Candidate Number

G5035

THE UNIVERSITY OF SUSSEX

BSc SECOND YEAR EXAMINATION
January 2019 (A1)

Compilers and Computer Architecture

Assessment Period: January 2019 (A1)

DO NOT TURN OVER UNTIL INSTRUCTED TO BY THE LEAD INVIGILATOR

Candidates should answer TWO questions out of THREE. If all three questions are attempted only the first two answers will be marked.

The time allowed is TWO hours.

Each question is worth 50 marks.

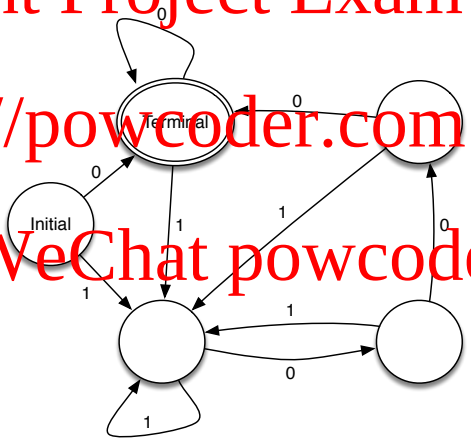
At the end of the examination the question paper and any answer books/answer sheets, used or unused, will be collected from you before you leave the examination room.

1. This question is about lexing and parsing.

   (a) This question is about ASTs (= abstract syntax trees). For each of the following statements, state whether they are true or false. Each correct answer gives 2 marks, each wrong answer gets -1 mark, omitted answers get 0 marks. Overall marks are capped from below to 0. [10 marks]

      i. The AST is constructed by the lexer.
      ii. The AST is constructed by the parser.
      iii. The purpose of constructing an AST is to make later compiler phases (e.g. type-checking and code-generation algorithms) simpler.
      iv. The purpose of constructing an AST is to allow the compiler to generate faster executable machine code.
      v. The AST data type is a list of tokens.

   (b) Consider the following finite state automaton with 5 states, one of them initial and another one terminal:



   Give a regular expression that accepts the same language as the automaton. Don't forget to specify the alphabet. [14 marks]

   (c) Let A be the alphabet $\{a, b\}$. For each of the following statements, state whether they are true or false. Each correct answer gives 1 mark, each wrong answer gets -1 mark, omitted answers get 0 marks. Overall marks are capped from below to 0. [10 marks]

      i. The empty string is in the language of $aaa(a|b)^*$ over A.
      ii. The string aa is in the language of $aaa(a|b)^*$ over A.
      iii. The string $aaabbbbbbbbbbbbbbbbbb$ is in the language of $aaa(a|b)^*$ over A.
      iv. The string $aabbbbbbbbbbbbbbbbbb$ is in the language of $aaa(a|b)^*$ over A.
      v. The string $bbaaaaaaaaaaaaaaaaaa$ is in the language of $aaa(a|b)^*$ over A.

vi. $aaa(a|b)^*$ is a regular expression of the alphabet $\{a, b, c, d\}$.

vii. Let R be a regular expression over some alphabet B. Then the languages of R^* and $(R^*)^*$ are different.

viii. Let R be a regular expression over some alphabet B. Then the languages of R^* and R^*R^* are identical.

ix. Let R be a regular expression over some alphabet B. Then the languages of R^+ and R^+R^+ are identical

x. The language of well-bracketed strings, e.g. $\epsilon, (), ()(), (()), (()(())), ...$ can be given by a regular expression R over the alphabet $\{(,)\}$.

(d) This question is about CFGs (= context free grammars). For each of the following statements, state whether they are true or false. Each correct answer gives 4 marks, each wrong answer gets -2 marks, omitted answers get 0 marks. Overall marks are capped from below to 0.

Consider the following CFGs. In all cases S is the initial variable.

i. The CFG over the alphabet $\{(,)\}$ with reductions

$$S \to (S) \quad | \quad SS$$

True or false: the language of this CFG are exactly the strings over the alphabet $\{(,)\}$ that feature only balanced brackets, including e.g. $\epsilon, (), ()(), (()), (()(())), ...$

ii. The CFG over the alphabet $\{(,)\}$ with reductions

$$S \to (T) \quad | \quad S \to TT \quad | \quad S \to \epsilon \quad | \quad T \to S$$

True or false: the language of this CFG are exactly the strings over the alphabet $\{(,)\}$ that feature only balanced brackets, including e.g. $\epsilon, (), ()(), (()), (()(())), ...$

iii. The CFG over the alphabet $\{(,), +, *, -, 0, 1\}$ with reductions

$$S \to (S) \quad | \quad S + S \quad | \quad S * S \quad | \quad -S \quad | \quad 0 \quad | \quad 1$$

True or false: the CFG is ambiguous.

iv. The CFG over the alphabet $\{(,)\}$ with reductions

$$S \to (T) \quad | \quad S \to TT \quad | \quad S \to \epsilon \quad | \quad T \to S$$

True or false: the CFG is left-recursive.

[16 marks]

2. This question is about code generation.

   (a) When translating conditionals and loops, the generated assembly code (e.g. MIPS, x86 or ARM) contains jumps. For the CPU to be able to execute a jump, the target of a jump must be a valid memory address. However, code generators that emit assembly code typically generate jumps to *symbolic addresses*. Here are two examples in MIPS assembly:

   ```
   b exit

   beq loop_body
   ```

   Answer the following questions about symbolic addresses.

   i. What are the advantages of using symbolic addresses?

   ii. Which program is responsible for translating such symbolic addresses to actual memory addresses?

   iii. How does the code generator create symbolic addresses?

   [10 marks]

   (b) Consider the following program fragment in a Java-like language:


   ```
   class A {
      A f ( int i ) {
         int j = i+1;
         A a = new A();
         return a;
      }
   }
   ```


   Can the local variable `a` in the method `f` be allocated in the activation record (for invocations) of `f`? If you think it can, sketch an appropriate layout for the activation record. If you think that's not possible, explain why not, and where `a` should be held instead. [10 marks]

   (c) In the lectures we used (a variant of) the following simple programming language to explain code-generation.

$$\begin{array}{lcl} E & ::= & n \mid x \mid E + E \mid E - E \\ P & ::= & \text{for } x \;=\; E \text{ to } E \text{ do } P \mid P; P \mid x := E \end{array}$$

   Here x ranges over variables, and n over integers. Design a code generator for the simple language above. The target machine architecture for your code generator is the **stack machine** (as introduced in the lectures). Make sure to explain your design appropriately, including any auxiliary procedures you might be using. Note that your code generator should translate both programs (ranged over by P) and expressions (ranged over by E).

For your help, the commands of the machine language for the stack machine are listed below. [30 marks]

Command	Meaning
Nop	Does nothing
Pop x	Removes the top of the stack and stores it in x
PushAbs x	Pushes the content of the variable x on stack
PushImm n	Pushes the number n on stack
CompGreaterThan	Pops the top two elements off the stack. If the first one popped is bigger than the second one, pushes a 1 onto the stack, otherwise pushes a 0.
CompEq	Pops the top two elements off the stack. If both are equal, pushes a 1 onto the stack, otherwise pushes a 0.
Jump l	Jumps to l
JumpTrue l	Jumps to address/label l if the top of the stack is not 0. Top element of stack is removed.
Plus	Adds the top two elements of the stack, and puts result on stack. Both arguments are removed from stack
Minus	Subtracts the second element of the stack from the top element, and pushes the result on the after popping the top two elements from the stack
Times	Multiplies the top two elements of the stack, and puts result on stack. Both arguments are removed from stack
Divide	Divides the top element of the stack by the second element on the stack, and puts result on stack. Both arguments are removed from stack
Negate	Negates the top element of the stack.

3. This question is about garbage collection (GC).

   (a) What does GC do? How do languages without GC (for example C and C++) deal with the problem GC solves? [10 marks]

   (b) Reachability of memory is an important concept in GC. Explain what it means. [10 marks]

   (c) Explain how a mark-and-sweep garbage collector works. [15 marks]

   (d) Explain how a stop-and-copy garbage collector works. [15 marks]

End of Paper