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) A character literal consists of one or more characters enclosed in single quotes and may not contain a newline or a single quote.

Solution: `'[^'\n]+'`

(b) (3 points) A real literal consists of one or more digits, a decimal point, one or more digits, 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. However, if the exponent is present then the fractional part of the literal, which includes the decimal point and subsequent digits, need not be present. For example, `3.14`, `3.14e10`, `314e-2`, and `314e2` are all legal, but 123 is not.

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

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

$$\begin{array}{lcl} A & \rightarrow & A\,a \\ & | & a\,A \\ & | & a \end{array}$$

Solution: The sentence aa has two leftmost derivations: $A \Rightarrow Aa \Rightarrow aa$ and $A \Rightarrow aA \Rightarrow aa$.

[LO5] 3. (10 points) Disambiguate the following expression grammar if $*$ and $\circ$ have the lowest precedence and are right associative, $\cup$ has the next highest precedence and is left associative, $\wedge$ and $\oplus$ have even higher precedence and are right associative, ! has the highest precedence and is left associative, and parentheses may be used to override precedence:

$$
\begin{array}{lcl}
E & \rightarrow & \textbf{num} \\
 & | & (\; E \;) \\
 & | & * \; E \\
 & | & \circ \; E \\
 & | & E \cup E \\
 & | & E \wedge E \\
 & | & E \oplus E \\
 & | & E ! \\
\end{array}
$$

Do not eliminate left recursion or left factor the grammar.

Solution:

$$
\begin{array}{lcl}
E & \rightarrow & * \; E \\
 & | & \circ \; E \\
 & | & F \\
 & & \\
F & \rightarrow & F \cup G \\
 & | & G \\
 & & \\
G & \rightarrow & H \wedge G \\
 & | & H \oplus G \\
 & | & H \\
 & & \\
H & \rightarrow & H ! \\
 & | & (\; E \;) \\
 & | & \textbf{num}
\end{array}
$$

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

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

Assume that your already have the following declarations:

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

Solution:

```
void A() {
    match('a');

    if (lookahead == 'b') {
        match('b');
        A();
    } else
        B();
}

void B() {
    match('c');
    match('b');
}
```

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

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

Solution: Lines 4, 11, 14, 18 and 19 each contain one error.

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

(a) `char **p;`

Solution: pointer(pointer(char))

(b) `double a[10];`

Solution: array(double, 10)

(c) `char *b[10];`

Solution: array(pointer(char),10)

(d) `int (*f)(double);`

Solution: pointer(double → int)

(e) `char *(*g)(int *)`

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

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 overloaded in Simple C.

   Solution: False

   (c) The / operator is overloaded in C.

   Solution: True

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

   Solution: True

   (e) The & operator is overloaded in C.

   Solution: True

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

```
int *a;

int f(int x, int **y) {
    static int *b;

    y = &b;
    b = &x;
    a = malloc(x);
}
```

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

(a) `a`

Solution: statically allocated

(b) `*b`

Solution: stack allocated

(c) `y`

Solution: stack allocated

(d) `*a`

Solution: dynamically allocated

(e) `**y`

Solution: stack allocated