COMS W4115
Programming Languages and Translators
Homework Assignment 2

Prof. Stephen A. Edwards
Columbia University

Due Aug 3rd, 2016
at 5:30 PM

Submit your solution on paper at the beginning of class

Write your name and your Columbia ID (e.g., se2007) on your solutions.

Do this assignment alone. You may consult the instructor or the TA, but not other students.

1. Scanners

   (a) Using Ocamllex-like syntax, write a scanner for C's floating point numbers, as defined by Ritchie:

   > A floating constant consists of an integer part, a decimal point, a fraction part, an `e`, and an optionally signed integer exponent. The integer and fraction parts both consist of a sequence of digits. Either the integer part or the fraction part (not both) may be missing; either the decimal point or the `e` and the exponent (not both) may be missing.

   Hint: make sure your scanner accepts constants such as `1. 0.5e-15 .3e+3 .2 1e5` but not integer constants such as `42`

   (b) Draw a DFA for a scanner that recognizes and distinguishes the following set of keywords. Draw accepting states with double lines and label them with the name of the keyword they accept. Follow the definition of a DFA given in class.

   `do double if else elsif uint int for foreach`

2. Construct nondeterministic finite automata for the following regular expressions using Algorithm 3.23 (p. 159, shown in class), then use the subset construction algorithm to construct DFAs for them using Algorithm 3.20 (p. 153, also shown in class).

   (a) $(x \mid yx)^*$

   (b) $((\epsilon \mid y)x)^*$

   (c) $(x \mid y)^* xxy$

   Number the NFA states; use the numbers to label DFA states while performing subset construction, e.g., like Figure 3.35 (p. 155).

3. Using the grammar

$$S \rightarrow (L) \mid a$$
$$L \rightarrow L, S \mid S$$

   (a) Construct a rightmost derivation for $((a), a, a)$ and show the handle of each right-sentential form.

   (b) Show the steps of a shift-reduce (bottom-up) parser corresponding to this rightmost derivation.

   (c) Show the concrete parse tree that would be constructed during this shift-reduce parse.

4. In an assembly-language-like notation (e.g., use MIPS or a pseudocode of your own choosing), write what an optimizing compiler would produce for the following two `switch` statements.

```
switch (a) {
case 4:  z = 2; break;
case 5:  x = 1; break;
case 6:  x = 8; break;
case 8:  x = 17; y = 10; break;
case 9:  y = 3; x = 5; break;
default: z = 5; break;
}

switch (b) {
case 2:    a = 18; break;
case 20:   a = 2; break;
case 108:  b = 7; c = 10;
case 254:  c = 8; break;
default:   c = 17; break;
}
```

(continued on next page)

5. For a 32-bit little-endian processor with the usual alignment rules, show the **memory layout** and **size in bytes** of the following three C variables.

```
union {
  struct {
    short a; /* 16 bits */
    char b;  /* 8 bits */
  } s;
  int c;     /* 32 bits */
} u1;
```

```
struct {
  char a;
  short b;
  short c;
  short d;
  char e;
} s1;
```

```
struct {
  int a;
  short b;
  char c;
  short d;
} s2;
```

6. Consider the following C-like program.

```
int w = 10;
int x = 8;

int incw() { return ++w; }
int incx() { return ++x; }

void foo(y, z){
  printf("%d\n", y + 1 + y);
  x = 7;
  printf("%d\n", z);
}

int main() {
  foo(incw(), incx()); return 0;
}
```

What does it print if the language uses

(a) Applicative-order evaluation?

(b) Normal-order evaluation?