

Source File: ~/4301/09/lab09. (C|CPP|cpp|c++|cc|cxx|cp)
Input: Under control of main function
Output: Under control of main function
Value: 3

For $\Sigma = \{a, b, c, =, \sqcup, 0, 1\}$, construct a Turing machine for the following:

$$\left\{ \begin{array}{l} (x =, x = ans) \mid x \in \{a, b, c\}^+, \\ ans = \begin{cases} 1, & \text{if the number of } a\text{'s plus the number of } b\text{'s} \\ & \text{plus twice the number of } c\text{'s is divisible by six} \\ 0, & \text{otherwise} \end{cases} \end{array} \right\}$$

A header file is shown in Figure 1, a sample main function for testing your implementation is shown in Figure 2, and a sample execution sequence is shown in Figure 3. To use the Makefile as distributed in class, add a target of lab09 to targets2srcfiles.

Additional notes:

- As each input line is read in, the main function creates the initial tape contents as follows: ten blanks followed by the original input line followed by more blanks.
- The halt state is 0 and the start state is 1.

```

1  #ifndef TURING_H
2  #define TURING_H
3
4  #include <iostream>
5  #include <string>
6  #include <map>
7
8  using namespace std;
9
10 class TableEntry
11 {
12 public:
13     TableEntry(char readChar, char writeChar, int move, uint nextState)
14     {
15         setReadCharacter(readChar);
16         setWriteCharacter(writeChar);
17         setMove(move);
18         setNextState(nextState);
19     }
20     void setReadCharacter(char ch)
21     {
22         readCharacter = ch;
23     }
24     void setWriteCharacter(char ch)
25     {
26         writeCharacter = ch;
27     }

```

Figure 1. /usr/local/4301/include/turing.h (Part 1 of 3)

```
28 void setMove(int moveAmt)
29 {
30     move = moveAmt;
31 }
32 void setNextState(uint state)
33 {
34     nextState = state;
35 }
36 char getReadCharacter() const
37 {
38     return readCharacter;
39 }
40 char getWriteCharacter() const
41 {
42     return writeCharacter;
43 }
44 int getMove() const
45 {
46     return move;
47 }
48 uint getNextState() const
49 {
50     return nextState;
51 }
52 private:
53     char readCharacter;
54     char writeCharacter;
55     int move;
56     uint nextState;
57 };
58
59 class Turing
60 {
61 public:
62     // default constructor -- initializes private data members name,
63     // labNumber, and description
64     Turing();
65     // Member function InitializeMachine() initializes the private data
66     // member machine, a multimap where the key is the current state and
67     // the value is a class object containing the read character, write
68     // character, tape move, and next state
69     void initializeMachine();
70     // Member function OutputID() writes name, class, lab number, and
71     // lab description to output stream out
72     void outputID(ostream& out) const;
73     // Member function ImplementTuring() executes the machine on the
74     // given tape
75     void implementTuring(string& tape) const;
```

Figure 1. /usr/local/4301/include/turing.h (Part 2 of 3)

```
76 private:
77     string name;
78     int labNumber;
79     string description;
80     multimap<uint, TableEntry> machine;
81 };
82
83 #endif
```

Figure 1. /usr/local/4301/include/turing.h (Part 3 of 3)

```
1 #include <turing.h>
2
3 using namespace std;
4
5 int main()
6 {
7     Turing myTuring;
8     string dataLine, tape;
9
10    myTuring.initializeMachine();
11    myTuring.outputID(cout);
12
13    while (getline(cin, dataLine))
14    {
15        cout << "Input: " << dataLine << endl;
16        tape = "          " + dataLine;
17        tape.resize(3 * tape.length(), ' ');
18        cout << "Output: ";
19        myTuring.implementTuring(tape);
20        while (tape.back() == ' ')
21            tape.pop_back();
22        cout << tape << endl << endl;
23    }
24
25    return 0;
26 }
27
28 void Turing::outputID(ostream& out) const
29 {
30     out << name << endl;
31     out << "CS 4301" << endl;
32     out << "Lab " << labNumber << endl;
33     out << description << endl << endl;
34 }
35
```

Figure 2. /usr/local/4301/src/lab09main.C (Part 1 of 2)

```
36 void Turing::implementTuring(string& tape) const
37 {
38     int currentState = 1;
39     string::iterator tapeItr = tape.begin();
40     multimap<uint, TableEntry>::const_iterator turingItr;
41
42     while (currentState > 0)
43     {
44         // Use find to return an iterator to the first entry with a key of
45         // currentState
46         turingItr = machine.find(currentState);
47         if (turingItr != machine.end()) // found a key of currentState
48         {
49             while (turingItr != machine.upper_bound(currentState) &&
50                 turingItr->second.getReadCharacter() != *tapeItr)
51                 ++turingItr;
52             if (turingItr != machine.upper_bound(currentState))
53             {
54                 currentState = turingItr->second.getNextState();
55                 *tapeItr = turingItr->second.getWriteCharacter();
56                 tapeItr += turingItr->second.getMove();
57             }
58             else
59             {
60                 currentState = -1;
61                 *tapeItr = '~';
62             }
63         }
64         else
65         {
66             currentState = -1;
67             *tapeItr = '~';
68         }
69     }
70 }
```

Figure 2. /usr/local/4301/src/lab09main.C (Part 2 of 2)

```
1 newuser@csunix ~> cd 4301
2 newuser@csunix ~/4301> ./getlab.ksh 09
3 * Checking to see if a folder exists for Lab 09. . .No
4 * Creating a folder for Lab 09
5 * Checking to see if Lab 09 has sample input and output files. . .Yes
6 * Copying input and output files for Lab 09
7   from folder /usr/local/4301/data/09 to folder ./09
8 * Checking to see if /usr/local/4301/src/lab09main.C exists. . .Yes
9 * Copying file /usr/local/4301/src/lab09main.C to folder ./09
10 * Checking to see if /usr/local/4301/include/lab09.h exists. . .No
11 * Copying file /usr/local/4301/src/Makefile to folder ./09
12 * Adding a target of lab09 to targets2srcfiles
13 * Touching file ./09/lab09.cpp
14 * Edit file ./09/lab09.cpp in Notepad++
15 newuser@csunix ~/4301> cd 09
16 newuser@csunix ~/4301/09> ls
17 00.dat      00.out      Makefile    lab09.cpp   lab09main.C
18 newuser@csunix ~/4301/09> make lab09
19 g++ -g -Wall -std=c++11 -c lab09main.C -I/usr/local/4301/include -I.
20 g++ -g -Wall -std=c++11 -c lab09.cpp -I/usr/local/4301/include -I.
21 g++ -o lab09 lab09main.o lab09.o -L/usr/local/4301/lib -lm
22 newuser@csunix ~/4301/09> cat 00.dat
23 aaaaaa=
24 bbbbbb=
25 ccc=
26 babac=
27 caabb=
28 bcbaa=
29 abcba=
30 ccaa=
31 cabc=
32 ccccc=
33 aacbcccc=
34 cccccccc=
35 aaaaaaaaaaabbcccccccc=
36 aaabbb=
37 =
38 a=
39 b=
40 aa=
41 ab=
42 ba=
43 bb=
44 aaa=
45 bbb=
46 abba=
47 abcb=
48 abbbbaa=
```

Figure 3. Commands to Compile, Link, & Run Lab 09 (Part 1 of 3)

```
49 newuser@csunix ~/4301/09> cat 00.dat | ./lab09
50 Your Name
51 CS 4301
52 Lab 9
53 {(x=, x=ans) | x is in {a,b,c}+,
54         ans = 1 if the number of a's plus the number of b's
55         plus twice the number of c's is divisible by six and
56         ans = 0 otherwise}
57
58 Input:  aaaaaa=
59 Output:          aaaaaa=1
60
61 Input:  bbbbbb=
62 Output:          bbbbbb=1
63
64 Input:  ccc=
65 Output:          ccc=1
66
67 Input:  babac=
68 Output:          babac=1
69
70 Input:  caabb=
71 Output:          caabb=1
72
73 Input:  bcbaa=
74 Output:          bcbaa=1
75
76 Input:  abcba=
77 Output:          abcba=1
78
79 Input:  ccaa=
80 Output:          ccaa=1
81
82 Input:  cabc=
83 Output:          cabc=1
84
85 Input:  ccccc=
86 Output:          ccccc=1
87
88 Input:  aacbccc=
89 Output:          aacbccc=1
90
91 Input:  cccccccc=
92 Output:          cccccccc=1
93
94 Input:  aaaaaaaaaabbbbbbbbb=
95 Output:          aaaaaaaaaabbbbbbbbb=1
96
97 Input:  aaabb=
98 Output:          aaabb=1
99
```

Figure 3. Commands to Compile, Link, & Run Lab 09 (Part 2 of 3)

```
100 Input:  =
101 Output:      =0
102
103 Input:  a=
104 Output:      a=0
105
106 Input:  b=
107 Output:      b=0
108
109 Input:  aa=
110 Output:      aa=0
111
112 Input:  ab=
113 Output:      ab=0
114
115 Input:  ba=
116 Output:      ba=0
117
118 Input:  bb=
119 Output:      bb=0
120
121 Input:  aaa=
122 Output:      aaa=0
123
124 Input:  bbb=
125 Output:      bbb=0
126
127 Input:  abba=
128 Output:      abba=0
129
130 Input:  abcb=
131 Output:      abcb=0
132
133 Input:  abbbbaa=
134 Output:      abbbbaa=0
135
136 newuser@csunix ~/4301/09> cat 00.dat | ./lab09 > my.out
137 newuser@csunix ~/4301/09> diff 00.out my.out
138 newuser@csunix ~/4301/09>
```

Figure 3. Commands to Compile, Link, & Run Lab 09 (Part 3 of 3)