

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer continue on the back of the page. No notes, books, or other aids may be used on the exam.

Student Id: _____ Answer Key _____

1. (5 points) _____
2. (5 points) _____
3. (10 points) _____
4. (5 points) _____
5. (5 points) _____
6. (10 points) _____
7. (5 points) _____
8. (5 points) _____
- Total (50 points) _____

- [LO3] 1. Write regular expressions for each of the following languages, using only the notation described in class.
- (a) (2 points) An end-of-line comment begins with two forward slashes (//) and continues up to and including the end of line.

Solution: `//[^\n]*\n`

- (b) (3 points) A real literal consists of an integer, a decimal point, a fraction, and an optional exponent, which itself consists of the letter e in either lower or upper case followed by an optional sign and then one or more digits. The integer and fraction each consist of one or more digits but at most one can be optional. For example, 3.14, 3.14e10, .5, .5e+4, 3., and 3.e-10 are all legal, but . alone is not.

Solution: `(([0-9]+.[0-9]*|[0-9]*.[0-9]+)([eE][+-]?[0-9]+)?`

- [LO5] 2. (5 points) Show that the following grammar is ambiguous:

$$\begin{array}{lcl} E & \rightarrow & E + E \\ & | & a \end{array}$$

Solution: The string $a + a + a$ has two different leftmost derivations:

- $E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E + E \Rightarrow a + a + E \Rightarrow a + a + a$
- $E \Rightarrow E + E \Rightarrow E + E + E \Rightarrow a + E + E \Rightarrow a + a + E \Rightarrow a + a + a$

- [LO5] 3. (10 points) Disambiguate the following expression grammar if Δ has the lowest precedence and is left associative, \star and $!$ have the next highest precedence and are left associative, \backslash and \vee have even higher precedence and are right associative, \ominus has the highest precedence and is right associative, and parentheses may be used to override precedence:

$$\begin{array}{lcl}
 E & \rightarrow & \text{num} \\
 & | & (E) \\
 & | & E \Delta E \\
 & | & E \star \\
 & | & E ! \\
 & | & E \backslash E \\
 & | & E \vee E \\
 & | & \ominus E
 \end{array}$$

Do not eliminate left recursion or left factor the grammar.

Solution:

$$\begin{array}{lcl}
 E & \rightarrow & E \Delta F \\
 & | & F \\
 \\
 F & \rightarrow & F \star \\
 & | & F ! \\
 & | & G \\
 \\
 G & \rightarrow & H \backslash G \\
 & | & H \vee G \\
 & | & H \\
 \\
 H & \rightarrow & \ominus H \\
 & | & (E) \\
 & | & \text{num}
 \end{array}$$

4. (5 points) Write a recursive-descent parser for the following grammar:

$$\begin{array}{lcl} A & \rightarrow & a \\ & | & B \\ \\ B & \rightarrow & b B b \\ & | & c \end{array}$$

Assume that you already have the following declarations:

```
int lookahead;
void match(int token);
```

Solution:

```
void A() {
    if (lookahead == 'a')
        match('a');
    else
        B();
}

void B() {
    if (lookahead == 'b') {
        match('b');
        B();
        match('b');
    } else
        match('c');
}
```

5. (5 points) Circle all declarations and uses that are in error in the following Simple C program:

```
1  int x, a[10];
2
3  int f(int x, int y) {
4      int y, *z;
5
6      {
7          int a, b;
8          a = b + g();
9      }
10
11     b = x + y;
12 }
13
14 int *g(int a), a[10];
15 int x, z;
16 int f(int w, int w), z;
17 int *x, z;
```

Solution: Lines 4, 11, 14, 16 and 17 each contain one error.

[LO6] 6. (10 points) Write type expressions for each of the following C declarations.

(a) `int *x;`

Solution: `pointer(int)`

(b) `int (*c)[5];`

Solution: `pointer(array(int, 5))`

(c) `int f(int, char *);`

Solution: `int × pointer(char) → int`

(d) `int *g(char *);`

Solution: `pointer(char) → pointer(int)`

(e) `double *(*d)[10];`

Solution: `pointer(pointer(array(pointer(double), 10)))`

7. (5 points) Indicate whether each of the following statements is true or false. No justification is required.

(a) The < operator is overloaded in Simple C.

Solution: True

(b) The / operator is polymorphic in Simple C.

Solution: False

(c) The % operator is overloaded in Simple C.

Solution: False

(d) The & operator is polymorphic in Simple C.

Solution: True

(e) The - operator is polymorphic in C.

Solution: False

[LO7] 8. (5 points) Consider the following C program:

```
int x, **p;

int f(int a) {
    int y;
    static int *q;

    p = &q;
    q = malloc(a);
}
```

Indicate whether the object named by each of the following expressions is statically allocated, stack allocated, or dynamically allocated.

(a) f

Solution: statically allocated

(b) *p

Solution: statically allocated

(c) **p

Solution: dynamically allocated

(d) *q

Solution: dynamically allocated

(e) a

Solution: stack allocated