SOLUTION NOTES

Compiler Construction 2003 Paper 6 Question 6 (AM)

[Questions taken from various parts the course]

1. Nonsense. Two alternatives to conservative garbage collection are two-space (copying) garbage collection or explicit de-allocation.

2. False. Consider the stack frame for g in

```
let main() =
{ let a = 1
  let g() = a
  let f() = g()
  return f();
}
```

3. True. The dynamic chain is the chain of callers and the exception chain is the chain of exception handlers (which also nest dynamically). Since each procedure call adds one exception frame the claim holds (modulo muttering about start and end frames).

4. False. Such a language would need to have explicit parse tree representation for parenthesised expression as a unary node which therefore means that the two expressions have different trees.

5. True. It would be a lot less efficient, and indeed wiring together two copies of the output of yacc could be quite tricky in implementation terms, but it would work. Incidentally (but not needed for marks) if you try to put both the lexical and syntax rules into a *single* grammar the result is unlikely to be LALR(1) even if both are separately.

6. True. This is just a special case of the type checking rules.

7. False.

```
let main() =
{ let g() = a
  let f(a) = g()
  return f(1);
}
```

8. False.

```
let H(p) = p()
let main() =
{ let g = let a = 1
          let f() = a
          in f;
  return H(f);
}
```

9. Nonsense. Dynamically typed programs means that types are checked at run-time. Dynamic linking means that libraries are loaded on demand.

10. True. There are a lot of differences, but the essential idea that both reflect compiled code, one for native execution and the other for JVM execution, together with information about what names are exported and what are external names are required during execution, still holds.