2003 Paper 6 Question 6

Compiler Construction

Each of the following statements may be true, false, or nonsensical. Indicate which and (respectively) provide a (one-sentence) justification of why it holds, a counter-example or other explanation of why it fails, or corrected statement.

(*a*) In a language with stack-allocated free variables, the static chain pointer always points to the caller's stack frame.

(*b*) If each procedure sets up one exception handler whose scope includes all of the procedure calls it makes, then the entries in the exception handler stack and those in the dynamic chain will be in 1–1 correspondence.

(*c*) Fortran gives semantic meaning to parentheses, i.e. `a+b+c` permits compiler re-arrangement whereas `(a+b)+c` does not. This matters for floating point. However, since Fortran '`+`' is left associative the above two expressions yield the same parse tree, and we have a contradiction.

(*d*) Two alternatives to conservative garbage collection are liberal garbage collection or new labour style de-allocation.

(*e*) Any type 3 (regular) grammar is also a type 2 (context-free) grammar. Hence any lexer generated by `lex` (or `JLex`) could instead be generated by `yacc` (or `Cup`).

(*f*) When a compiler for a language like Java compiles e_1+e_2, it computes the types of e_1 and e_2 so that it can treat it as e_1+`(float)`e_2 if e_1 is of type `float` and e_2 is of type `int`.

(*g*) A syntax-tree interpreter can fail (give an error) when evaluating a variable name which does not appear in the current environment. Any program which fails using static scoping will fail using dynamic scoping.

(*h*) A syntax-tree interpreter can fail (give an error) when evaluating a variable name which does not appear in the current environment. Any program which fails using dynamic scoping will fail using static scoping.

(*i*) A dynamically-typed language is one in which the type of a value is carried around at run-time; a type error is given at run-time if values are used inappropriately. Such languages must be dynamically linked otherwise type errors will occur.

(*j*) Java `.class` files and ELF-style `.o` (or `.obj`) object files represent similar information, i.e. compiled code, a list of symbols made available to other functions, and a list of undefined symbols which the file expects to be defined elsewhere.

[2 marks each]