1 Concepts in Programming Languages (AM)

(a) Various languages provide a built-in ‘eval’ operator which evaluates an expression passed as an argument. Discuss the extent to which this: (i) fits with existing language features, naming languages or classes of languages for which it is easy or hard to implement; (ii) easily deals with variable scoping; (iii) is a security risk. [4 marks]

(b) (i) Explain and justify what goes wrong when the following code is given to a Standard ML system:

```ml
fun id x = x;
val fnlist = ref [id];
fnlist := (fn x=>x+1) :: !fnlist;
fnlist := Math.sqrt :: !fnlist;
print (hd(!fnlist)(1))
```

(ii) Explain, giving an example, a related problem involving polymorphic exceptions. [5 marks]

(c) (i) Explain the concept of a “value type” in an object-oriented language, including which, if any, primitive and non-primitive types in Java can be seen as value types.

(ii) Discuss to what extent a programmer can use `final` to create value types in Java, and whether this implementation gives the expected space and time usage. [Hint: You may find it useful to discuss arrays of complex numbers.] [5 marks]

(d) An implementation of finite sets of natural numbers in Standard ML uses `int list` as its representation. However, certain client code has been found to be buggy, because it misuses `::` to add elements (creating duplicates) and `length` to obtain the number of elements (miscounting duplicates).

(i) Explain how ML modules might be helpful for addressing such bugs.

(ii) Use the ML modules language to create a type `natset` which uses `int list` internally but only exposes operations (a) to create an empty set, (b) to (functionally) insert one (non-negative) element into a set, (c) to sum the elements in a set, (d) to count the number of elements in a set. No other operation may create or manipulate an `natset` value. [6 marks]