2008 Paper 11 Question 3

Compiler Construction

(*a*) Given the following program fragment in C or Java

```
static int a = 3;
static int f(int x, int y) { int z = a+x; ...; return ... }
```

explain how the four variables (`a`, `x`, `y` and `z`) are accessed from within `f` at the instruction level for an architecture such as MIPS, ARM or x86. Pay particular attention as to how and when storage is allocated for these variables, and which system components of a standard compile-link-and-execute model are involved in selecting the instruction and determining the run-time address calculation. Your answer should briefly explain what occurs when `f` makes a recursive call to itself. [8 marks]

(*b*) Suppose we extend the language to allow nested function definitions:

```
static int a = 3;
static int f(int x) {
  static int g(int y) { int z = a+x; ...; return ... }
  return g(7);
}
```

(*i*) Complete the definition of `g` in such a way that, during execution of a call to `f`, the difference between the addresses allocated to `x` and `y` varies. [2 marks]

(*ii*) Explain a possible run-time mechanism that allows all four variables to be accessed from inside `g`. [6 marks]

(*c*) Suppose function-valued variables are added to the language, e.g.:

```
funvar double v(double);
v = (...) ? sin : cos;
```

and a pointer to code or to data is represented as a 32-bit value. How many bits might naturally be used to hold `v`? Justify your answer, emphasising the way in which it might differ according to whether the language is as given in part (*a*) or part (*b*) above. [4 marks]