Course Name:	**TACD**	**Semester:**	**V**
Date of Performance:	**18 / 10 / 2024**	**Batch No:**	**B - 1**
Faculty Name:	**Prof. Amrita Naiksatam**	**Roll No:**	**16014022050**
Faculty Sign & Date:		**Grade / Marks:**	**___ / 25**

IA - 1

Question: Construct a PDA for the language L = {wcwR | w ϵ{a,b}*}.

Constructing PDA:

TACD IA1
Ketaki Mahajan
B-1
16014022050

19/10/24

TACD IA1 (1)

Q.6. Construct a PDA for the language $L = \{wcw^R \mid w \in \{a,b\}^*\}$.

For example lets take a string w, which is combination of {a,b},

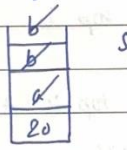
w = abb

w^R = bba

∴ concatenating = abb c bba (pop)

(abb = w, bba = w^R)

push into stack



Skip

- Transitions for the PDA are given by,

$\delta(q_0, a, Z) = (q_0, aZ)$
$\delta(q_0, b, Z) = (q_0, bZ)$
$\delta(q_0, c, Z) = (q_0, \varepsilon)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, a, b) = (q_0, ab)$
$\delta(q_0, b, a) = (q_0, ba)$
$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, c, a) = (q_1, a)$
$\delta(q_0, c, b) = (q_1, b)$
$\delta(q_1, a, a) = (q_1, \varepsilon)$
$\delta(q_1, b, b) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, Z) = (q, \varepsilon)$

- Z is the initial stack symbol.
- q_0 is the initial state.
- String is accepted through an empty stack.

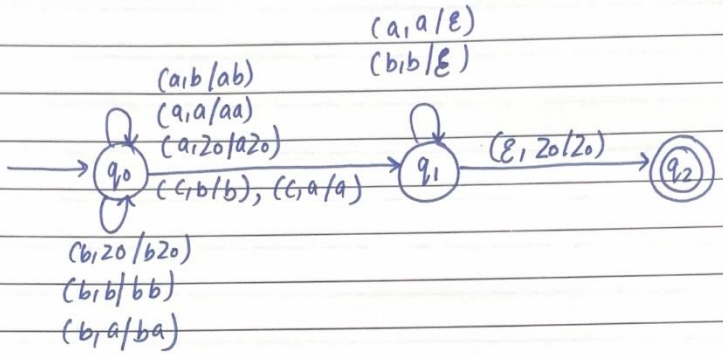
- Rules for implementing FA,

Initial setup : load input string in input buffer, push Z as initial stack symbol & consider the machine in at initial state q_0.

Rules:

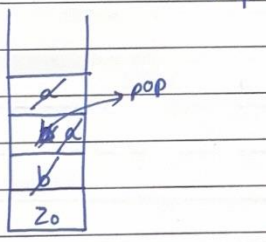
1). IIP symbol is 0 & stack top is Z, push 'A' into stack & read next char.
2). IIP symbol is 1 & stack top is Z, push symbol 'B' into stack & read next char.
3). IIP symbol is 0 & stack top is A/B, push 'A' into stack & reach next char.
4). IIP symbol is 1 & stack top is A/B, push 'B' into stack & read next char.
5). IIP symbol is C, stack top is A or B, change state from q_0 to q_1 & read next char.
6). IIP ~~this~~ symbol is 0, machine state is q_1 & stack top is A then pop stack top & read next char.
7). IIP symbol is 1, machine state is q_1 and stack top is B, pop stack top & read next char.
8). If all characters of input strings are parsed, stack top is Z and machine state is q_1, meaning string is valid, pop Z from stack & change state from q_1 to q_2.

- Transition diagram,



- Example string,

w = baacaab ε



Let PDA machine be defined as $M(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, where,

- Q: set of states $\{q_0, q_1, q_2\}$
- Σ: set of symbols : $\{a, b, c\}$
- Γ: set of stack symbols : $\{A, B, Z_0\}$
- q_0: initial state (q_0)
- z_0: initial stack symbol
- F: set of final state : $\emptyset$ (null as decision is based on if stack is empty or not)
- δ: transition function

Implementation using C++ Code:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char Input[100];
    char stack[100];
    int Top = -1;

    cout << "Enter string to be validated:\n";
    cin.getline(Input, 100);
    stack[++Top] = 'Z';

    int i = -1;

q0:
    i++;
    if (Input[i] == 'a' && stack[Top] == 'Z')
    {
        stack[++Top] = 'A';
        goto q0;
    }
    else if (Input[i] == 'b' && stack[Top] == 'Z')
    {
        stack[++Top] = 'B';
        goto q0;
    }
    else if (Input[i] == 'a' && (stack[Top] == 'A' || stack[Top] == 'B'))
    {
        stack[++Top] = 'A';
        goto q0;
    }
    else if (Input[i] == 'b' && (stack[Top] == 'A' || stack[Top] == 'B'))
    {
        stack[++Top] = 'B';
        goto q0;
    }
    else if (Input[i] == 'c' && (stack[Top] == 'A' || stack[Top] == 'B'))
    {
        goto q1;
    }
    else
```

```
    {
        goto Invalid;
    }

q1:
    i++;
    if (Input[i] == 'a' && stack[Top] == 'A')
    {
        Top--;
        goto q1;
    }
    else if (Input[i] == 'b' && stack[Top] == 'B')
    {
        Top--;
        goto q1;
    }
    else if (Input[i] == '\0' && stack[Top] == 'Z')
    {
        goto Valid;
    }
    else
    {
        goto Invalid;
    }

Valid:
    cout << "\nOutput: Valid String";
    goto exit;

Invalid:
    cout << "\nOutput: Invalid String";
    goto exit;

exit:
    return 0;
}
```

Code Output:

- Test String – baacaab
- Test String – abababbcbbababb
- Test String – aaabbbbcaaabbbb
- Test String – bababcbabab

```
PS C:\Users\admin\OneDrive\Desktop\sem 5\theory of automata> cd "c:\Users\admin\OneDrive\Deskto
p\sem 5\theory of automata\" ; if ($?) { g++ IA1_PDA.cpp -o IA1_PDA } ; if ($?) { .\IA1_PDA }
Enter how many strings you have: 4

Enter string 1 to be validated: baacaab
Output: Valid String

Enter string 2 to be validated: abababbcbbababb
Output: Invalid String

Enter string 3 to be validated: aaabbbbcaaabbbb
Output: Invalid String

Enter string 4 to be validated: bababcbabab
Output: Valid String
PS C:\Users\admin\OneDrive\Desktop\sem 5\theory of automata>
```