall atm lst)
 (COND
  ((NULL? lst) '())
   ((EQ? atm  (CAR lst)) (deleteall atm (CDR lst)))
   (ELSE (CONS (CAR lst) (deleteall atm (CDR lst)))
))
```

Assume:
(define lst '(3 4 14 3 27))
(define atm 3)

'(4 (deleteall 3 '(14 3 27)))

'(4 14 (deleteall 3 '(3 27))

'(4 14 27 (deleteall 3 '()))

Q7. What dangers are avoided in Java by having garbage collection, relative to C++?

Answer: garbage collection removes the necessity of allowing users to deallocate objects, thereby eliminating the possibility of user-created dangling pointers.

Q8. What is the advantage of binding things as early as possible? Is there any advantage to delaying binding?

Early binding generally leads to greater efficiency (compilation approach)
Late binding general leads to greater flexibility

Q9. In some languages, the user manages the heap. Why is this potentially dangerous?

For example in C++ the user manages the heap by explicitly requesting memory from the heap via new and releasing it by delete. Two common problems that occur are:

(1) dangling pointers: the user releases memory back to the heap "too early" or rather while there are still references to that memory in their program that may be accessed in the future.

(2) memory leaks: when the user neglects to return memory to the heap when it is no longer needed, may cause you to eventually run out of memory on the heap.

10. Consider the following C++ code that contains a memory leak:

```
void foo(){
   int *x;
   while (1)
        x = new int;
}
```

Rewrite this code by inserting a “delete” operator at the appropriate place to eliminate the memory leak. Do not change the code otherwise – simply add the appropriate number of delete statement(s) in the appropriate place(s).

Solution:

```
void foo(){
   int *x;
   while (1){
        x = new int;
       delete x;
  }
}
```

Q11. Consider the Fibonacci series of numbers, where each number in the series is the sum of the preceding two:

0, 1, 1, 2, 3, 5, 8, 13, 21, …

We can define the Fibonacci function in mathematical notation in the following way:

$$Fib(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ Fib(n-1) + Fib(n-2) & \text{otherwise} \end{cases}$$

Define a Scheme procedure Fib that implements this function.

```
(define (fib n)
 (cond ((= n 0) 0)
       ((= n 1) 1)
      else (+ (fib (- n 1)) (fib (- n 2))))))
```

Q12. What will be the output if we run the following program?

```
public class Foo {
   public static void main(String[] args)     {
      int[] a = {9, 11};
         System.out.println( a[0] );

      try {
         if(a[1] > 10 && a[1] < 15)
         {
              throw new RuntimeException();
         }

                System.out.println( a[1] );
      }
      catch( RuntimeException e )
      {
         System.out.println("inside catch");
      }

      System.out.println("program ends");
   }
}
```

Q13. In C++, how is a reference different from a pointer?

Syntactically, a reference is treated as a regular variable. Moreover, once a reference is initialized to refer to a certain memory location, future assignments will keep the reference referring to the same place, whereas a pointer can be made to point somewhere else.

Q14 Consider the following Java program.

```
class Base {
   void m() { System.out.println("Base.m()"); }
   void n() { System.out.println("Base.n()"); }
}

class Der1 extends Base {
    void m() { System.out.println("Der1.m()"); }
}

class Der2 extends Base {
    void n() { System.out.println("Der2.n()"); }

}
```

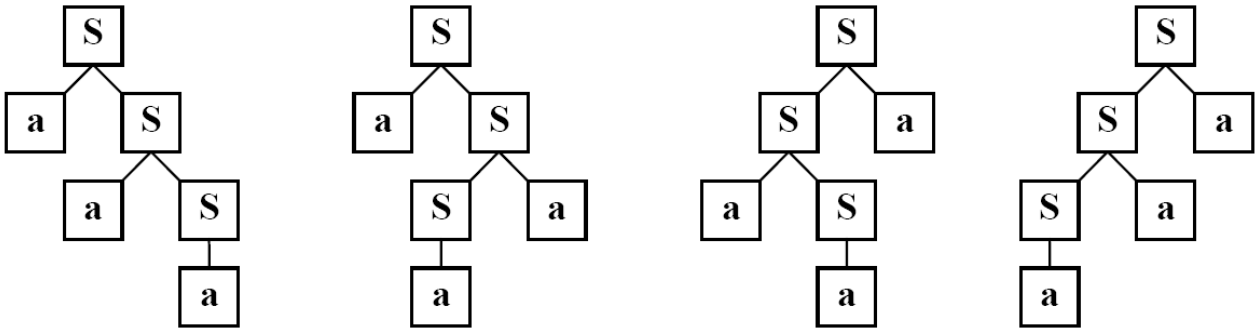
```
class Main{
  static void p(Base x)
  { x.m();
    x.n();
  }

  public static void main(String[] args) {
       p(new Base());
       p(new Der1());
       p(new Der2());
       }
}
```

What does this program print?

Example solutions

Base.m()
Base.n()
Der1.m()
Base.n()
Base.m()
Der2.n()

15. Draw all possible parse trees for the string aaa using this grammar: <S> ::= a <S> | <S> a | a



16. Consider the set of strings that consist of zero or more a's, followed by zero or more b's, followed by zero or more c's, followed by zero or more d's, such that the number of a's equals the number of b's, and also the number of c's equals the number of d's. For example, string aabbcccddd is in this set because it has 2 a's, 2 b's, 3 c's, 3 d's.

Write an unambiguous BNF grammar for this language.

<S> ::= <X> <Y>
<X> ::= a <X> b | <empty>
<Y> ::= c <Y> d | <empty>

17. Discuss the tradeoffs between providing an explicit memory deallocation operator and providing a complete and correct implementation of garbage collection.

Garbage collection is computationally expensive and sometimes unnecessary. Explicit deallocation allows the programmer to control when deallocation is performed and is generally less expensive than garbage collection. However, explicit deallocation also allows for programmer errors in terms of dangling pointers and memory leaks.

18. T/F Two primary reasons for subclassing are to achieve **code-reuse** or **polymorphism**.

Answer 1: True. There are other reasons, but usually you subclass because you've found a class that already does *almost* what you want to do (reuse) or you want to be able to treat all subclasses in terms of their parent (polymorphism).

19 In C++, static methods and data members are only accessible from static methods. Static methods and data cannot be accessed from non-static methods.

Answer: False. A non-static method can call a static one.

20. The compiler provides your class with a built-in no-argument constructor, but once you define *any* constructor, the built-in one is no longer generated.

Answer: True. Even if you only write a copy constructor, the no-argument constructor will go away.

You can define multiple constructors to provide optional ways to create and initialize instances of the class.

```
class Test {
  int i;
  boolean b=true;

  Test(int j, boolean a){
    i = j;
    b = a; }

  Test(int j){
    i = j; }

  int get(){
    return i;}
}
```

This constructor initializes the two properties to the values passed as arguements.

A second constructor passes an initial value to just one of the variables.

If a class includes one or more explicit constructors, the java compiler does *not* create the default constructor. So for the class shown above, the code

```
Test ex = new Test();
```

will generate a compiler error stating that no such constructor exists.

21. If a function provides a throw list but doesn't include any exceptions in the list, then it can safely throw *any* exception. In other words, the following function can throw any exception without invoking the unexpected handler:

```
void tomServo(int i) throw()
{
[etc]
}
```

Answer: False. An empty throw list means that you won't throw anything. If you omit the throw list entirely, that means you can throw anything.

```
22. /* Represents a square matrix. */
   typedef struct squarematrix {
        int n; /* The dimension of this matrix. */
   } SquareMatrix;
    /* Returns the element A i,j. */
   int get(const SquareMatrix A, const int i, const int j) {
    /* Implementation omitted. */
   }
```

Using those fragments, complete the function isSymmetric below, which takes as arguments a square matrix A and returns true if A is symmetric, false otherwise.

```
int isSymmetric(const SquareMatrix A) {
  int i, j;
  for (i = 0; i != A.n; i++) {
    for (j = i+1; j < A.n; j++) {
        if (get(A, i, j) != get(A, j, i)) {
  return 0;}
  }
}
return 1;
}
```