CS 324 – Language Processors

MID-SEMESTER EXAMINATION

DATE: 20th September 2003 **Weightage: 20%**
Maximum Marks: 80 **Open Notes Exam Exam**

---

1. The intention behind this question is to test your overview of the area of language processing. Parts of this question relate to topics which have been touched upon, but not covered in details. Therefore, wherever appropriate, I am willing to tolerate certain amount of imprecision in your answers.

   Consider the Pascal program and the compilation of the function `average` shown below:

```
1.  type arr = array [1..10] of integer;
2.  var a  : arr;   sum : integer;

3.  function average(var a: arr; max : integer): integer;
4.  var i : integer;
5.  begin
6.     sum := 0;
7.     for i:= 1 to max do sum := sum + a[i];
8.     average := sum div  max;
9.  end; { average }

10. begin
11.    { the array a is read here }
12.    writeln('the average is', average(a, 10), 'the sum is', sum);
13. end.
```

```
average:
pushl %ebp
movl %esp,%ebp
subl $8,%esp
movw $0,_SUM
movw 12(%ebp),%di
movw %di,-8(%ebp)
movw $1,-4(%ebp)
movw -4(%ebp),%ax
cmpw -8(%ebp),%ax
jg .L8
```

```
.L9:
movl 8(%ebp),%eax
movswl -4(%ebp),%edx
movswl -2(%eax,%edx,2),%eax
movswl _SUM,%edx
addl %eax,%edx
movw %dx,_SUM
movw -4(%ebp),%ax
cmpw -8(%ebp),%ax
jge .L8
incw -4(%ebp)
jmp .L9
```

```
.L8:
movswl _SUM,%eax
movswl 12(%ebp),%edx
movl %edx,%edi
cltd
idivl %edi
movw %ax,-2(%ebp)
movw -2(%ebp),%ax
leave
ret $8
```

   Now answer the following questions:

   (a) Find all lexemes in line 12 and group them according to tokens.

   (b) Write a grammar to express `for` statements whose body consists of a single assignment statement. The assignment statement should be made of simple and (one dimensional) array variables and the operators `:=` and `+`. The bounds of the for statement are variables or constants.

(c) Construct the parse tree for the `for` statement in line 7 using the above grammar.

(d) What part of the parse tree would have been constructed after reading the second assignment operator at line 7, if the parser is a (i) top-down (ii) bottom-up parser?

(e) Show the contents of the symbol table after processing line 4.

(f) Describe how the symboltable would be used to locate the type of `sum` at line 7.

(g) Describe in not more than three lines, what is involved in type checking the statement `sum := sum + a[i]` at line 7.

(h) What would the overloaded operator `+` in line 7 resolve to? How has this information been employed in the generated code?

(i) What addresses have been given to the following items – (i) `max` (ii) `i`, and (iii) `sum`.

(j) What items have been given the following addresses relative to the base pointer of `average` – (i) `ebp + 8` (ii) `ebp - 8`, and (iii) `ebp + 4`?

$(2+5+2+3+4+3+3+2+6+6 = 36$ Marks)

2. Explain why the following Lex patterns are incorrect:

(a) `({DIGIT}*\.{DIGIT}*(e-?{DIGIT}+)?` as representation of real numbers.

(b) `"/*"[^*]*(\*([^/][^*]*)?)*"*/"` as representation of comments.

$(4+5 = 9$ Marks)

3. The grammar $S \to a\ S\ a \mid a\ a$ generates all even length strings of a's except for the empty string. Consider a recursive descent parser with backtracking that tries the $a\ S\ a$ alternative before $a\ a$.

(a) Does the parser succeed on (i) 4 a's (ii) 6 a's?

(b) Let us denote by s_n the n^{th} shortest string accepted by the parser. Now complete the following.

$s_1 =$ ________

$s_n =$ ________ (express in terms of s_{n-1}) $(10+5 = 15 Marks)$

4. A *no* answer to any part of the following question has to be justified by writing a parsing table entry with a conflict. A *yes* answer requires no explanation.

(a) Is the following grammar (i) SLR(1) (ii) LALR(1) (iii) LR(1)?

$S \to d\ M\ N\ c \mid N \mid L$
$N \to d\ b \mid c\ d$
$L \to d\ M\ d\ M$
$M \to b$

(b) Is the following grammar (i) LL(1) (ii) SLR(1) (iii) LALR(1)?

$S \to d\ B \mid A\ d \mid B\ c$
$A \to C\ c$
$B \to a$
$C \to d\ a$

$(7+7 = 14$ Marks)

5. Construct a grammar which is LL(1) but not LALR(1). Explain your answer. (6 Marks)