CSC-363: Problem Set 2: Language Characteristics II

1. Consider class `C` that defines method `update`.

```
class C
{
   int _a;

   void update()
   {
      // ...
   }
}
```

Consider a function `func` in which `c` is an object of type `C` being passed by constant reference.

```
void func(const reference C c) { c.udpate(); }
```

a) Describe an implementation of `update` for when when `c.update()` may be called in `func`.
b) Describe an implementation of `update` for when `c.update()` cannot be called in `func`.
c) Propose a modification to the language to ensure quick compile-time verification of whether `update` can be called or not.

2. State the output of the following code.

```
int A(name int a)
{
   print(a);
   return a;
}

int B(reference int b1, const reference int b2)
{
   b1 = b2 * b2;

   return b1;
}

void C(value int c1, value result c2)
{
   c2 = A(B(c1, c2));

   A(B(c1, c2));
}

int m = 2;
int n = 3;

C(m, n);

print(m);
print(n);
```

In stepping through this code, how many times is `B` executed? Hint: recall that pass-by-name refers to a lazy evaluation procedure (only evaluating the expression when the argument is used.)

3. Many programming languages allow large programs to be subdivided into **units** (often calls **packages** or **modules**). Each unit is given a name and may be separately compiled. A unit contains a collection of definitions, including types, constants, variables, classes, and subprograms.

A member of a unit may be accessed directly using the unit name and the member's name (for example `U.a` means identifier `a` in unit `U`). Java uses this notation to access members of a package. If we intend to use a number of names defined in one or more units (or to use a given name several times), it is useful to **import** the contents of a unit into the current scope. Assume that

$$\texttt{import } u_1, u_2, \ldots, u_n;$$

causes all the declarations that appear in u_1 through u_n to be imported into the current scope. After an import statement imported names may be used without qualification (just like locally declared identifiers).

Two potential ambiguities arise:

(i) What if the same identifier is declared in more than one imported unit?
(ii) What if an imported identifier clashes with an identifier already visible using normal scoping rules?

For each of these two potential ambiguities, propose and analyze possible resolutions to the ambiguities. Which resolutions do you recommend, and why?