2008 Paper 13 Question 6

Compiler Construction

(*a*) Languages like Lisp, Prolog and Python are said to be *dynamically typed*. Explain this concept and its implications for the size of a run-time storage cell needed to hold a value which may be an integer or floating-point value. [4 marks]

(*b*) Consider the following object-oriented program in Java style:

```
class A { int a,b; };
class B extends A { int c,d1,d2,d3,d4,d5,d6,d7,d8,d9; };
...
static void f(A x) { x.a = 1; }
static void g(B x) { x.c = 2; }
static void h() { A p = new A(); f(p); g(p); }
static void k() { B p = new B(); f(p); g(p); }
static void main() { h(); k(); }
```

(*i*) Explain the run-time structure of values of type `A` and `B`. Indicate a constraint on the layout of these structures needed to support *inheritance*. [3 marks]

(*ii*) Indicate why the above program would not compile in Java and insert a single *cast* to make it compile. Why are two casts not required? [2 marks]

(*iii*) What happens when your Java program in part (*ii*) is executed? [2 marks]

(*iv*) Make an analogy to part (*a*) to argue why a Java value of type `A` requires more storage than that required for two integers. [2 marks]

(*v*) C++ traditionally allows values of type `A` to occupy just the space required by two integers. Comment on the implications for safety if this were allowed in Java. [2 marks]

(*c*) Explain where storage for `new` comes from. Some languages have a primitive `dispose` which de-allocates space allocated by `new`, but Java does not. Explain the implications of this for Java implementation, particularly how a program can perform `new A()` every millisecond but never run out of memory. Suppose that, while executing such a program, a `new B()` is executed. Explain, giving reasons, whether this is guaranteed to succeed in a situation where exactly half the memory available for `new` is in use. [5 marks]