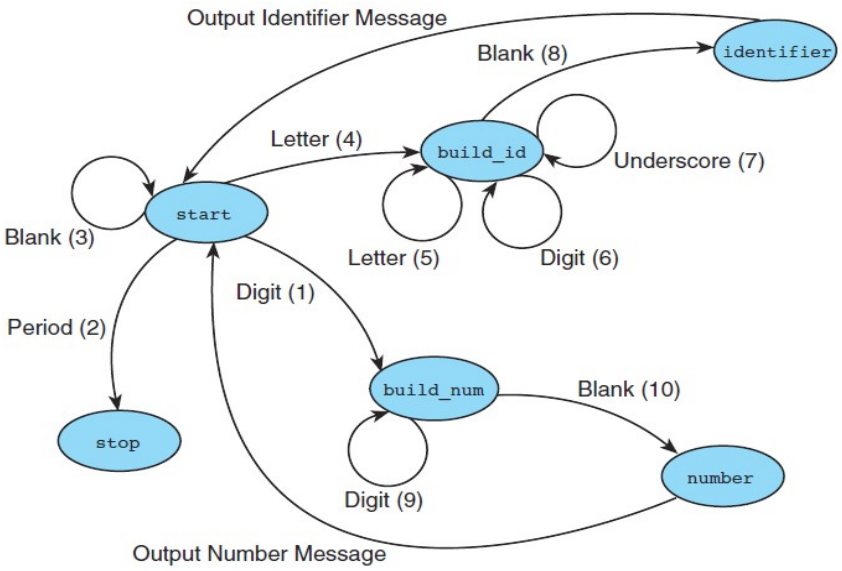
Assignment-4: Design of lexical analyzer

A finite state machine (FSM) consists of a set of states, a set of transitions, and a string of input data. In the FSM of Fig. 6.19, the named ovals represent states, and the arrows connecting the states represent transitions. The FSM is designed to recognize a list of C identifiers and nonnegative integers, assuming that the items are ended by one or more blanks and that a period marks the end of all the data. The following table traces how the diagrammed machine would process a string composed of one blank, the digits 9 and 5, two blanks, the letter K, the digit 9, one blank, and a period. The machine begins in the start state.

FIGURE 6.19

Finite State Machine for Numbers and Identifiers



Trace of Fig. 6.19 FSM on data " 95 K9 ."

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		

Write a program that uses an enumerated type to represent the names of the states. Your program should process a correctly formatted line of data, identifying each data item. Here is a sample of correct input and output.

Input : rate R2D2 48 2 time 555666 .

Output : rate – Identifier

R2D2 – Identifier

48 – Number

2 – Number

time – Identifier

555666 – Number

Use the following code fragment in main, and design function **transition** to return the next state for all the numbered transitions of the finite state machine. If you include the header file **ctype.h**, you can use the library function **isdigit** which returns 1 if called with a digit character, 0 otherwise. Similarly, the function **isalpha** checks whether a character is a letter. When your program correctly models the behaviour of the FSM shown, extend the FSM and your program to allow optional signs and optional fractional parts (i.e., a decimal point followed by zero or more digits) in numbers.

```
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);
```