Andrew Morris
2-8-16
CSCI 330
Assignment 2

BOOK EXERCISES:

a. The most common syntax error in LISP is most likely is involved in parentheses matching.
b. This is because LISP began as a list processing language. As it's applicability grew is areas like AI, the more features were designed and added to the language.
c. Their assumption was correct in that inevitably business applications, and scientific calculations would cross paths. However, this feat was far too broad to be covered effectively by one language.
d. Strong typing easily allows the compiler to perform type checking at compile time. This ensures that type conflicts do not result when the program is running. If the language is type-less, then the compiler cannot perform type checking which can lead to corrupted data. However, type-less languages are more flexible, and syntax is brief, and variables can be reused for other purposes.
e. Adding new features should be avoided until the core functionality of a language is optimized. If the core functionality of the language is not sufficiently optimized, then how are new features of a program that *use* the core features of a program supposed to be optimized. FORTRAN has survived because of it's ability to do a specific task exceedingly well, not being a 'jack of all trades'.
f. Situations where pure interpretation is acceptable:
    a. When a program is interactive with a human, pure interpretation is acceptable. Many recent scripting languages drive online web applications. Because of the interpretation, they are also incredibly portable.
    b. When there is not a large amount of computing that is needed. In this situation, the time to interpret a language does not outweigh the time it takes to compile a program.
g. Yes, I am noticing more and more that scripting languages have been slowly taking over. I believe this to be in large contribution to the speeds of machines. Now, because of exceedingly high amounts of unused memory and CPU time, machines generally have the resources to offer to cover the operation overhead of interpretation. Interpreted languages are by nature, portable. This makes these languages more appealing as well.

SUPPLEMENTARY QUESTIONS:

Imperative(C):

```
int * studentGrades;
Int sum, n;
float average;
studentGrades = malloc(n * sizeof(int));
/*Add students grades*/
for(int i = 0; i < n; i++) {
```

```
   sum += *(studentGrades + i);
}
average = sum/ (float)n;
```

Logical Language(Prolog):

```
Sum_list([],0).
Sum_list([H|T], S) :-
        Sum_list(T, Rest),
Sum is H + Rest.
Avg is Sum / n.
```

Functional Language(LISP):

```
(setq avg / sum(+ studentList) n)
```

Scripting Language(python):

```
Avg = Sum(student_GradeList) / (float(n))
```

Object Oriented Language(Java):

```
List<Student> students = new LinkedList();
/*populate list*/
Integer sum;
for (Student i: students) {
        sum+= i.grade();
}
Double avg = Double.toDouble(sum) / n;
```