Source File: `~/4301/08/lab08.(C|CPP|cpp|c++|cc|cxx|cp)`
Input: Under control of `main` function
Output: Under control of `main` function
Value: 2

For $\Sigma = \{a, b\}$ and $\Gamma = \{a, b, @\}$, construct a pushdown transducer that accepts the set consisting of

$$\{a^i b^j, c^k |\ i, j > 0 \text{ and } k = \min(i, j)\}$$

Valid strings include `ab,c`; `aab,c`; `abb,c`; `aaab,c`; `aabb,cc`; `abbb,c`; `aaaab,c`; `aaabb,cc`; and `aabbb,cc`; but not ϵ,ϵ; `a`,ϵ; `b`,ϵ; `aa`,ϵ; `ba`,ϵ; `bb`,ϵ; `aba,c`; `baa`,ϵ; `bab`,ϵ; `bba`,ϵ; or `bbb`,ϵ.

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 `lab08` to `targets2srcfiles`.

Additional notes:

- The `main` function appends the character '`%`' to each line as it is read in.
- The halt state is `0` and the start state is `1`.
- For each input string tested, the stack is initialized to '`@`'.

```
1  #ifndef PDT_H
2  #define PDT_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 stackSymbol, char inputSymbol, string pushPop,
14              uint state, char outputSymbol)
15   {
16     setStackSymbol(stackSymbol);
17     setInputSymbol(inputSymbol);
18     setPushPop(pushPop);
19     setNextState(state);
20     setOutputSymbol(outputSymbol);
21   }
22   void setStackSymbol(char ch)
23   {
24     stackSymbol = ch;
25   }
26   void setInputSymbol(char ch)
27   {
28     inputSymbol = ch;
29   }
```

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

```
30    void setPushPop(string s)
31    {
32      pushPop = s;
33    }
34    void setNextState(uint state)
35    {
36      nextState = state;
37    }
38    void setOutputSymbol(char ch)
39    {
40      outputSymbol = ch;
41    }
42    char getStackSymbol() const
43    {
44      return stackSymbol;
45    }
46    char getInputSymbol() const
47    {
48      return inputSymbol;
49    }
50    string getPushPop() const
51    {
52      return pushPop;
53    }
54    uint getNextState() const
55    {
56      return nextState;
57    }
58    char getOutputSymbol() const
59    {
60      return outputSymbol;
61    }
62   private:
63    char stackSymbol;
64    char inputSymbol;
65    string pushPop;
66    uint nextState;
67    char outputSymbol;
68  };
69
70  class PDT
71  {
72   public:
73    // default constructor -- initializes private data members name,
74    // labNumber, and description
75    PDT();
```

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

```
76    // Member function InitializeMachine() initializes the private data
77    // member machine, a multimap where the key is the current state and
78    // the value is a class object containing the stack symbol, input
79    // symbol, push_pop, next state, and output symbol
80    void initializeMachine();
81    // Member function OutputID() writes name, class, lab number, and
82    // lab description to output stream out
83    void outputID(ostream& out) const;
84    // Member function ImplementPDT() returns true if dataLine is
85    // recognized by the PDT as valid and false otherwise
86    bool implementPDT(string dataLine, ostream& out) const;
87   private:
88    string name;
89    int labNumber;
90    string description;
91    multimap<uint, TableEntry> machine;
92  };
93
94  #endif
```

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

```
1   #include <pdt.h>
2   #include <stack>
3
4   using namespace std;
5
6   int main()
7   {
8     PDT myPDT;
9     string dataLine;
10
11    myPDT.initializeMachine();
12    myPDT.outputID(cout);
13
14    while (getline(cin, dataLine))
15    {
16      cout << "Input:  " << dataLine << "  ";
17      dataLine += "%";
18      cout << "Output:  ";
19      if (myPDT.implementPDT(dataLine, cout))
20        cout << "     ** accepted **" << endl << endl;
21      else
22        cout << "     -- NOT accepted --" << endl << endl;
23    }
24
25    return 0;
26  }
```

Figure 2. `/usr/local/4301/src/lab08main.C` (Part 1 of 3)

```
27
28  void PDT::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
36  bool PDT::implementPDT(string dataLine, ostream& out) const
37  {
38    int currentState = 1;
39    string::iterator dataItr = dataLine.begin();
40    multimap<uint, TableEntry>::const_iterator pdtItr;
41    stack<char> pdtStack;
42    bool done;
43
44    pdtStack.push('@');
45
46    while (currentState > 0)
47    {
48      // Use find to return an iterator to the first entry with a key of
49      // currentState
50      pdtItr = machine.find(currentState);
51      if (pdtItr != machine.end()) // found a key of currentState
52      {
53        done = false;
54        while (!done && pdtItr != machine.upper_bound(currentState))
55          if (pdtItr->second.getInputSymbol() == '*' &&
56              (pdtItr->second.getStackSymbol() == '*' ||
57               (!pdtStack.empty() &&
58                pdtItr->second.getStackSymbol() == pdtStack.top())))
59            done = true;
60          else if (pdtItr->second.getInputSymbol() == *dataItr &&
61                   (pdtItr->second.getStackSymbol() == '*' ||
62                    (!pdtStack.empty() &&
63                     pdtItr->second.getStackSymbol() == pdtStack.top())))
64            done = true;
65          else
66            ++pdtItr;
67
68        if (pdtItr != machine.upper_bound(currentState))
69        {
70          if (pdtItr->second.getStackSymbol() == '*' ||
71              (!pdtStack.empty() &&
72               pdtItr->second.getStackSymbol() == pdtStack.top()))
73          {
74            currentState = pdtItr->second.getNextState();
```

Figure 2. `/usr/local/4301/src/lab08main.C` (Part 2 of 3)

```
75            switch (pdtItr->second.getPushPop()[0])
76            {
77            case '+':
78              pdtStack.push(pdtItr->second.getPushPop()[1]);
79              break;
80            case '-':
81              if (pdtStack.empty())
82                currentState = -1;
83              if (pdtItr->second.getPushPop()[1] != pdtStack.top())
84                currentState = -1;
85              pdtStack.pop();
86            }
87            if (pdtItr->second.getInputSymbol() != '*' && *dataItr != '%')
88              ++dataItr;
89            if (pdtItr->second.getOutputSymbol() != '*')
90              out << pdtItr->second.getOutputSymbol();
91          }
92          else
93            currentState = -1;
94        }
95        else
96          currentState = -1;
97      }
98      else
99        currentState = -1;
100   }
101
102   return currentState == 0 && *dataItr == '%' &&
103          pdtStack.size() == 1 && pdtStack.top() == '@';
104 }
```

Figure 2. `/usr/local/4301/src/lab08main.C` (Part 3 of 3)

```
1  newuser@csunix ~> cd 4301
2  newuser@csunix ~/4301> ./getlab.ksh 08
3    * Checking to see if a folder exists for Lab 08. . .No
4    * Creating a folder for Lab 08
5    * Checking to see if Lab 08 has sample input and output files. . .Yes
6    * Copying input and output files for Lab 08
7      from folder /usr/local/4301/data/08 to folder ./08
8    * Checking to see if /usr/local/4301/src/lab08main.C exists. . .Yes
9    * Copying file /usr/local/4301/src/lab08main.C to folder ./08
10   * Checking to see if /usr/local/4301/include/lab08.h exists. . .No
11   * Copying file /usr/local/4301/src/Makefile to folder ./08
12   * Adding a target of lab08 to targets2srcfiles
13   * Touching file ./08/lab08.cpp
14   * Edit file ./08/lab08.cpp in Notepad++
15 newuser@csunix ~/4301> cd 08
```

Figure 3. Commands to Compile, Link, & Run Lab 08 (Part 1 of 4)

```
newuser@csunix ~/4301/08> ls
01.dat       01.out       Makefile     lab08.cpp    lab08main.C
newuser@csunix ~/4301/08> make lab08
g++ -g -Wall -std=c++11 -c lab08main.C -I/usr/local/4301/include -I.
g++ -g -Wall -std=c++11 -c lab08.cpp -I/usr/local/4301/include -I.
g++ -o lab08 lab08main.o lab08.o -L/usr/local/4301/lib -lm
newuser@csunix ~/4301/08> cat 01.dat

a
b
aa
ab
ba
bb
aaa
aab
aba
abb
baa
bab
bba
bbb
aaab
aabb
abbb
aaaab
aaabb
aabbb
abbbb
aaaaab
aaaabb
aaabbb
aabbbb
abbbbb
aaaaaba
aaaabba
aaabbbab
aabbbbaba
abbbbbaabb
newuser@csunix ~/4301/08> cat 01.dat | ./lab08
Your Name
CS 4301
Lab 8
{a^ib^j,c^k | i, j > 0; k = min(i, j)}

Input:    Output:       -- NOT accepted --

Input:  a  Output:       -- NOT accepted --

```

Figure 3. Commands to Compile, Link, & Run Lab 08 (Part 2 of 4)

```
Input:  b  Output:       -- NOT accepted --

Input:  aa  Output:       -- NOT accepted --

Input:  ab  Output:  c     ** accepted **

Input:  ba  Output:       -- NOT accepted --

Input:  bb  Output:       -- NOT accepted --

Input:  aaa  Output:       -- NOT accepted --

Input:  aab  Output:  c     ** accepted **

Input:  aba  Output:  c     -- NOT accepted --

Input:  abb  Output:  c     ** accepted **

Input:  baa  Output:       -- NOT accepted --

Input:  bab  Output:       -- NOT accepted --

Input:  bba  Output:       -- NOT accepted --

Input:  bbb  Output:       -- NOT accepted --

Input:  aaab  Output:  c     ** accepted **

Input:  aabb  Output:  cc     ** accepted **

Input:  abbb  Output:  c     ** accepted **

Input:  aaaab  Output:  c     ** accepted **

Input:  aaabb  Output:  cc     ** accepted **

Input:  aabbb  Output:  cc     ** accepted **

Input:  abbbb  Output:  c     ** accepted **

Input:  aaaaab  Output:  c     ** accepted **

Input:  aaaabb  Output:  cc     ** accepted **

Input:  aaabbb  Output:  ccc     ** accepted **

Input:  aabbbb  Output:  cc     ** accepted **
```

Figure 3. Commands to Compile, Link, & Run Lab 08 (Part 3 of 4)

```
113 Input:  abbbbb  Output:  c     ** accepted **
114
115 Input:  aaaaaba  Output:  c     -- NOT accepted --
116
117 Input:  aaaabba  Output:  cc     -- NOT accepted --
118
119 Input:  aaabbbab  Output:  ccc     -- NOT accepted --
120
121 Input:  aabbbbaba  Output:  cc     -- NOT accepted --
122
123 Input:  abbbbbaabb  Output:  c     -- NOT accepted --
124
125 newuser@csunix ~/4301/08> cat 01.dat | ./lab08 > my.out
126 newuser@csunix ~/4301/08> diff 01.out my.out
127 newuser@csunix ~/4301/08>
```

Figure 3. Commands to Compile, Link, & Run Lab 08 (Part 4 of 4)