UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER SCIENCE THREE

COMPILING TECHNIQUES

Thursday 8 June 2000

14:00 to 15:30

Examiners:

Prof. K. Turner (External)

Prof. G. Brebner (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. (a) Explain what it means for a context-free grammar to be *ambiguous*, and why this is undesirable for grammars used to specify the syntax of programming languages. *[4 marks]*

(b) Consider the following grammar:

$$
\begin{array}{lcl}
E & \rightarrow & E\,B\,E \\
E & \rightarrow & \mathsf{num} \\
E & \rightarrow & (E) \\
B & \rightarrow & + \\
B & \rightarrow & - \\
B & \rightarrow & * \\
B & \rightarrow & \backslash
\end{array}
$$

i. Explain why this grammar is not suitable to form the basis for a recursive descent parser. *[3 marks]*

ii. Use left-factoring and left-recursion removal to obtain an equivalent grammar which can be used as the basis for a recursive descent parser. *[5 marks]*

iii. Assuming procedures to check for lexical items such as num, write a recursive descent parser for your grammar in part (ii). *[5 marks]*

(c) Write an unambiguous grammar for statement blocks as in ML, where the semi-colons *separate* the statements. E.g.

(`statement` ; (`statement` ; `statement`) ; `statement`)

[4 marks]

(d) Write an unambiguous grammar for statement blocks as in C, where the semi-colons *terminate* the statements. E.g.

{`expression`; {`expression`; `expression`; } `expression`; }

[4 marks]

2. (a) Each function call during the execution of a program causes a *stack frame* to be allocated. Describe the layout of a typical stack frame, identifying the different kinds of values which are stored there. [*5 marks*]

(b) State conditions under which a local variable of a function is said to *escape* from that function. How does a variable escaping affect how it can be stored? [*3 marks*]

(c) For each of the variables a, b, c, d, e in the following C program, say whether they should be kept in memory or a register, and why.

```
int f(int a, int b)
{ int c[5], d, e;
  d = a+c[2];
  e = g(&b, c);
  return b + e;
}
```

[*5 marks*]

(d) Explain the concept of a *static link*. Give an example of a Tiger program to illustrate the use of static links. [*7 marks*]

(e) Which class of functions in Tiger do *not* need a static link passed to them? [*5 marks*]

3. (a) Describe the liveness analysis phase of compilation. Explain its purpose, and outline how it is performed. [*5 marks*]

(b) Explain the distinction between *dynamic* and *static* liveness, and the sense in which the liveness analysis performed by a compiler gives a *conservative approximation* of the dynamic semantics. Why is this unavoidable? [*5 marks*]

(c) Consider the following program:

```
   a := 0
L1:c := a+1
   d := a+c
   b := 2*d
   a := b-d
   if a<3 then goto L2
   c := a
   goto L1
L2:return c
```

(i) Draw the control-flow graph for this program. [*3 marks*]

(ii) Calculate the live-in and live-out at each node of the graph. [*5 marks*]

(iii) Draw the register interference graph. [*2 marks*]

(d) Based on the analysis of part (c), how could later phases of compilation optimise the storage requirements and run-time of the program? [*5 marks*]