Tuesday, 15th May 2012
9.30 am – 11.00 am
(Duration: 1 hour 30 minutes)

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

COMPUTING SCIENCE 3Z:
PROGRAMMING LANGUAGES 3

Answer all 3 questions.

This examination paper is worth a total of 60 marks.

You must not leave the examination room within the first half-hour or the last fifteen minutes of the examination.

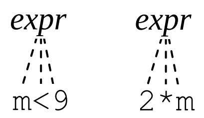
1. (*Syntax*)

Box 1 shows part of the BNF grammar of a fictional programming language called FPL. It shows the syntax of statements and sequential statements. It does not show the syntax of expressions (not needed in this question).

(a) Show the syntax tree of the following FPL statement:

```
while m<9 loop m := 2*m; end;
```

You may assume that `m<9` and `2*m` are expressions. Your syntax tree should show these expressions in outline:



[5]

(b) Suppose that FPL is to be extended with a loop-statement with multiple conditional exits. For example, the following loop-statement contains two conditional exits:

```
loop
     m := m+1;
     exit when m=n;
     f := f*m;
     exit when f>1000;
end;
```

In general, the loop-statement may contain any number of statements and conditional exits, in any order, all enclosed between "`loop`" and "`end`". Conditional exits are permitted immediately inside a loop-statement, but nowhere else.

Modify the grammar to allow for loop-statements. You may use either BNF or EBNF.

[5]

statement	=	*ident* "`:=`" *expr* "`;`"
	|	"`while`" *expr* "`loop`" *seq-statement* "`end`" "`;`"
	|	…
seq-statement	=	*statement*
	|	*seq-statement statement*

Box 1 Part of the grammar of programming language FPL.
(Here *expr* is an expression, and *ident* is an identifier.)

2. (*Implementation*)

(a) Consider the assignment statement:

```
a = b*(c-4*d)
```

where `a`, `b`, `c`, and `d` are all 32-bit integer numbers. Give two assembly-code translations of the statement:

(i) using stack code; and

(ii) using register code.

[16]

Now give a quantitative analysis of the relative efficiencies of the two translations, in terms of the number of clock cycles.

[5]

(b) In a Basic compiler similar to the one you constructed in your course, what would be the intermediate code tree and the assembly code generated for the following statement?

```
LET A(I):= 9
```

[9]

3. (*Concepts*)

(a) Briefly explain how the concepts of *Cartesian products*, *disjoint unions*, and *mappings* are relevant to the understanding of programming languages.

[3]

(b) Using the notation of Cartesian products, disjoint unions, and mappings, write equations defining the set of values of each of the following C types:

```
enum Colour {RED, GREEN, BLUE};
struct CharCount {char c; int i;};
typedef CharCount[] CharProfile;
```

[3]

(c) Again using the notation of Cartesian products, disjoint unions, and mappings, write an equation defining the set of objects in a Java program that includes the following classes:

```
abstract class Animal {
     private float weight;
     private boolean can_fly, can_swim;
     ...  // methods
}

class Bird extends Animal {
     private int eggs;
     ...  // methods
}

class Mammal extends Animal {
     private float gestation;
     ...  // methods
}
```

Note that `Animal` is an abstract class.

[3]

(d) Explain the difference between the *copy-in* and *reference* parameter mechanisms.

[2]

(e) Which of the parameter mechanisms of part (d) are supported by Java, and for which types of parameters?

Illustrate your answer using the following method:

```
static void p (Animal b, float[] fs, float f) { … }
```

What happens to this method's parameters (i) on call and (ii) on return?

[4]

(e) Suppose that the class `Animal` contains the following abstract method:

```
abstract public void m (float x);
```

and that the classes `Bird` and `Mammal` define different versions of this method.

Consider the method call in the following code:

```
Animal b = …;
b.m(1.5);
```

What determines the target object of the method call? What determines which version of the method is called? How does the called method access the target object?

[5]