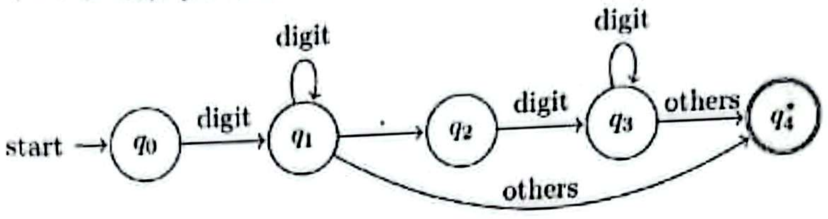
Compiler Construction

Midterm Exam Solution

Q 1: Lexical Analyzer Transition Diagrams: Consider the transition diagram shown below, which recognizes numbers. Write a C function to implement the transition diagram for recognizing numbers. In the diagram, a `digit` represents any numeric character (i.e., `[0-9]`). **[10 Marks]**



Transition Diagram of numbers

Solution

Compiler Link: https://www.onlinegdb.com/

```
#include < iostream >
using namespace std;

bool isNum(const char * str) {
  int state = 0;
  for (int i = 0; str[i] != '\0'; i++) {  →   loop (-3) iff not
    char ch = str[i];                              present
    if (state == -1 || state == 4) break;
    switch (state) {
      case 0:
        if (isdigit(ch))
          state = 1;
        else                         } 2
          state = -1;
        break;

      case 1:
        if (isdigit(ch))
          state = 1;
        else if (ch == '.')          } 3
          state = 2;
        else
          state = 4;
        break;

      case 2:
        if (isdigit(ch))             } 2
          state = 3;
        else
```

Q 2: Give output of a lexical analyzer for the following C code: **[10 Marks]**

```
for (int i = 0; i <= 10; i++) {   2
    char letter,,   2
    num int;   2
    num = 12 34 56;   2
    printf('Hello world!');   2
}
```

=> iff tables present

Solution:

Line	Tokens
1	< *keyword*, 1 > < (> < *keyword*, 2 > < *id*, 1 > <=> < 0 > <;> < *id*, 1 > < ≤ > < 10 > < ;> < *id*, 1 > < ++> <) > < {> 2
2	< *keyword*, 3 > < *id*, 2 > <,> <,> 2
3	< *id*, 3 > < *keyword*, 2 > <;> 2
4	< *id*, 3 > <=> < 12 > < 34 > < 56 > <;> 2
5	< *keyword*, 4 > < (> < *literal*, 1 > <) > <;> 2
6	<}> ✓

Supporting Tables:

Symbol Table	
Position	Lexeme
1	i
2	letter
3	num

Reserved Word Table	
Position	Lexeme
1	for
2	int
3	char
4	printf

Literal Table	
Position	Lexeme
1	Hello world!

Q 3: Lexical Analyzer Regular Expressions: Give a transition diagram and a regular expression for the following token: An identifier is a string of letters and digits. It starts with a letter, and contains an odd number of digits **[5+5 Marks]**

Solution

Regular Expression / Regular Definition

$letter\_ = [a - z\ A - Z\ \_]$

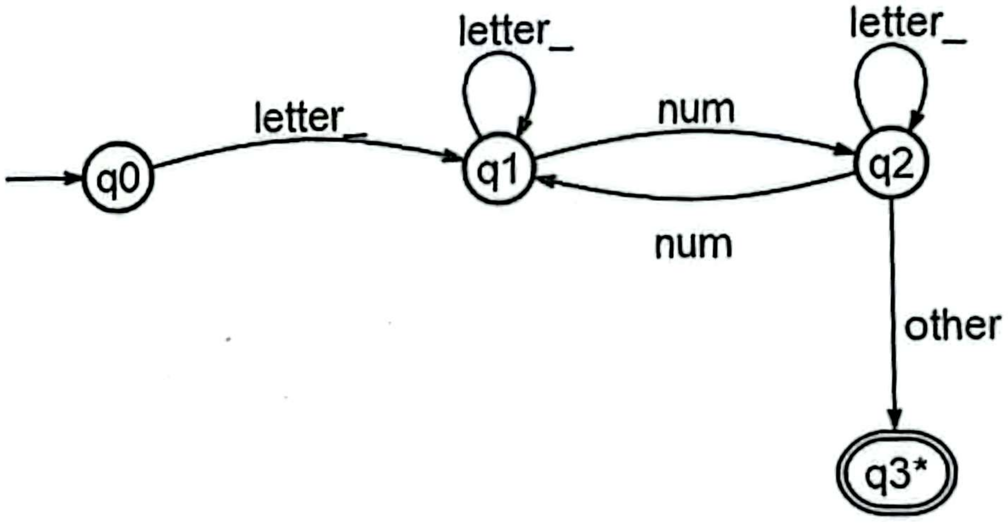
$num = [0 - 9]$

$Identifier = \ letter\_^{*}(letter\_^{*}\ num)\ (\ letter\_^{*} \mid (num\ \ letter\_^{*}\ num)^{*})^{*}$

] binary

0/5

Finite Automata / Transition Diagram



- $\Sigma_{letter\_} = \{a, b, c, \ldots z, A, B, C, \ldots Z, \_\}$
- $\Sigma_{num} = \{0, 1, 2, \ldots 9\}$
- $\Sigma_{id} = \Sigma_{letter\_} \cup \Sigma_{num}$
- $other = \Sigma_{c} - \Sigma id$

$Language_{Accept} = \{num1,\ num111,\ \ldots\}$

```
                state = −1;
              break;

            case 3:
              if (isdigit(ch))
                state = 3;
              else
                state = 4;      } 2
              break;
          }
      }

      switch (state){
        case −1: return false;
        case  1: return true;
        case  3: return true;   } 1
        case  4: return false;
      }
      return false;
  }

  int main() {
    const char * test1 = "123";
    const char * test2 = "12.34";
    const char * test3 = "12.";
    const char * test4 = ".34";
    const char * test5 = "12.34.56";

    cout << (isNum(test1) ? "True" : "False") << endl;
    cout << (isNum(test2) ? "True" : "False") << endl;
    cout << (isNum(test3) ? "True" : "False") << endl;
    cout << (isNum(test4) ? "True" : "False") << endl;
    cout << (isNum(test5) ? "True" : "False") << endl;

    return 0;
  }
```