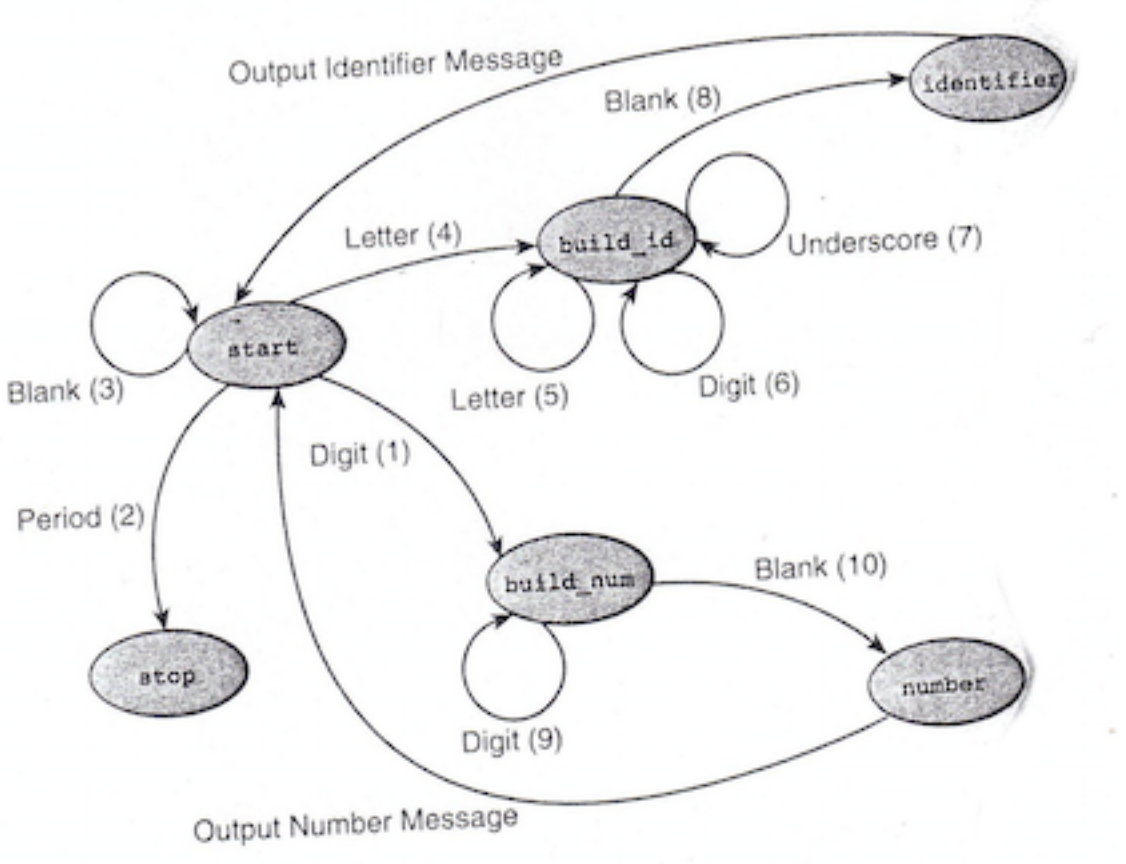
# CIS 343 – Structure of Programming Languages
# Fall 2014, Wednesday, September 24, 2014

## Programming Assignment #2
## A Simple Scanner
## Due Date: Wednesday, October 8, 2014

**Problem Specification**

Write a program in C language that implements a simple scanner based on a finite state machine. The implemented scanner should recognize these tokens – identifiers, numbers, and a selected set of operators.

A finite state machine (FSM) consists of a set of states, a set of transitions, and a string of input data. In the FSM shown below, the named ovals represent states, and the arrows connecting the states represent transitions. The tokens (items) in the input are separated by one or more blanks and a period (.) marks the end of all the data. **Note: Your implementation of the scanner based on the FMS below will earn only 75 points out of possible 100 points.**



The following table traces how this FSM would process this string **“ 95 K9 .”**. The FSM begins in the start state.

| State | Next Character | Transition |
|---|---|---|
| start | ' ' | 3 |
| start | '9' | 1 |
| build_num | '5' | 9 |
| build_num | ' ' | 10 |
| number | | Output number message |
| start | ' ' | 3 |
| start | 'K' | 4 |
| build_id | '9' | 6 |
| build_id | ' ' | 8 |
| identifier | | Output identifier message |
| start | ' ' | 2 |
| stop | | |

Your program must use an enumerated type to represent the names of the states in the FSM. Your program should process (i.e., assume) a correctly formatted data in the input file and identify and display each data item/token and its type. Here is a sample of correct file input and expected output from your program.

**File Input:**

```
rate   R2D2  48       2       time    555666   045  .
```

**Expected Output:**

```
rate – Identifier
R2D2 – Identifier
48 – Number
2 – Number
time – Identifier
555666 – Number
045 – Number
```

Use the code fragment shown below in your `main()` method and design function `transition` to return the next state for all the numbered transitions of the finite state machine.

```
current_state = start
do {
   if (current_state == identifier) {
      printf(“ – Identifier\n”);
      current_state = start;
   } else if (current_state == number) {
```

```
      printf(" – Number\n");
      current_state = start;
   }
   scanf("%c", &transition_char);
   if (transition_char != ' ') {
      printf("%c", transition_char);
   }
   current_state = transition(current_state, transition_char);
} while (current_state != stop);
```

## Add Recognition of Operators (10 points)

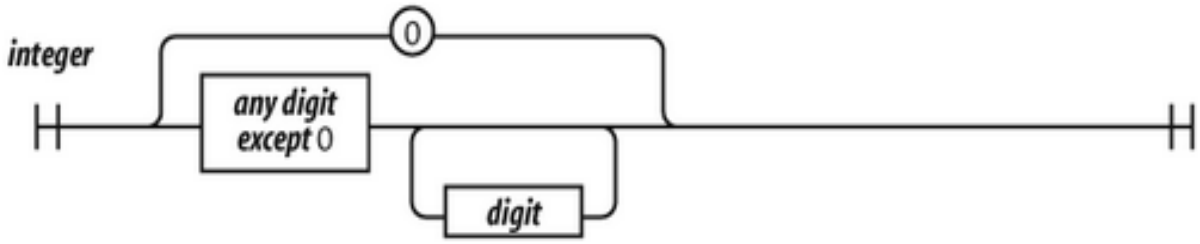
Modify the FSM to handle the following set of six single-character operators: +, -, *, /, =, ;.

Add the following states to the FSM – `addop`, `minusop`, `multop`, `divop`, `assignop`, `stmttermop`.

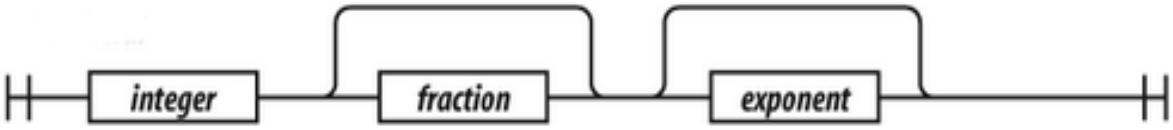
## Add Recognition of Integers and Floating-Point Numbers (15 points)

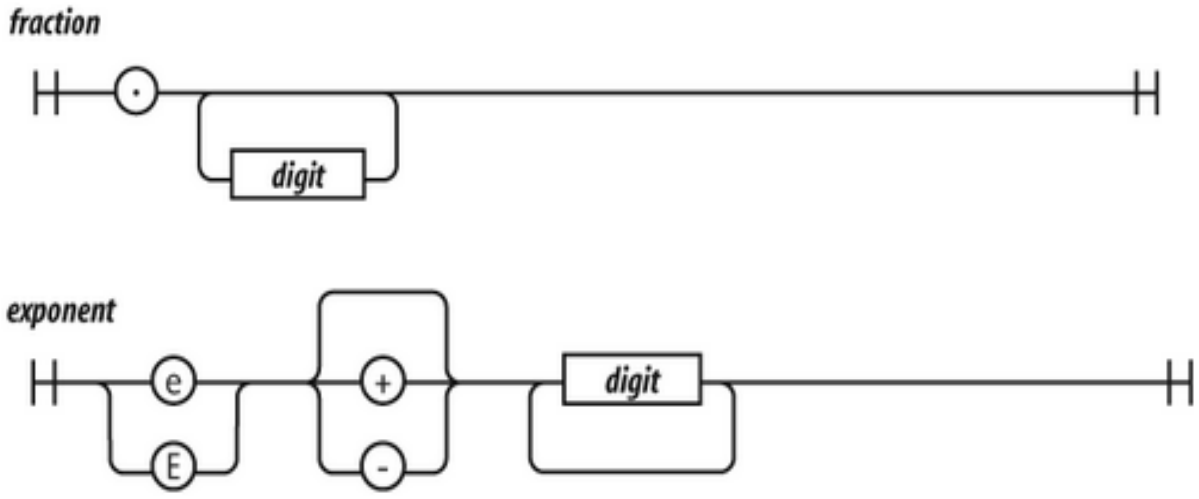
Modify the FSM to handle the integers and floating-point numbers in the input based on the following syntax diagrams.

Add the following two states to the FSM – `integer`, and `floatpointnumber`. Remove the state `number` from FSM.




***Floating-point number***






**Sample Execution**

```
$ ./scanner < input-sample1.txt > output-sample1.txt
```

## Deliverables

1. Upload the source file (`scanner.c`) on Blackboard by midnight on due date.
2. I will use the submission date/time on Blackboard as your official submission date/time.
3. It is your responsibility to make sure the submission on Blackboard went through successfully.
4. Because of possible portability issues, make sure your program compiles and runs on EOS machines before submitting any source file(s) on Blackboard. I will compile, run, and test your program on EOS when grading.
5. Use the following commands on EOS to compile and run your program:
   ```
   $ gcc –Wall –std=c99 scanner.c –o scanner
   $ ./scanner < input-sample1.txt > output-sample1.txt
   ```
6. Late penalty (10% per day) applies after Wednesday, October 8th.