1. **foo() || bar()** and **foo() && bar**
   a. `return foo() ? true: bar();`
   b. `return foo() ? bar():false;`
2. **Here are the different versions:**
   a. without Boolean (NOT Midtest)

```
line = read_line();
while(!all_blanks(line)){
      consume_line();
      line = read_line();
}
```

   b. with Boolean flag

```
boolean blank = false;
while(!blank){
      line = read_line();
      blank = all_blanks(line);
      if(!blank) consume_line();
}
```

These both will consume the same thing, the first one is better because it has a fewer number of conditions.

3. **Garbage Collection**

a. Most languages that have garbage collection with the exception of a few functional languages are not really considered to be efficient. When memory efficiency matters this kind of function or method is available for use. For example in C the free() function frees up memory that the programmer doesn't think he/she needs. In a language with a garbage collector and a `delete` method it would be difficult for a programming team to determine if their objects were still around.
b. "tenure" of object is a bad idea, because it would lead to large memory leaks in a program. Typically a garbage collector does not error on the side of deleting an object too soon.

4. **How are arguments passed to subroutines in your pet language? If there is more than one option, how does the system determine which one to use?**

In Coffeescript arguments are passed by value:

```
test = (a) -> a+3

a = 3
alert(a)
test(a)
alert(a)
```

The following code will print "3" twice. This means that arguments are passed by value. If they were passed by reference then a would have the value 6 stored in it after calling test(a), resulting in the second alert printing 6.

5. **Here is java code to determine which parameter is called first**

```
class Order{

	public static void main(String[] Args){
		order(a(),b(),c(),d());
	}


	public static void order(boolean a,boolean b,boolean
c,boolean d){
		System.out.println("Done!");
	}


	public static boolean a(){
		System.out.println("A");
		return true;
	}

	public static boolean b(){
		System.out.println("B");
		return true;
	}

	public static boolean c(){
		System.out.println("C");
		return true;
	}

	public static boolean d(){
		System.out.println("D");
		return true;
	}
}
```

This code will return the following

```
A
B
C
D
Done!
```

So Java must evaluate expressions in the order which they are given as parameters.

Here is coffeescript code to determine how parameters are evaluated.

```
order = (a,b,c,d) -> alert "Done!"

a = alert "a"

b = alert "b"

c = alert "c"

d = alert "d"
```

This program alerted me in the following order…... a,b,c,d.