IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2015

EIE PART II: MEng, BEng and ACGI

LANGUAGE PROCESSORS

Corrected Copy

Friday, 12 June 2:00 pm

Time allowed: 2:00 hours

There are THREE questions on this paper.

Answer ALL questions.
Q1 carries 40% of the marks. Questions 2 and 3 carry equal marks (30% each).

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : D.B. Thomas
Second Marker(s) : J.V. Pitt

LANGUAGE PROCESSORS

Ensure throughout that your written characters are unambiguous, especially in terms of '*' versus '+' and white-space. If necessary, use a square under-bracket to indicate space characters.

1. a) Define the terms AST and IR, and briefly explain the role of each in a compiler. [3]

b) What are the four levels of Chomsky's hierarchy? [3]

c) A grammar can be defined as a tuple (V_T, V_N, P, s). Give a name for each element of the tuple which describes its role. [3]

d) What are the conditions for a grammar to be left-linear? [3]

e) Define synthesised versus inherited attributes. [4]

f) Describe an ε transition, and explain why it is desirable to eliminate them from grammars before constructing a digital state machine. [4]

g) What is an LL(1) grammar, and why are they considered efficient? [4]

h) Explain the worst-case behaviour of a top-down versus bottom-up parser when processing sentences from this grammar:

```
A ::= 'x' A 'y'
    | 'x' A 'z'
    | 'c'
```

[4]

i) Describe the role and signature (input and output types) of the transition function in a PDA. [4]

j) How can register allocation be modelled as graph colouring? [4]

k) Define a basic block. [4]

2. A configuration language is needed for describing the dependencies between different software packages. A brain-storming session has resulted in the following language example, which needs to be turned into a grammar and parser.

```
package libc {
  description : [C standard library];
  version     : 3.7.2 ;
  source      : http://source.gnu.org/gcc/3.7.2/src.tar.gz ;
};
package gcc {
  description : [The GNU C compiler];
  version     : 3.7.2;
  dependsOn   : libc:3.7.2 ;
  source      : ftp://ftp.gnu.org/gcc/3-7-2/sources.tar.gz ;
  binary[x86]  : ftp://ftp.gnu.org/gcc/3-7-2/ia32.tar.gz ;
  binary[mips] : ftp://ftp.gnu.org/gcc/3-7-2/mips.tar.gz ;
};
package go {
  version     : 1.3;
  binary[x86] : http://google.com/go/Releases/FAE21A787FFE17/x86.tar.bz2;
  dependsOn   : libjpg:3.1.2;
  binary[x86_64] : http://google.com/go/Releases/FAE21A787FFE17/x86_64.tar.bz2;
  dependsOn   : libgocore:1.3 ;
description : [The Go compiler];
};
```

Currently the only legal architectures are x86, x86_64, mips, and arm, though it is intended that more can be supported as needed.

a) Given that only ftp and http are allowed, and that all sources must be located at a .com or .org address, give a regular expression for matching legal URLs used in this language. Make sure your written characters are easily readable and unambiguous. [5]

b) Define terminal symbols for this language, giving the names of each terminal and the associated regular expression. Keywords such as "package" and "source" do not need to be defined. You can refer to the previous answer where needed, rather than writing out the regex again. [5]

c) Give a grammar for the language, with a start symbol called REPOSITORY. This grammar does not need to be efficient, but it should be unambiguous and capture the language. If necessary, describe any assumptions or restrictions you made. [6]

d) Design an AST for this language using C or C++. The AST should be designed to be constructed from a parser, then manipulated and queried. [6]

e) Describe the language using grammar rules compatible with the bison parser generator, including semantic actions needed to build the AST. This does not need to use the same grammar you gave in part c). [8]

3. A simple digital circuit is connected to the world via a UART (character device) and processes read and write requests encoded as streams of characters. Addresses precede data in the stream and both are encoded as variable length decimal numbers.

The two transaction types are:

- Read request: `A[0-9]+R`
- Write request: `A[0-9]+W[0-9]+`

Both requests start with a base Address; R will cause data to be read from that address and sent to another channel (not seen here); while W indicates a piece of data to write at that address.

a) Give one example each of a valid read and write request. [2]

b) Use a diagram to show how Thompson's algorithm is used to construct an NFA for the sequence AB, where A and B are arbitrary regular expressions. [3]

c) Rewrite the read request using the Kleene star. [3]

d) Construct the NFA for a read request using Thompson's algorithm. [6]

USING THE REGEX FROM 3c

e) Define the ε-closure function, and apply it to the nodes in the NFA you created in part e). [5]

f) Give the name of the general algorithm used to convert an NFA to a DFA, and describe it using pseudo-code. [7]

g) A request can be either a read request or a write request. Using the algorithm from the previous question, or any other method, generate the DFA for a full request (i.e. either a read or write request). [4]