國 立 交 通 大 學 試 題 紙

九十六學年度第一次
博士班資格考

科目：編輯器設計 (A)　　日期： 97 年 1 月 25 日　第 1 頁 共 1 頁

請" ✓" 明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答冊

1. (20 points) Construct a deterministic, minimal finite state machine for the following regular expression: $(a^+b^*|bc^*ab^+)$. You have to show the construction process, not just the final result.

2. (15 points) Modify the following grammar with the standard associativity and precedence. The resulting grammar must be un-ambiguous.

$S \rightarrow S + S|S - S|S * S|S/S|(S)| - S|S@S|number$

Here $S@S =_{def} S^S$, which means exponentiation. You may make your own assumptions if needed. However, please write down your assumptions clearly.

3. (15 points) Is the following grammar LL(1)? If not, explain why. If so, construct the parse table.

$S \rightarrow E + S|E$

$E \rightarrow id|(S)$

◎請用深黑色鋼筆或原子筆出題

命題老師簽名：

國立交通大學試題紙

九十六學年度第一次
博士班資格考

科目：編輯器設計（B）

請"✓"明　✓ 不可看書　☐ 可看書

1. Syntax-directed Translation (33%)

A one-pass compiler performs analysis and synthesis in one pass such that scanning, parsing, checking, and translation to target machine code are interleaved.

甲、Explain the working of a one-pass compiler that employs action symbols. Specifically, you need to explain the concepts such as semantic records, semantic routines, semantic stack and their relations (e.g. how semantic routines communicate during parsing and semantics processing). (8%)

乙、Point out the principle advantages and disadvantages of one-pass compilers when compared to multi-pass ones. (4%)

丙、Explain briefly why action symbols work more smoothly with LL parsers than with LR parsers. What is the main modification needed to use action symbols in LR parsers? (5%)

Consider the following grammar for simple expressions:

```
<expression> -> <primary> { <op> <primary> }
<primary> -> <number>
<primary> -> <id>
<op> -> PLUS
<op> -> MINUS
<number> -> NUMBER
<id> -> IDENTIFIER
```

Design a one-pass compiler that translates a given expression into a sequence of machine instructions:

丁、Add action symbols into the grammar in some proper places and describe briefly what their corresponding semantic routines do. (8%)

戊、Illustrate the changes of the semantic stack and the generated output, step by step, using the input expression A-B+3. You need to briefly describe the machine instructions (or intermediate representation) you use. (8%)

2. Symbol Tables (8%)

Languages that allow nested name scopes are sometimes known as block-structured languages. To help compilers maintain nested scopes, there are different symbol table implementation strategies.

甲、Describe the approach that creates individual table for each scope; include both the implementation and performance issues. (4%)

乙、Similarly, describe the approach that uses only one single symbol table. (4%)

3. Subprogram Invocation: Saving and Restoring Registers (9%)

Registers in use by a subprogram must be preserved across calls it makes. There are two dimensions of choices when preserving registers: one dimension is that whether the registers that are saved are the ones that the caller is currently using, the ones that the callee might modify, or all. The other dimension is whether the caller or the callee does the saving (i.e. caller-save versus callee-save methods).

甲、What is the main advantage of callee-save methods over caller-save methods? (3%)

乙、What is the main advantage of saving caller's registers over saving callee's registers? (3%)

丙、Describe briefly how to implement the callee-save method that saves the registers used by the caller. What are the considerations? (3%)