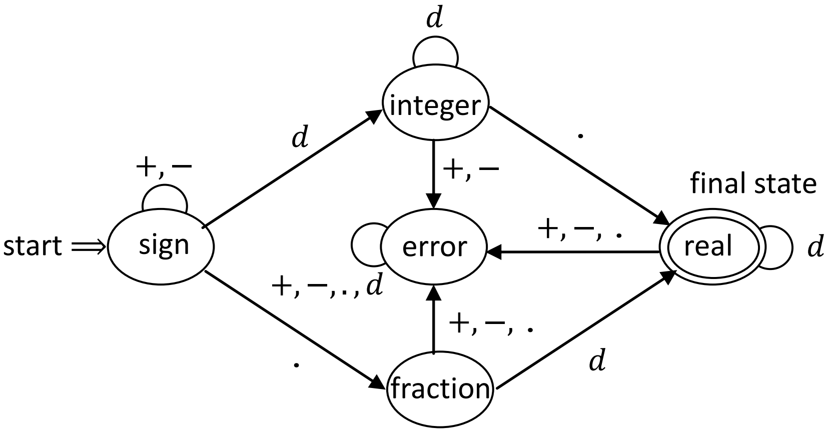
Homework #5

Due date: 12/8

Real constant recognizer

Let $\Sigma = \{+, -, ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ be the alphabet of the language of real constants generated by the following finite automaton, where $d = 0,1,2, \ldots ,9$.



The real constants accepted (or recognized) by this finite automaton are of the form

$$s_1 s_1 \cdots s_m d_1 d_2 \cdots d_n . e_1 e_2 \cdots e_p$$

where s_i is a plus or minus sign, d_j and e_k are decimal digits, and m, n, p satisfy (1) $m \geq 0$ and (2) $n \geq 0, p \geq 0$, but not both zero

For examples, the following real constants are accepted

`12.34` `12.` `.34` `-12.3` `+2.34` `+-+12.34` *

but the following aren't

`1234` `1.2.3` `+1..2` `.+23` `.` `+-+-.` *

Comment

The real constants recognized by this finite automaton are essentially those of C/C++. In particular, the real constants in the red-starred line are also legal in C/C++, and those in the blue-starred line are also illegal in C/C++.

The only difference is that `++` and `--` are consecutive here, but they must be separated by spaces in C/C++. For examples, the finite automaton accepts

`++2.3` `--2.3`

which must be written In C/C++ as

`+ +2.3` `- -2.3`

Your job is to implement the preceding finite automaton in three ways:

1 Represent states as statement labels

2 Represent states as enumerators of an enumeration type, say

```
enum state {sign,integer,fraction,real,error};
```

Determine the next state to transit by computation

3 Use the same representation as method 2

But, build a transition table in advance, and determine the next state to transit by table lookup

Hint: A 5×3 table suffices. (Why?) DO NOT create a 5×13 table.

Hint: Define an inline function to map the 13 symbols $+, -, ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ into 3 array indices.

Requirements

1 You shall write three functions, say

```
void recognizer1(void);     // for method 1
void recognizer2(void);     // for method 2
void recognizer3(void);     // for method 3
```

2 Use the following code to test your functions

```
switch (rand()%3) {
case 0: recognizer1(); break;
case 1: recognizer2(); break;
case 2: recognizer3(); break;
}
```

It is up to you to decide if you want to set a new seed for the pseudorandom number generator.

3 See the sample run for the required output format.

The sample run uses the default seed. The method used to recognize each test datum may be different if a different seed is employed.

Sample run

```
Enter a real constant: 123.45
Accepted by method 3

Enter a real constant: 123.
Accepted by method 3
```

```
Enter a real constant: .45
Accepted by method 2

Enter a real constant: +23.456
Accepted by method 2

Enter a real constant: -0.
Accepted by method 3

Enter a real constant: +.0
Accepted by method 2

Enter a real constant: ++--23.45
Accepted by method 1

Enter a real constant: 1234
Rejected by method 1

Enter a real constant: 1..2
Rejected by method 2

Enter a real constant: 1.2.3
Rejected by method 3

Enter a real constant: +12.+34
Rejected by method 3

Enter a real constant: +123.45+
Rejected by method 3

Enter a real constant: .
Rejected by method 2

Enter a real constant: +-+-.
Rejected by method 1

Enter a real constant: ^Z
```