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 $\Sigma = \{a, b\}$, construct a pushdown automata that accepts the set consisting of

$$\{a^{n+1}b^{2n} \mid n \geq 0\}$$

Valid strings include `a`, `aabb`, `aaabbbb`, and `aaaabbbbbb` but not ϵ, `b`, `ab`, `aaba`, `aabba`, `bb`, `abab`, `baaaa`, `bbba`, or `babb`.

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   }
24   void setInputSymbol(char ch)
25   {
26     inputSymbol = ch;
27   }
```

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

```
28    void setPushPop(string s)
29    {
30      pushPop.resize(2);
31      pushPop[0] = s[0];
32      if (s.length() > 1)
33        pushPop[1] = s[1];
34    }
35    void setNextState(uint state)
36    {
37      nextState = state;
38    }
39    char getStackSymbol() const
40    {
41      return stackSymbol;
42    }
43    char getInputSymbol() const
44    {
45      return inputSymbol;
46    }
47    string getPushPop() const
48    {
49      return pushPop;
50    }
51    uint getNextState() const
52    {
53      return nextState;
54    }
55   private:
56    char stackSymbol;
57    char inputSymbol;
58    string pushPop;
59    uint nextState;
60  };
61
62  class PDA
63  {
64   public:
65    // default constructor -- initializes private data members name,
66    // labNumber, and description
67    PDA();
68    // Member function InitializeMachine() initializes the private data
69    // member machine, a multimap where the key is the current state and
70    // the value is a class object containing the stack symbol, input
71    // symbol, push_pop, and next state
72    void initializeMachine();
73    // Member function OutputID() writes name, class, lab number, and
74    // lab description to output stream out
75    void outputID(ostream& out) const;
```

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

```
76    // Member function ImplementPDA() returns true if dataLine is
77    // recognized by the PDA as valid and false otherwise
78    bool implementPDA(string dataLine) const;
79   private:
80    string name;
81    int labNumber;
82    string description;
83    multimap<uint, TableEntry> machine;
84  };
85
86  #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
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
```

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

```
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
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;
```

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

```
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
```

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

```
newuser@csunix ~/4301/06> cat 01.dat
a
aabb
aaabbbb
aaaabbbbbb

b
ab
aaba
aabba
bb
abab
baaaa
bbba
babb
newuser@csunix ~/4301/06> cat 01.dat | ./lab06
Your Name
CS 4301
Lab 6
{a^(n+1)b^2n | n >= 0}

Input:  a  Result:  ** accepted **

Input:  aabb  Result:  ** accepted **

Input:  aaabbbb  Result:  ** accepted **

Input:  aaaabbbbbb  Result:  ** accepted **

Input:    Result:  -- NOT accepted --

Input:  b  Result:  -- NOT accepted --

Input:  ab  Result:  -- NOT accepted --

Input:  aaba  Result:  -- NOT accepted --

Input:  aabba  Result:  -- NOT accepted --

Input:  bb  Result:  -- NOT accepted --

Input:  abab  Result:  -- NOT accepted --

Input:  baaaa  Result:  -- NOT accepted --

Input:  bbba  Result:  -- NOT accepted --

Input:  babb  Result:  -- NOT accepted --

```

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

```
71 newuser@csunix ~/4301/06> cat 01.dat | ./lab06 > my.out
72 newuser@csunix ~/4301/06> diff 01.out my.out
73 newuser@csunix ~/4301/06>
```

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