

**Source File:** ~/4301/06/lab06.(C|CPP|cpp|c++|cc|cxx|cp)  
**Input:** Under control of main function  
**Output:** Under control of main function  
**Value:** 2

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

$$\left\{ \begin{array}{l} w \mid w \in \{a, b, c\}^* \text{ and} \\ \text{all the } a\text{'s come before the } b\text{'s and} \\ \text{there are the same number of } a\text{'s as } b\text{'s and} \\ \text{arbitrarily many } c\text{'s that can be in front, behind, or among the } a\text{'s and } b\text{'s} \end{array} \right\}$$

Valid strings include  $\epsilon$ , ab, abc, acb, cab, aabb, abcc, acbc, cabc, cacb, ccab, aaabbb, aabbcc, aabcbc, aabccb, aaccbb, accabb, caacbc, ccaabb, aaaabbbb, aaaaabbbbb, and ccacaabcccbbc, but not ac, ba, bc, ca, cb, aabc, abbc, bbcc, aaabb, abbbc, aabbbc, abbbcc, bbbccc, aaabbbbc, aabbbccc, abbbcccc, bbbccccc, aaaabbbbbc, aaabbbccc, aabbbccc, abbbcccc, or bbbcccc.

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 lab06 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 PDA_H
2  #define PDA_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, uint state)
14     {
15         setStackSymbol(stackSymbol);
16         setInputSymbol(inputSymbol);
17         setPushPop(pushPop);
18         setNextState(state);
19     }
20     void setStackSymbol(char ch)
21     {
22         stackSymbol = ch;
23     }

```

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

```
24 void setInputSymbol(char ch)
25 {
26     inputSymbol = ch;
27 }
28 void setPushPop(string s)
29 {
30     pushPop = s;
31 }
32 void setNextState(uint state)
33 {
34     nextState = state;
35 }
36 char getStackSymbol() const
37 {
38     return stackSymbol;
39 }
40 char getInputSymbol() const
41 {
42     return inputSymbol;
43 }
44 string getPushPop() const
45 {
46     return pushPop;
47 }
48 uint getNextState() const
49 {
50     return nextState;
51 }
52 private:
53     char stackSymbol;
54     char inputSymbol;
55     string pushPop;
56     uint nextState;
57 };
58
59 class PDA
60 {
61     public:
62         // default constructor -- initializes private data members name,
63         // labNumber, and description
64         PDA();
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 stack symbol, input
68         // symbol, push_pop, and next state
69         void initializeMachine();
```

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

```
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 ImplementPDA() returns true if dataLine is
74 // recognized by the PDA as valid and false otherwise
75 bool implementPDA(string dataLine) const;
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/pda.h (Part 3 of 3)

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

Figure 2. /usr/local/4301/src/lab06main.C (Part 1 of 3)

```
27 void PDA::outputID(ostream& out) const
28 {
29     out << name << endl;
30     out << "CS 4301" << endl;
31     out << "Lab " << labNumber << endl;
32     out << description << endl << endl;
33 }
34
35 bool PDA::implementPDA(string dataLine) const
36 {
37     int currentState = 1;
38     string::iterator dataItr = dataLine.begin();
39     multimap<uint, TableEntry>::const_iterator pdaItr;
40     stack<char> pdaStack;
41     bool done;
42
43     pdaStack.push('@');
44
45     while (currentState > 0)
46     {
47         // Use find to return an iterator to the first entry with a key of
48         // currentState
49         pdaItr = machine.find(currentState);
50         if (pdaItr != machine.end()) // found a key of currentState
51         {
52             done = false;
53             while (!done && pdaItr != machine.upper_bound(currentState))
54                 if (pdaItr->second.getInputSymbol() == '*' &&
55                     (pdaItr->second.getStackSymbol() == '*' ||
56                      (!pdaStack.empty() &&
57                       pdaItr->second.getStackSymbol() == pdaStack.top())))
58                     done = true;
59             else if (pdaItr->second.getInputSymbol() == *dataItr &&
60                     (pdaItr->second.getStackSymbol() == '*' ||
61                      (!pdaStack.empty() &&
62                       pdaItr->second.getStackSymbol() == pdaStack.top())))
63                 done = true;
64             else
65                 ++pdaItr;
66         }
```

Figure 2. /usr/local/4301/src/lab06main.C (Part 2 of 3)

```
67     if (pdaItr != machine.upper_bound(currentState))
68     {
69         if (pdaItr->second.getStackSymbol() == '*' ||
70             (!pdaStack.empty() &&
71              pdaItr->second.getStackSymbol() == pdaStack.top()))
72         {
73             currentState = pdaItr->second.getNextState();
74             switch (pdaItr->second.getPushPop()[0])
75             {
76                 case '+':
77                     pdaStack.push(pdaItr->second.getPushPop()[1]);
78                     break;
79                 case '-':
80                     if (pdaStack.empty())
81                         currentState = -1;
82                     if (pdaItr->second.getPushPop()[1] != pdaStack.top())
83                         currentState = -1;
84                     pdaStack.pop();
85             }
86             if (*dataItr != '%' && pdaItr->second.getInputSymbol() != '*')
87                 ++dataItr;
88         }
89         else
90             currentState = -1;
91     }
92     else
93         currentState = -1;
94 }
95 else
96     currentState = -1;
97 }
98
99 return currentState == 0 && *dataItr == '%' &&
100        pdaStack.size() == 1 && pdaStack.top() == '@';
101 }
```

Figure 2. /usr/local/4301/src/lab06main.C (Part 3 of 3)

```
1 newuser@csunix ~> cd 4301
2 newuser@csunix ~/4301> ./getlab.ksh 06
3 * Checking to see if a folder exists for Lab 06. . .No
4 * Creating a folder for Lab 06
5 * Checking to see if Lab 06 has sample input and output files. . .Yes
6 * Copying input and output files for Lab 06
7   from folder /usr/local/4301/data/06 to folder ./06
8 * Checking to see if /usr/local/4301/src/lab06main.C exists. . .Yes
9 * Copying file /usr/local/4301/src/lab06main.C to folder ./06
10 * Checking to see if /usr/local/4301/include/lab06.h exists. . .No
11 * Copying file /usr/local/4301/src/Makefile to folder ./06
12 * Adding a target of lab06 to targets2srcfiles
13 * Touching file ./06/lab06.cpp
14 * Edit file ./06/lab06.cpp in Notepad++
15 newuser@csunix ~/4301> cd 06
16 newuser@csunix ~/4301/06> ls
17 01.dat      01.out      Makefile    lab06.cpp   lab06main.C
18 newuser@csunix ~/4301/06> make lab06
19 g++ -g -Wall -std=c++11 -c lab06main.C -I/usr/local/4301/include -I.
20 g++ -g -Wall -std=c++11 -c lab06.cpp -I/usr/local/4301/include -I.
21 g++ -o lab06 lab06main.o lab06.o -L/usr/local/4301/lib -lm
22 newuser@csunix ~/4301/06> cat 01.dat
23
24 ab
25 abc
26 acb
27 cab
28 aabb
29 abcc
30 acbc
31 cabc
32 cacb
33 ccab
34 aaabbb
35 aabbcc
36 aabcbc
37 aabccb
38 aaccbb
39 accabb
40 caabc
41 ccaabb
42 aaaabbbb
43 aaaaabbbbb
44 ccacaabcccbbc
45 ac
46 ba
47 bc
48 ca
49 cb
```

**Figure 3.** Commands to Compile, Link, & Run Lab 06 (Part 1 of 4)

```
50 aabc
51 abbc
52 bbcc
53 aaabb
54 abbbc
55 aabbbc
56 abbbcc
57 bbbccc
58 aaabbbbc
59 aabbbbcc
60 abbbbccc
61 bbbbcccc
62 aaaabbbbbc
63 aaabbbbcc
64 aabbbbbc
65 abbbbccc
66 bbbbcccc
67 newuser@csunix ~/4301/06> cat 01.dat | ./lab06
68 Your Name
69 CS 4301
70 Lab 6
71 {w | w in {a, b, c}* and
72   all the a's come before the b's and
73   there are the same number of a's as b's and
74   arbitrarily many c's that can be in front, behind, or among the a's and b's}
75
76 Input:      Result:  ** accepted **
77
78 Input: ab   Result:  ** accepted **
79
80 Input: abc  Result:  ** accepted **
81
82 Input: acb  Result:  ** accepted **
83
84 Input: cab  Result:  ** accepted **
85
86 Input: aabb Result:  ** accepted **
87
88 Input: abcc Result:  ** accepted **
89
90 Input: acbc Result:  ** accepted **
91
92 Input: cabc Result:  ** accepted **
93
94 Input: cacb Result:  ** accepted **
95
96 Input: ccab Result:  ** accepted **
97
```

**Figure 3.** Commands to Compile, Link, & Run Lab 06 (Part 2 of 4)

```
98 Input: aaabbb Result: ** accepted **
99
100 Input: aabbcc Result: ** accepted **
101
102 Input: aabcbc Result: ** accepted **
103
104 Input: aabccb Result: ** accepted **
105
106 Input: aaccbb Result: ** accepted **
107
108 Input: accabb Result: ** accepted **
109
110 Input: caabcb Result: ** accepted **
111
112 Input: ccaabb Result: ** accepted **
113
114 Input: aaaabbbb Result: ** accepted **
115
116 Input: aaaaabbbbb Result: ** accepted **
117
118 Input: ccacaabcccbbc Result: ** accepted **
119
120 Input: ac Result: -- NOT accepted --
121
122 Input: ba Result: -- NOT accepted --
123
124 Input: bc Result: -- NOT accepted --
125
126 Input: ca Result: -- NOT accepted --
127
128 Input: cb Result: -- NOT accepted --
129
130 Input: aabc Result: -- NOT accepted --
131
132 Input: abbc Result: -- NOT accepted --
133
134 Input: bbcc Result: -- NOT accepted --
135
136 Input: aaabb Result: -- NOT accepted --
137
138 Input: abbbc Result: -- NOT accepted --
139
140 Input: aabbbc Result: -- NOT accepted --
141
142 Input: abbbcc Result: -- NOT accepted --
143
144 Input: bbbccc Result: -- NOT accepted --
145
```

**Figure 3.** Commands to Compile, Link, & Run Lab 06 (Part 3 of 4)



```
146 Input: aaabbbbc Result: -- NOT accepted --
147
148 Input: aabbbbcc Result: -- NOT accepted --
149
150 Input: abbbbccc Result: -- NOT accepted --
151
152 Input: bbbbcccc Result: -- NOT accepted --
153
154 Input: aaaabbbbbc Result: -- NOT accepted --
155
156 Input: aaabbbbcc Result: -- NOT accepted --
157
158 Input: aabbbbccc Result: -- NOT accepted --
159
160 Input: abbbbcccc Result: -- NOT accepted --
161
162 Input: bbbbccccc Result: -- NOT accepted --
163
164 newuser@csunix ~/4301/06> cat 01.dat | ./lab06 > my.out
165 newuser@csunix ~/4301/06> diff 01.out my.out
166 newuser@csunix ~/4301/06>
```

**Figure 3.** Commands to Compile, Link, & Run Lab 06 (Part 4 of 4)