University of Glasgow

Tuesday, 7 May 2013
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 three 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.

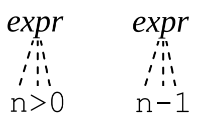
1. (*Syntax*)

Box 1 shows part of the BNF grammar of the programming language Fun. It shows the syntax of commands and sequential commands. It omits the syntax of expressions (not needed in this question).

(a) Show the syntax tree of the following Fun command:

```
if n>0 : n = n-1 .
```

You may assume that `n>0` and `n-1` are expressions. Your syntax tree should show these expressions in outline:



[4]

(b) Suppose that Fun is to be extended with a switch-command with multiple cases. For example, the following switch-command contains two cases:

```
switch n :
   case 1: a = 1  b = 2
   case 3: c = 3
.
```

and the following switch-command contains one case and a default:

```
switch n :
   case 1: a = 1
   default: x = 0  y = 0
.
```

In general, the switch-command may contain any number of cases and at most one default. Each case contains a numeral and a sequential command. The default (if any) comes last, and contains a sequential command.

Modify the grammar to allow for switch-commands. You may use either BNF or EBNF.

[6]

```
    com  =  id '=' expr
         |  'if' expr ':' seq-com '.'
         |  …

seq-com  =  com
         |  seq-com  com
```

Box 1 Part of the BNF grammar of programming language Fun (*com* is a command, *seq-com* is a sequential command, *expr* is an expression, *id* is an identifier, *num* is a numeral).

2. (*Concepts*)

(a) Java is a statically-typed language, whereas Python is a dynamically-typed language. Carefully explain the difference between *static typing* and *dynamic typing*.

What are the advantages of static typing, and what are the advantages of dynamic typing?

Illustrate your answer using the following Java method:

```
static int min (int m, int n) {
   if (m < n)
      return m;
   else
      return n;
}
```

and the following Python function:

```
def min (m, n):
   if m < n:
      return m
   else:
      return n
```

[6]

(b) In what circumstances would the Python function of part (a) fail at run-time? In what circumstances would the Java method of part (a) fail at run-time?

[2]

(c) Carefully explain the concept of *encapsulation* in a programming language. Compare and contrast the support (or lack of support) for this concept in Java and in Python.

[6]

(d) Suppose that you are required to design a program unit that maintains a single queue (first-in-first-out sequence) of strings. Application code must be able to add an element to the rear of the queue, to remove the element at the front of the queue, and to test whether the queue is empty. If possible, application code should be prevented from accessing the queue in irregular ways.

Outline such a program unit (i) in Java, and (ii) in Python. For simplicity, assume that the elements are strings. Show declarations of variables, functions, etc., but do not implement them.

How effective is each program unit in preventing application code from accessing the queue in irregular ways?

[6]

3. (*Implementation*)

(a) Explain the difference between a *compiler* and an *interpreter*.

[2]

(b) Explain the role of the *syntactic analysis*, *contextual analysis*, and *code generation* phases of a compiler. How do these phases typically communicate with each other?

[4]

The rest of this question concerns the ANTLR compiler generator, which you have seen being used to generate a Fun → SVM compiler.

(c) Box 3a shows parts of a *grammar file*. Explain carefully what ANTLR does with this grammar file.

[6]

(d) Box 3b shows parts of a *tree grammar file*. Explain carefully what ANTLR does with this tree grammar file.

[8]

(e) Suppose that the Fun language is to be extended with a repeat-command such as the following:

```
repeat :
   p = p*2
   g = g+1
until p < n
```

The syntax should allow any sequential command between '`:`' and '`until`'. The semantics should be to execute the sequential command repeatedly until the expression following '`until`' yields true. (The expression is to be evaluated *after* the sequential command is executed.)

Show how the files of Box 3a and 3b should be modified to achieve this extension.

[10]

```
grammar Fun

…

com
     : ID ASSN expr                   -> ^(ASSN ID expr)
     | IF expr COLON seqcom           -> ^(IF expr seqcom)
     | …
     ;

seqcom
     : com*                           -> ^(SEQ com*)
     ;

…

IF     : 'if' ;
ID     : LETTER+ ;
ASSN   : '=' ;
COLON  : ':' ;
…
```

Box 3a Outline of an ANTLR grammar file.

```
tree grammar FunEncoder

…

com
     : ^(ASSN ID expr)      { let d be the address of variable ID
                              emit an instruction 'STORE d'
                            }
     | ^(IF expr            { let c1 be the address of the next instruction
                              emit an instruction 'JUMPF 0'
                            }
          com)              { let c2 be the address of the next instruction
                              patch c2 into the instruction at address c1
                            }
     | ^(SEQ com*)
     | …
     ;

…
```

Box 3b Outline of an ANTLR tree grammar file.
(For clarity, actions are expressed in English rather than Java.)