RHODES UNIVERSITY

November Examinations - 1996

Computer Science 3 - Paper 3

Examiners:
Prof P.D. Terry
Prof A.G. Sartori Angus

Time 3 hours
Marks 180

Answer all questions from sections A and B, and ONE of the two questions in section C.
Answers may be written in any medium except red ink.

(For the benefit of future readers of this paper, various free information was made available to the students 24 hours before the formal examination. This included all material marked as appendices, the full text of Section B, and the information that there might be other be questions based on the language defined by the grammar for Topsy++. During the examination, candidates were given machine executable versions of the Coco/R compiler generator, the sources for a version of the Topsy compiler as used in section B, and access to a computer.)

Section A [90 marks]

A1 Explain what is meant by each of the following. Where possible, give simple examples to illustrate your answers.

(a) Syntax
(b) Static semantics
(c) Dynamic semantics
(d) Ambiguous grammars
(e) Context free grammar
(f) Context sensitive grammar

[18]

A2 What are the main differences between a compiler and an interpreter? What advantages and disadvantages accrue from the use of an interpreter rather than a compiler?

[5]

A3 Explain what is meant by the FIRST and FOLLOW sets as applied to grammar analysis, describe how these sets are evaluated, and state the rules that a grammar must obey for it to be LL(1).

[16]

A4 Shakespearian plays all have five acts. In each act there may be several scenes, and in each scene appear one or more actors, who gesticulate and make speeches to one another (for the benefit of the audience, of course). Actors come onto the stage at the start of each scene, and come and go as the scene proceeds - to all intents and purposes between speeches - finally leaving at the end of the scene (in the Tragedies some leave dead, of course, but they usually revive themselves for the after-show party). Plays are staged with an interval between the third and fourth acts.

Describe the form that a play takes as a formal grammar, with the productions expressed in BNF or EBNF notation.

[10]

Go on to test whether your grammar obeys the LL(1) constraints. Show your workings in some detail.

[8]

If your grammar is not LL(1), can you write an alternative grammar that you believe is LL(1)? If not, can you suggest a modification to the way in which plays are presented that could be described by an LL(1) grammar?

[4]

A5 Consider the following Cocol/R description of a simple language

```
COMPILER A
  CHARACTERS
    digit  = "0123456789" .
    letter = "abcdefgefghijklmnopqrstuvwxyz" .
  TOKENS
    number = digit { digit } .
    identifier  = "a" { letter } .
  PRODUCTIONS
    A = B "." .
    B = identifier | number | "(" C ")" | "(." B ".)" | .
    C = B D .
    D = { "+" B } .
END A.
```

(a) Construct a scanner for recognizing the tokens used by this language.

[12]

(b) What are the advantages and disadvantages of using a FSA approach to scanner construction?

[3]

A6 Recursive descent parsers have to pay particular attention to the problem of error recovery to handle the not-uncommon situations where they are presented with source programs that are syntactically incorrect. Describe a systematic approach that might be taken to handling this problem.

[14]

SECTION B [60 Marks]

Please note that there is no obligation to produce a machine readable solution for this section. Coco/R and other files are provided so that you can enhance, refine, or test your solution if you desire. If you choose to produce a machine readable solution, you should create a working directory, unpack EXAM.ZIP, modify any files that you choose, and then copy all the files back to the blank diskette that will be provided. In this case it would help if you were to highlight your changes with `(* +++++++++ comments ++++++++ *).`

The current meeting of ISO/IEC JTC1/CS22/WG1984 has erupted into chaos. ISO/IEC JTC1/CS22/WG1984 is, as you may recall, the international committee of experts charged with the responsibility of standardizing the language Topsy++ which is poised to Take Over The World.

The trouble started when delegates pointed out that, while they were happy that in Topsy++ each identifier should be declared before it was used, the current definition requires that *all* identifiers be declared before *any* are used. Thus, while a fragment of code like

```
void main (void) {
  int j, k = 10;
  bool okay;
  j = k + 100;
  okay = j > 55;
}
```

is currently legal both in C++ and Topsy++, a fragment like

```
void main (void) {
  int j, k = 10;
  j = k + 100;
  bool okay = j > 55;
}
```

is currently illegal in Topsy++, even though it would be acceptable in C++.

Further trouble erupted when some delegates pointed out that Topsy++ and its implementations are supposed to be very safe. They have objected to the fact that one is currently allowed to write `for` loops like

```
for (i = 0; i < 10; i++) {
  cout << i;  // So far, so good
  i--;        // Oh dear - this will cancel out the effect of i++
}
```

which would result in an infinite loop.

These complaints have been referred to a Working Group of delegates from Rhodes University, South Africa. The Committee have requested that the definition of Topsy++, and the prototype implementation under development by the Working Group, be modified so that:

(a) Identifiers may be declared at any convenient point before they are used (that is, declarations and statements may be intermingled).

(b) An identifier may be declared at most once in any scope, but the scope of identifiers should be constrained as tightly as possible. Thus, for example, in code reading

```
if (List[i] > List[i+1]) { // swap two elements
  int temp = List[i]; List[i] = List[i+1]; List[i+1] = temp;
}
```

the scope of the variable `temp` should extend only from the point where it is declared to the end of the statement sequence forming the body of the `if` statement. However, a declaration of `temp` in a surrounding scope would be legal; the scope of this previous declaration would not extend over the body of the `if` statement, but would extend over statements that preceded and followed the body of the `if` statement.

(d) The existence of variables is to be made as economical as possible. Thus, to use the same example as before, storage allocated for the variable `temp` may be used for other purposes at other stages in the program's execution.

(e) Some means must be found of preventing a `for` loop control variable from being altered, either by assignment, or by the action of an input statement within the statement that forms the body of the `for` loop.

Since the standardization of Topsy++ has already taken far too long, the Working Group have been given a strict 24 hour limit to study these requests, at the end of which they are required to come up with satisfactory recommendations.

As a member of this development team, how would you attempt to satisfy the requests of the standardization Committee?

You will, hopefully, be relieved to know that copies of the draft definition of Topsy++ have been made available to you, and will also be available during the reportback meeting. Besides this, machine readable copies of a prototype implementation of Topsy++ are (and will be) available, as well as copies of Coco/R (the tool used to develop this implementation), and documentation on the use of Coco/R. Lastly, the Committee have supplied some fragmentary specimen programs written in Topsy++ that highlight their concerns.

Your suggestions can be presented to the Committee in machine readable form, or you might elect to present a hand written report, showing how the Coco/R specification and any supporting modules need alteration.

As members of the team you have, naturally, been allowed to interact with one another before the 24 hour deadline is reached, but each one of you will be called upon to present your suggestions to the Committee individually at their next 3 hour session, whereafter judgement will be passed on your efforts.

SECTION C [30 marks]

C1 *Answer either this question or C2 but not both.*

This question relates to the Topsy++ system as described in section B.

To their horror, if and when the development team tackle the challenges posed by the Topsy++ standards committee, they will also discover that the programmers responsible for the prototype implementation had shirked their responsibilities. They had almost completely omitted to deal with the fact that Topsy++ has three data types - integers, characters, and Boolean, and that these are required to obey various compatibility constraints. Discuss in some detail where these constraints need to be checked, and the form that the checks would take. You are not required to produce a fully detailed answer (although you are free to do so), but you should answer in sufficient detail to make it clear that you understand the problem and how to solve it.

C2 *Answer either this question or C1 but not both.*

Suppose that one wishes to introduce the regular procedure concept into an imperative programming language on the lines of that used in Modula-2 or Pascal, where procedures may be nested, declare their own local variables, call one another (or even recursively, themselves), communicate by means of parameters, and access variables that are in scope. What particular features are needed to provide compile-time and run-time support for these ideas? Discuss these features in some detail; it is not, however, necessary to provide detailed code for a compiler.