Christian Mesina
Hugo Romero
Luis Escobar Urrutia

Lab 1

<u>Introduction</u>:

- For this lab we wrote a program that simulates a DFA in C++. The program accepts the language L=(a|b)*abb and L=(a|b)*bba. The input to the DFA is a string; in the course of processing this string character-by-character, the DFA will undergo the specified state transitions. The DFA accepts the string if it is an accepting state when it has consumed its input; otherwise it rejects the string.

<u>Source Code</u>:

```
/*
* Christian Mesina, Hugo Romero, Luis Escobar
* CSE 570 Compilers Lab 1
* Instructor: Dr. Ernesto Gomez
*
**/

#include <iostream>
#include <string>
#include <vector>
using namespace std;
#define ACCEPTING_STATE 3
#define STATES 4
#define SYMBOLS 2
typedef int State;
// Transition table for DFA1: L = (a|b)*abb
static int transitionTable1[STATES][SYMBOLS] = {{1, 0}, {1, 2}, {1, 3},
{1, 0}};
// Transition table for DFA2: L = (a|b)*bba
static int transitionTable2[STATES][SYMBOLS] = {{0, 1}, {0, 2}, {3, 2},
{0, 1}};
// Initial starting state for both DFA1 and DFA2
State getInitialState()
{
  return 0;
}
// Checks if a given state is in its' accepting state
// Returns true if state == 3, otherwise false
bool isFinalState(State state)
{
  return(state == ACCEPTING_STATE) ? true : false;
}
// Determines what the transition code is given an character
```

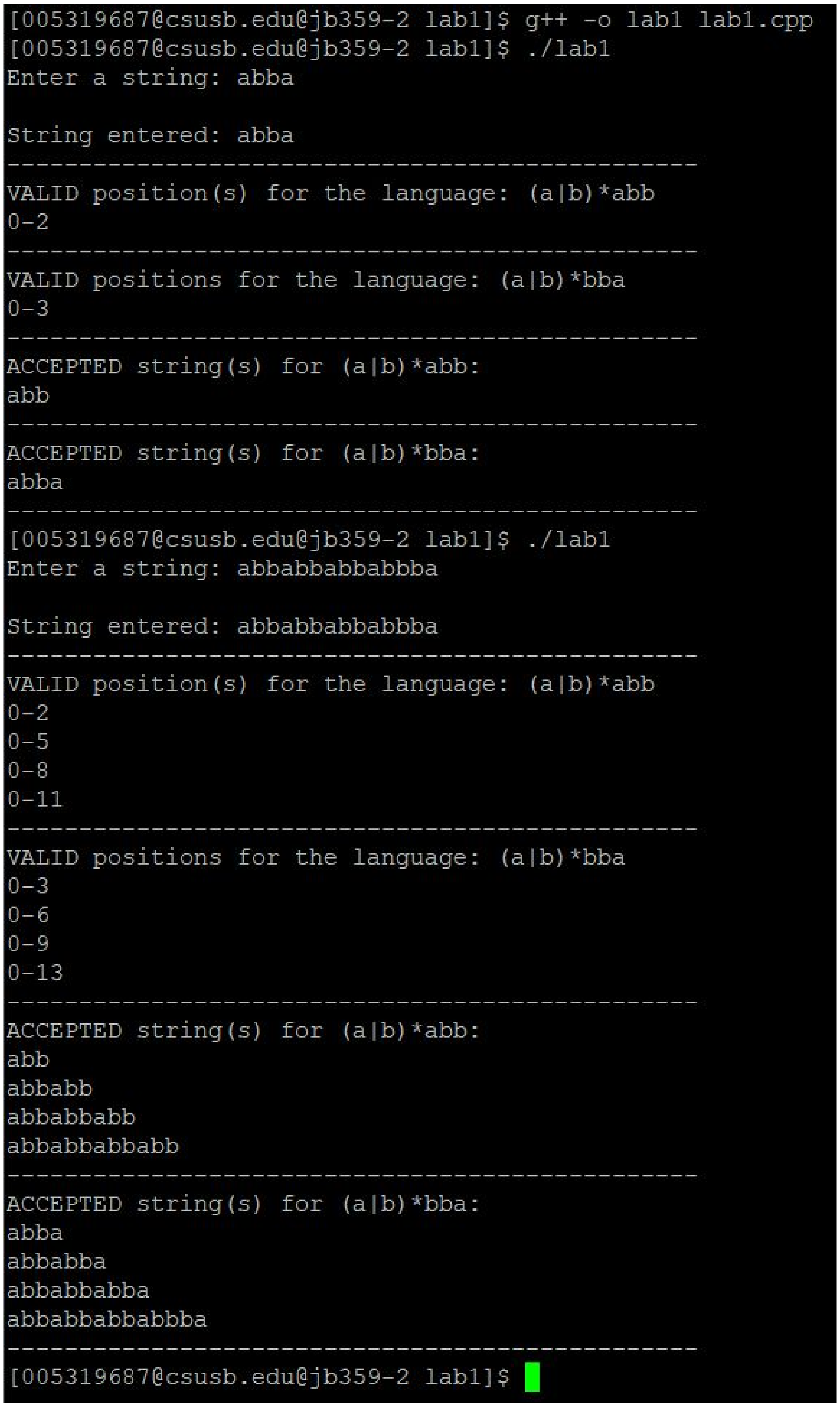
```cpp
int getTransition(char input)
{
  if(input != 'a' && input != 'b')
  {
      return -1;
  }
  switch(input)
  {
  case 'a':
      return 0;
      break;
  case 'b':
      return 1;
      break;
  default:
      return -1;
      break;
  }
}
int main()
{
  const string language1 = "(a|b)*abb"; // Language for DFA1
  const string language2 = "(a|b)*bba"; // Language for DFA2
  string testString, currentString;
  vector<string> acceptedStringsDFA1;
  vector<string> acceptedStringsDFA2;
  vector<pair<int, int>> validPositionsForDFA1;
  vector<pair<int, int>> validPositionsForDFA2;
  cout << "Enter a string: ";
  getline(cin, testString);
  if(testString.empty() || testString.length() < 3)
  {
      cerr << "INVALID string!" << '\n';
      return EXIT_FAILURE;
  }
  cout << '\n';
  cout << "String entered: " << testString << "\n";
  cout << "------------------------------------------------\n";
  State state1 = getInitialState();
  State state2 = getInitialState();
  int startingPosForDFA1 = 0, startingPosForDFA2 = 0;
  int currentPosForDFA1 = 0;
  int currentPosForDFA2 = 0;
  int endPosForDFA1 = 0;
  int endPosForDFA2 = 0;
  for(size_t i = 0; i < testString.length(); i++)
  {
      char input = testString[i];
      int transition = getTransition(input);
      if(transition == -1)
```

```
        {
            state1 = 0;
            state2 = 0;
            startingPosForDFA1 = i + 1;
            startingPosForDFA2 = i + 1;
        }
        else
        {
            currentString += input;
            state1 = transitionTable1[state1][transition];
            state2 = transitionTable2[state2][transition];
            switch(state1)
            {
            case 0:
                startingPosForDFA1 = i;
                break;
            case 1:
            case 2:
                currentPosForDFA1 = i + 1;
                break;
            case 3:
                endPosForDFA1 = currentPosForDFA1;
                break;
            }
            switch(state2)
            {
            case 0:
                startingPosForDFA2 = i;
                break;
            case 1:
            case 2:
                currentPosForDFA2 = i + 1;
                break;
            case 3:
                endPosForDFA2 = currentPosForDFA2;
                break;
            }
        }
        if(isFinalState(state1))
        {
            validPositionsForDFA1.push_back(make_pair(startingPosForDFA1,
endPosForDFA1));
            acceptedStringsDFA1.push_back(currentString);
            state1 = getInitialState();
        }
        else if(isFinalState(state2))
        {
            validPositionsForDFA2.push_back(make_pair(startingPosForDFA2,
endPosForDFA2));
            acceptedStringsDFA2.push_back(currentString);
```

```cpp
            state2 = getInitialState();
        }
    }
    if(acceptedStringsDFA1.empty() && acceptedStringsDFA2.empty())
    {
        cout << "REJECTED for both languages" << '\n';
        return EXIT_FAILURE;
    }
    if(!validPositionsForDFA1.empty() && !acceptedStringsDFA1.empty())
    {
        cout << "VALID position(s) for the language: " << language1 << '\n';
        for(auto v : validPositionsForDFA1)
        {
            cout << v.first << '-' << v.second << '\n';
        }
        cout << "--------------------------------------------------\n";
    }
    if(!validPositionsForDFA2.empty() && !acceptedStringsDFA2.empty())
    {
        cout << "VALID positions for the language: " << language2 << '\n';
        for (auto v : validPositionsForDFA2)
        {
            cout << v.first << '-' << v.second << '\n';
        }
        cout << "--------------------------------------------------\n";
    }
    if(!acceptedStringsDFA1.empty())
    {
        cout << "ACCEPTED string(s) for " << language1 << ":" << '\n';
        for(auto a : acceptedStringsDFA1)
        {
            cout << a << ' ' << "\n";
        }
        cout << "--------------------------------------------------\n";
    }
    else
    {
        cout << "INVALID string(s) for the language: " << language1 << '\n';
        cout << "--------------------------------------------------\n";
    }
    if(!acceptedStringsDFA2.empty())
    {
        cout << "ACCEPTED string(s) for " << language2 << ":" << '\n';
        for(auto a : acceptedStringsDFA2)
        {
            cout << a << ' ' << "\n";
        }
        cout << "--------------------------------------------------\n";
    }
    else
```

```
  {
      cout << "INVALID string(s) for the language: " << language2 << '\n';
      cout << "------------------------------------------------\n";
  }
  return 0;
}
```

Output:



```
[005319687@csusb.edu@jb359-2 lab1]$ g++ -o lab1 lab1.cpp
[005319687@csusb.edu@jb359-2 lab1]$ ./lab1
Enter a string: abba

String entered: abba
------------------------------------------------
VALID position(s) for the language: (a|b)*abb
0-2
------------------------------------------------
VALID positions for the language: (a|b)*bba
0-3
------------------------------------------------
ACCEPTED string(s) for (a|b)*abb:
abb
------------------------------------------------
ACCEPTED string(s) for (a|b)*bba:
abba
------------------------------------------------
[005319687@csusb.edu@jb359-2 lab1]$ ./lab1
Enter a string: abbabbabbabbba

String entered: abbabbabbabbba
------------------------------------------------
VALID position(s) for the language: (a|b)*abb
0-2
0-5
0-8
0-11
------------------------------------------------
VALID positions for the language: (a|b)*bba
0-3
0-6
0-9
0-13
------------------------------------------------
ACCEPTED string(s) for (a|b)*abb:
abb
abbabb
abbabbabb
abbabbabbabb
------------------------------------------------
ACCEPTED string(s) for (a|b)*bba:
abba
abbabba
abbabbabba
abbabbabbabbba
------------------------------------------------
[005319687@csusb.edu@jb359-2 lab1]$
```