Introduction to Theoretical Computer Science

Exercise tasks

Preparation for mid-term exam – part 3

Faculty of Electrical Engineering and Computing
University of Zagreb

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4__	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4__	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4__	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4____	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4____	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**

A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 20

- An LR Parser is given for the following grammar. Use the LR parser to parse the input sequence: ***aabb***.

A→BA (1) **B→aB (3)**
A→ε (2) **B→b (4)**

STACK	INPUT
∇0	aabb
∇0a4	abb
∇0a4a4	bb
∇0a4a4b5	b
∇0a4a4B6	b
∇0a4B6	b
∇0B2	b
∇0B2b5	⊥
∇0B2B2	⊥
∇0B2B2A3	⊥
∇0B2A3	⊥
∇0A1	⊥
prihvati	

	a	b	⊥	A	B
0	s4	s5	r2	1	2
1			accept		
2	s4	s5	r2	3	2
3			r1		
4	s4	s5			6
5	r4	r4	r4		
6	r3	r3			

- Configuration of an LR Parser is represented by the state on top of the stack and the unread part of the input sequence.
- The parser is LR because the sequence is read from left to right, and the sequence is generated by replacing the rightmost nonterminal symbol.
- $\underline{A} \rightarrow B\underline{A} \rightarrow BB\underline{A} \rightarrow B\underline{B} \rightarrow \underline{B}b \rightarrow a\underline{B}b \rightarrow aa\underline{B}b \rightarrow aabb$

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.

$$a^i\ b^j\ c^k\ d^j\ e^i$$

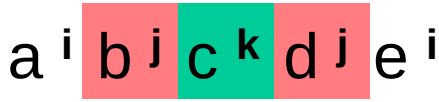
Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.

$a^i b^j c^k d^j e^i$

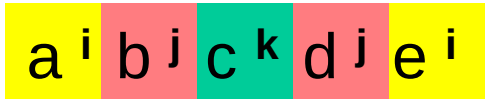
Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



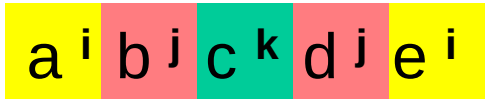
Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



Task 21

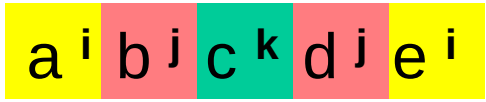
- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



- **Type of grammar = ?**

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.

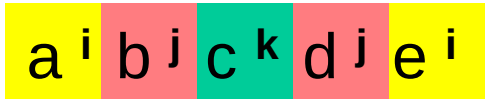


- **Type of grammar = ?**
- Context-free grammar

Task 21

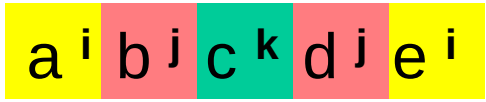
- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.

  - Symmetric sequences
  - **Type of grammar = ?**
  - Context-free grammar



Task 21

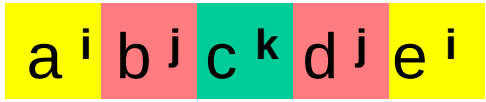
- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:

Task 21

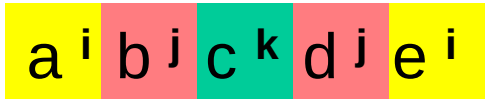
- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



  - Symmetric sequences
  - **Type of grammar = ?**
  - Context-free grammar
  - Productions simultaneously generate:
    - **a** i **e**

Task 21

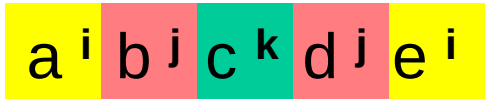
- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.

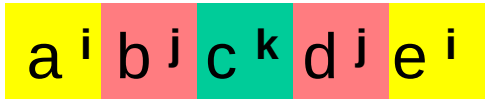


- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

S →aAe

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.

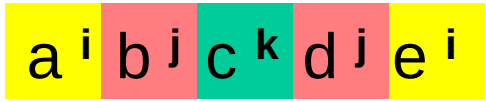


- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

S →aAe **A→aAe**

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



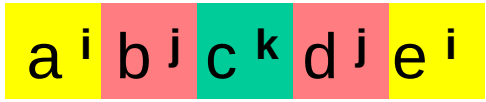
- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

S →aAe **A→aAe**

A→bBd

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



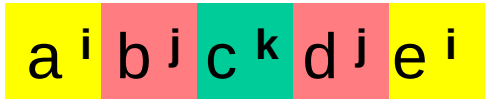
- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

S →aAe **A→aAe** **B→bBd**

A→bBd

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



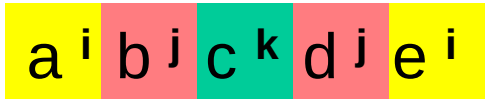
- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

S →aAe **A→aAe** **B→bBd**

A→bBd **B→cC**

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



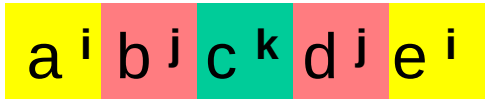
- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

S →aAe **A→aAe** **B→bBd** **C→cC**

A→bBd **B→cC**

Task 21

- Construct a grammar that generates sequences in the following format: $a^i b^j c^k d^j e^i$, i, j, k ≥ 1.



- Symmetric sequences
- **Type of grammar = ?**
- Context-free grammar
- Productions simultaneously generate:
  - **a** i **e**
  - **b** i **d**

S →aAe **A→aAe** **B→bBd** **C→cC**

A→bBd **B→cC** **C→ε**

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w \in (0+1)^*\}$.

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w \in (0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**

Task 22

- Construct a Pushdown Automata that accepts language:

  $L=\{ww^R \mid w \in (0+1)^*\}$.

  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols

Task 22

- Construct a Pushdown Automata that accepts language:

  $L=\{ww^R \mid w \in (0+1)^*\}$.

  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**

Task 22

- Construct a Pushdown Automata that accepts language:

  $L=\{ww^R \mid w \in (0+1)^*\}$.

  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack

Task 22

- Construct a Pushdown Automata that accepts language:

  $L=\{ww^R \mid w \in (0+1)^*\}$.

  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack

Task 22

- Construct a Pushdown Automata that accepts language:

  $L=\{ww^R \mid w \in (0+1)^*\}$.

  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence

Task 22

- Construct a Pushdown Automata that accepts language:

  $L=\{ww^R \mid w \in (0+1)^*\}$.

  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**

Task 22

- Construct a Pushdown Automata that accepts language:

  $L=\{ww^R \mid w \in (0+1)^*\}$.

  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

Task 22

- Construct a Pushdown Automata that accepts language:
  $L=\{ww^R \mid w \in (0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

$\delta(q_0,0,P)=(q_0,NP)$

$\delta(q_0,1,P)=(q_0,JP)$

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w \in (0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

$\delta(q_0,0,P)=(q_0,NP)$

$\delta(q_0,1,P)=(q_0,JP)$

$\delta(q_0,1,N)=(q_0,JN)$

$\delta(q_0,0,J)=(q_0,NJ)$

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w\in(0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

$\delta(q_0,0,P)=(q_0,NP)$

$\delta(q_0,1,P)=(q_0,JP)$

$\delta(q_0,1,N)=(q_0,JN)$

$\delta(q_0,0,J)=(q_0,NJ)$

$\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

$\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w\in(0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

$\delta(q_0,0,P)=(q_0,NP)$

$\delta(q_0,1,P)=(q_0,JP)$

$\delta(q_0,1,N)=(q_0,JN)$

$\delta(q_0,0,J)=(q_0,NJ)$

$\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

$\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w\in(0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

$\delta(q_0,0,P)=(q_0,NP)$

$\delta(q_0,1,P)=(q_0,JP)$

$\delta(q_0,1,N)=(q_0,JN)$

$\delta(q_0,0,J)=(q_0,NJ)$

$\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

$\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

$\delta(q_1,0,N)=(q_1,\varepsilon)$

$\delta(q_1,1,J)=(q_1,\varepsilon)$

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w\in(0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

$\delta(q_0,0,P)=(q_0,NP)$

$\delta(q_0,1,P)=(q_0,JP)$

$\delta(q_0,1,N)=(q_0,JN)$

$\delta(q_0,0,J)=(q_0,NJ)$

$\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

$\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

$\delta(q_1,0,N)=(q_1,\varepsilon)$

$\delta(q_1,1,J)=(q_1,\varepsilon)$

$\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

Task 22

- Construct a Pushdown Automata that accepts language: $L=\{ww^R \mid w\in(0+1)^*\}$.
  - **$M=(\{q_0,q_1\},\{0,1\},\{P,N,J\},\delta,q_0,P,\varnothing)$**
  - **q_0** remembers input symbols
  - **N** and **J** code **0** and **1**
  - **q_1** removes symbols from the stack
  - Pushdown Automata accepts by an empty stack
  - Pushdown Automata accepts an empty sequence
  - **Nondeterministic!**
    - Nondeterministic transition occurs when the same input symbol is repeated

$\delta(q_0,0,P)=(q_0,NP)$

$\delta(q_0,1,P)=(q_0,JP)$

$\delta(q_0,1,N)=(q_0,JN)$

$\delta(q_0,0,J)=(q_0,NJ)$

$\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

$\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

$\delta(q_1,0,N)=(q_1,\varepsilon)$

$\delta(q_1,1,J)=(q_1,\varepsilon)$

$\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

$\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued

(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued

$(q_0, 001100, P)$

(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued

(q_0, 001100, P)

(1) (10)

(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

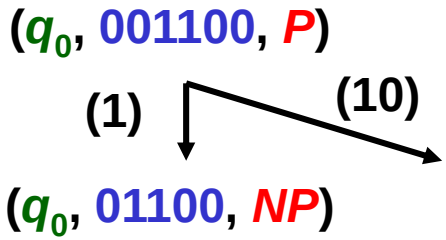
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

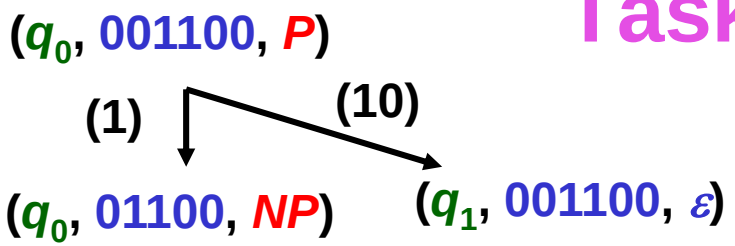
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

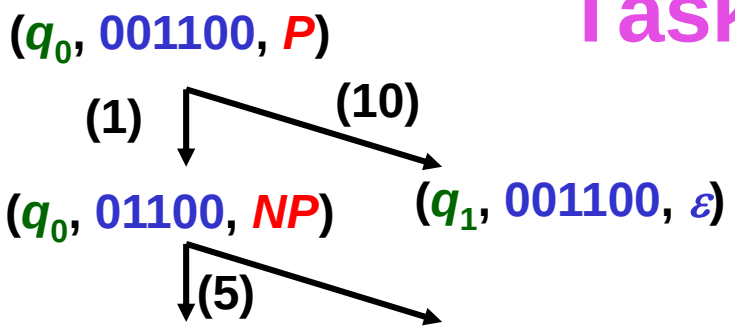
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

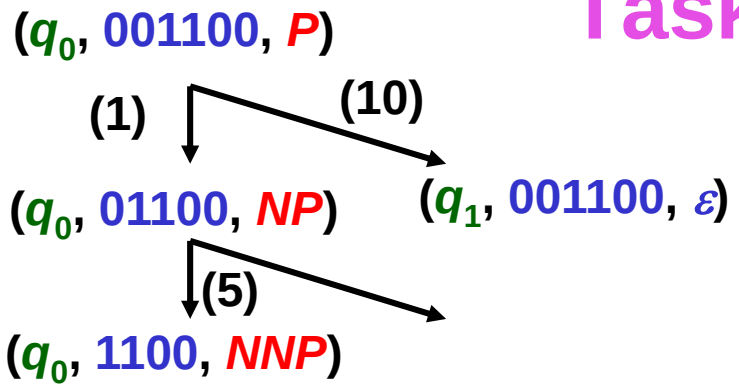
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

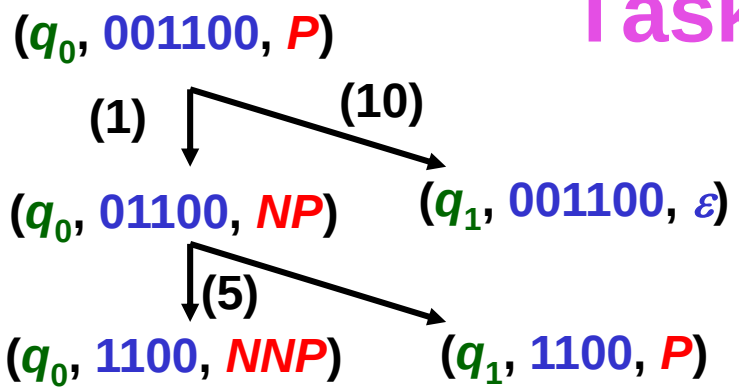
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

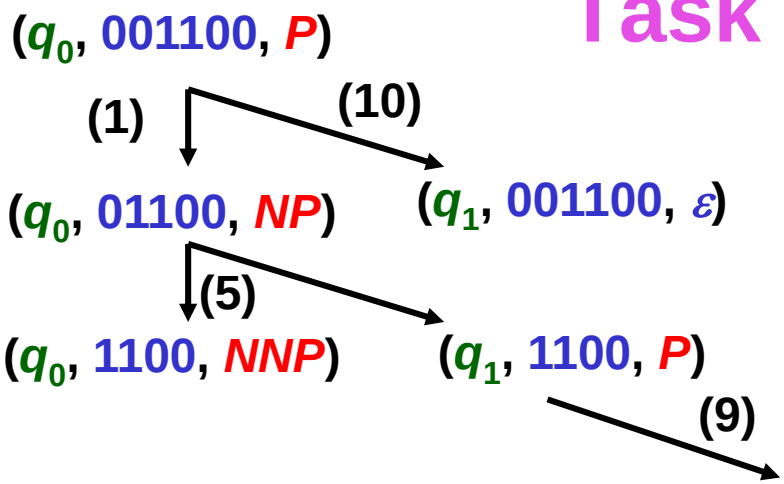
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

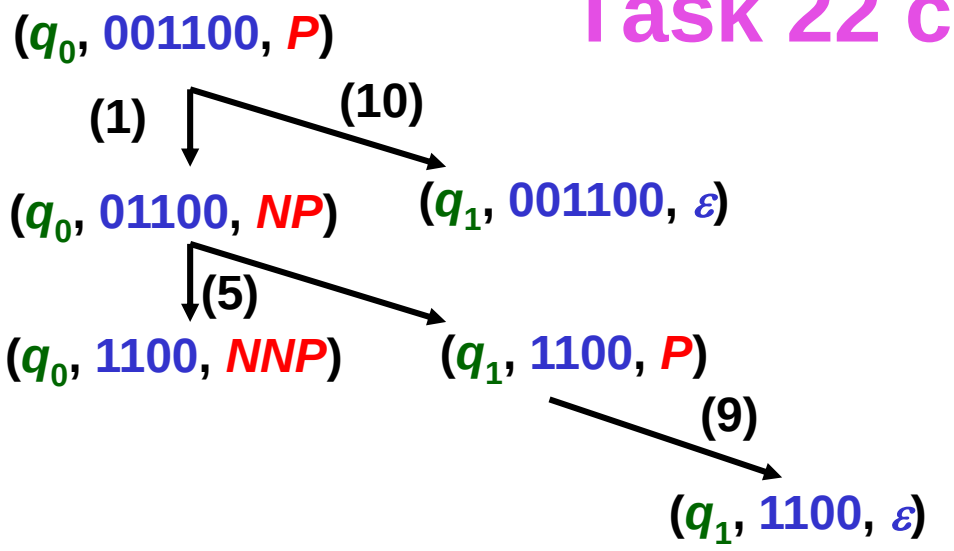
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

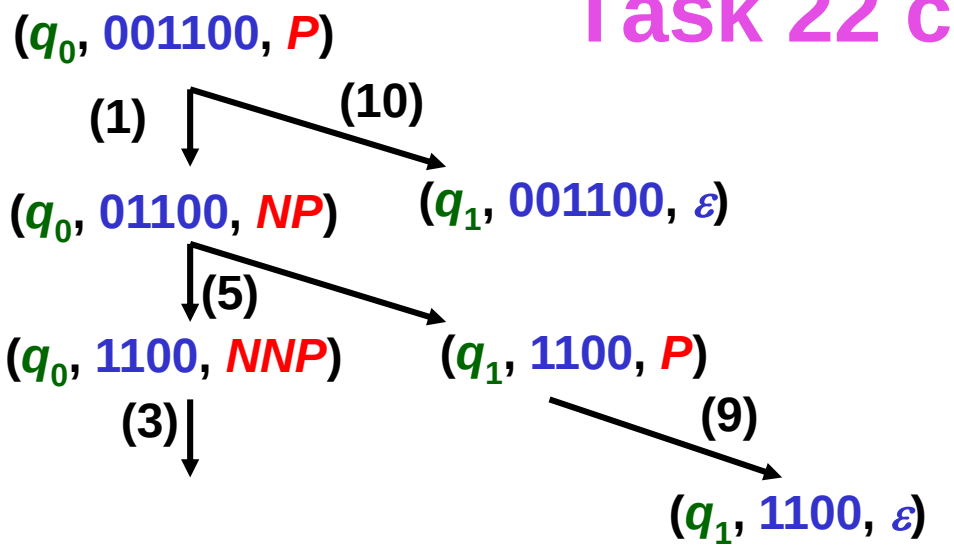
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

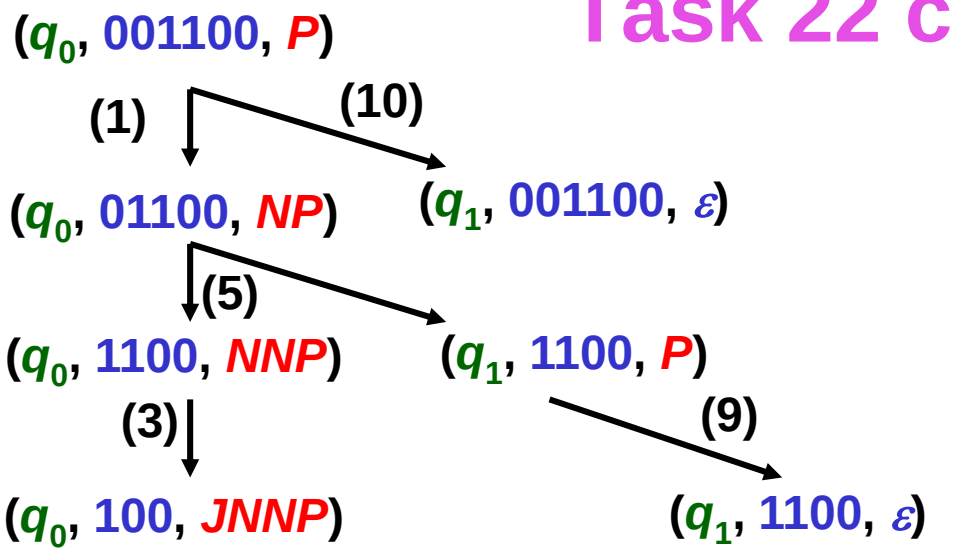
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

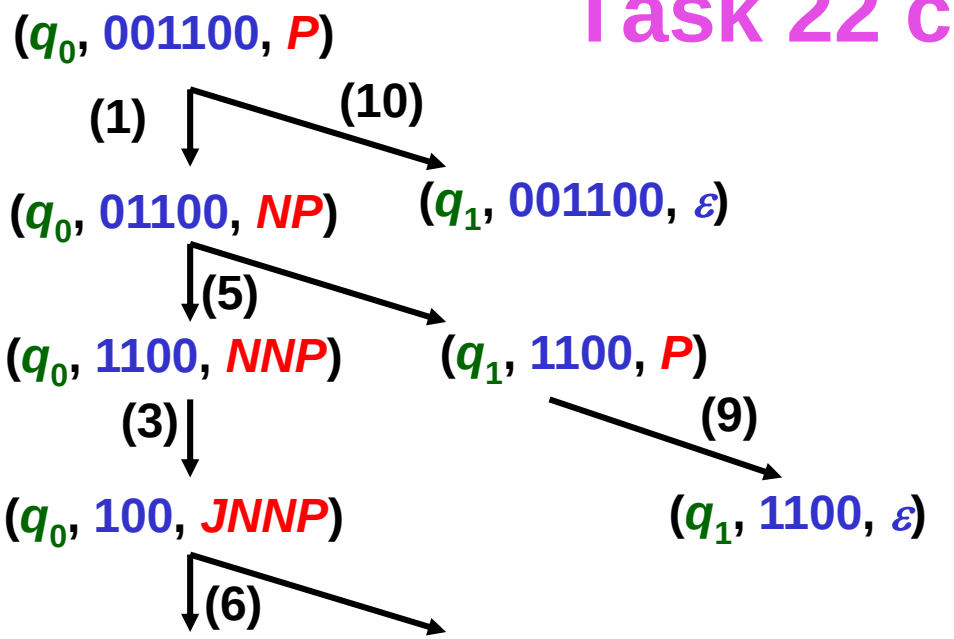
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

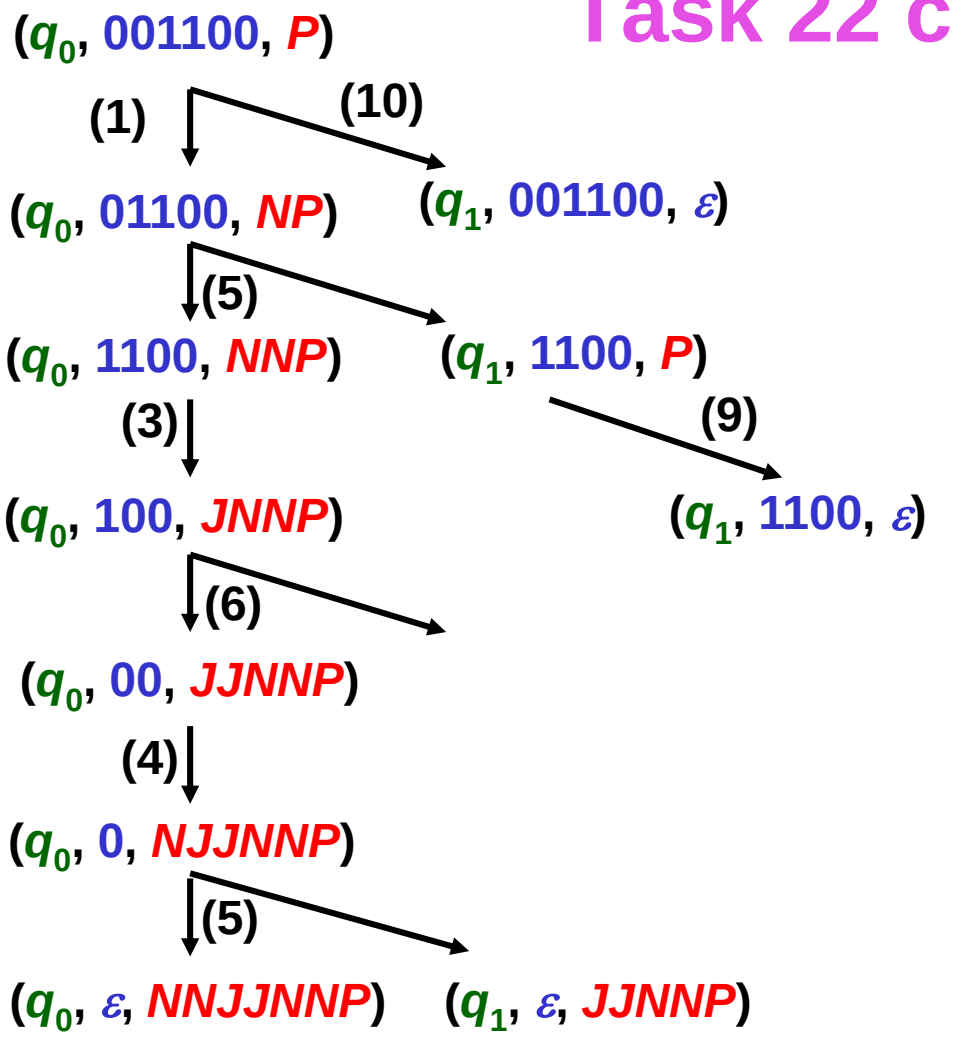
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

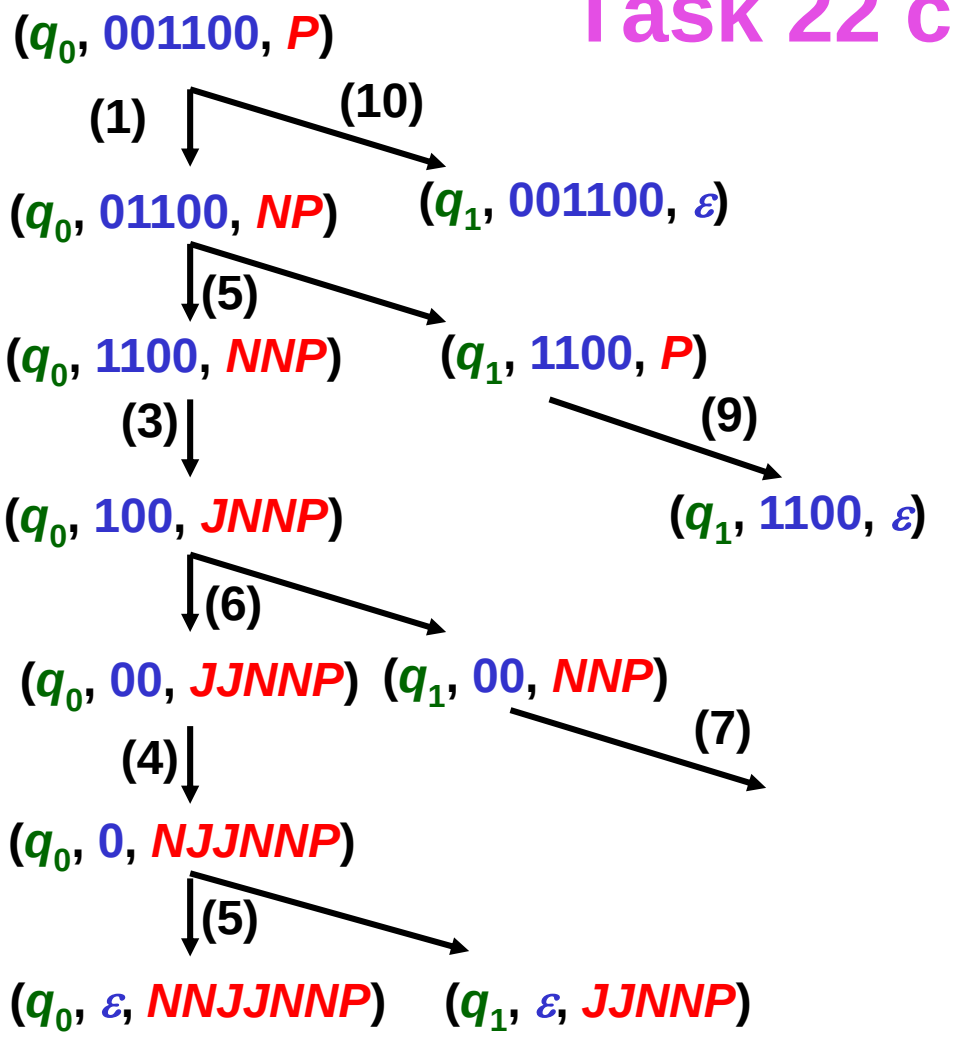
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

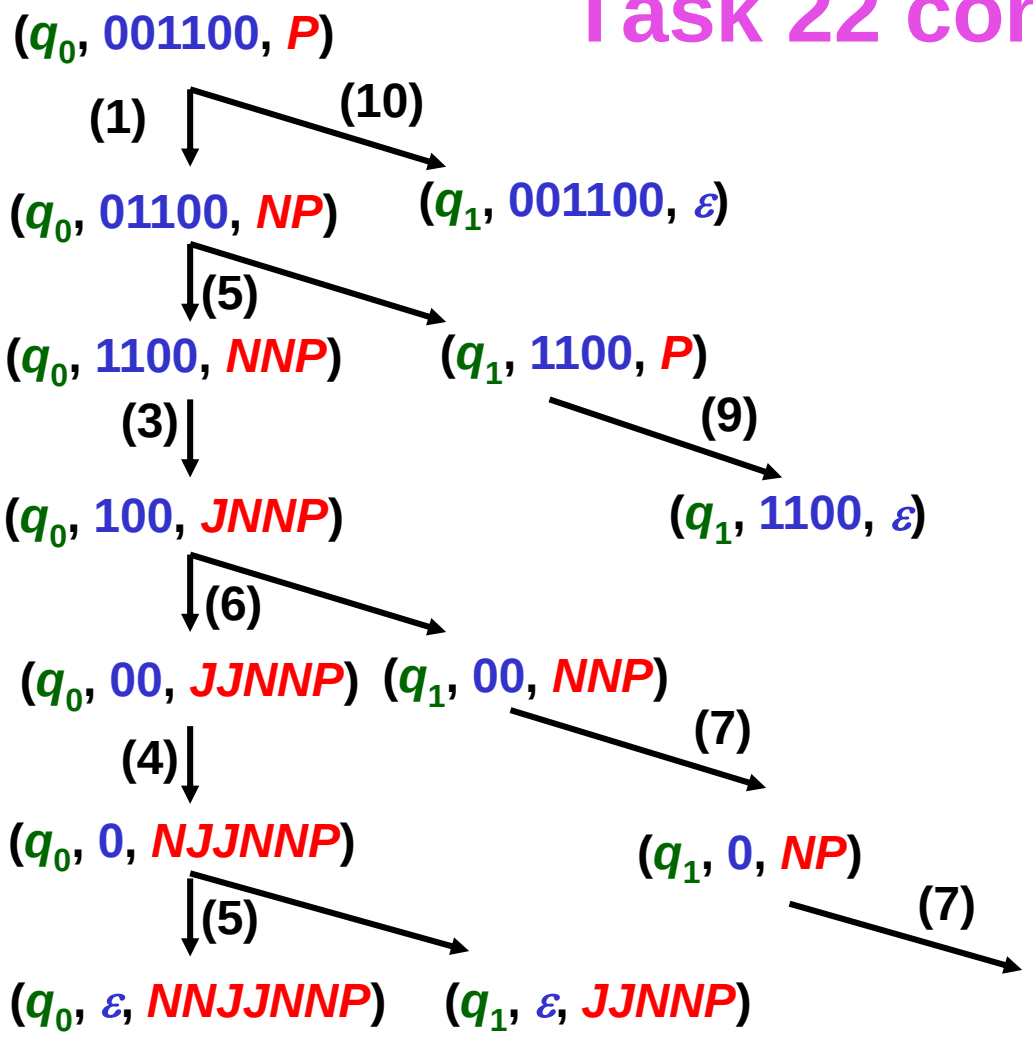
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

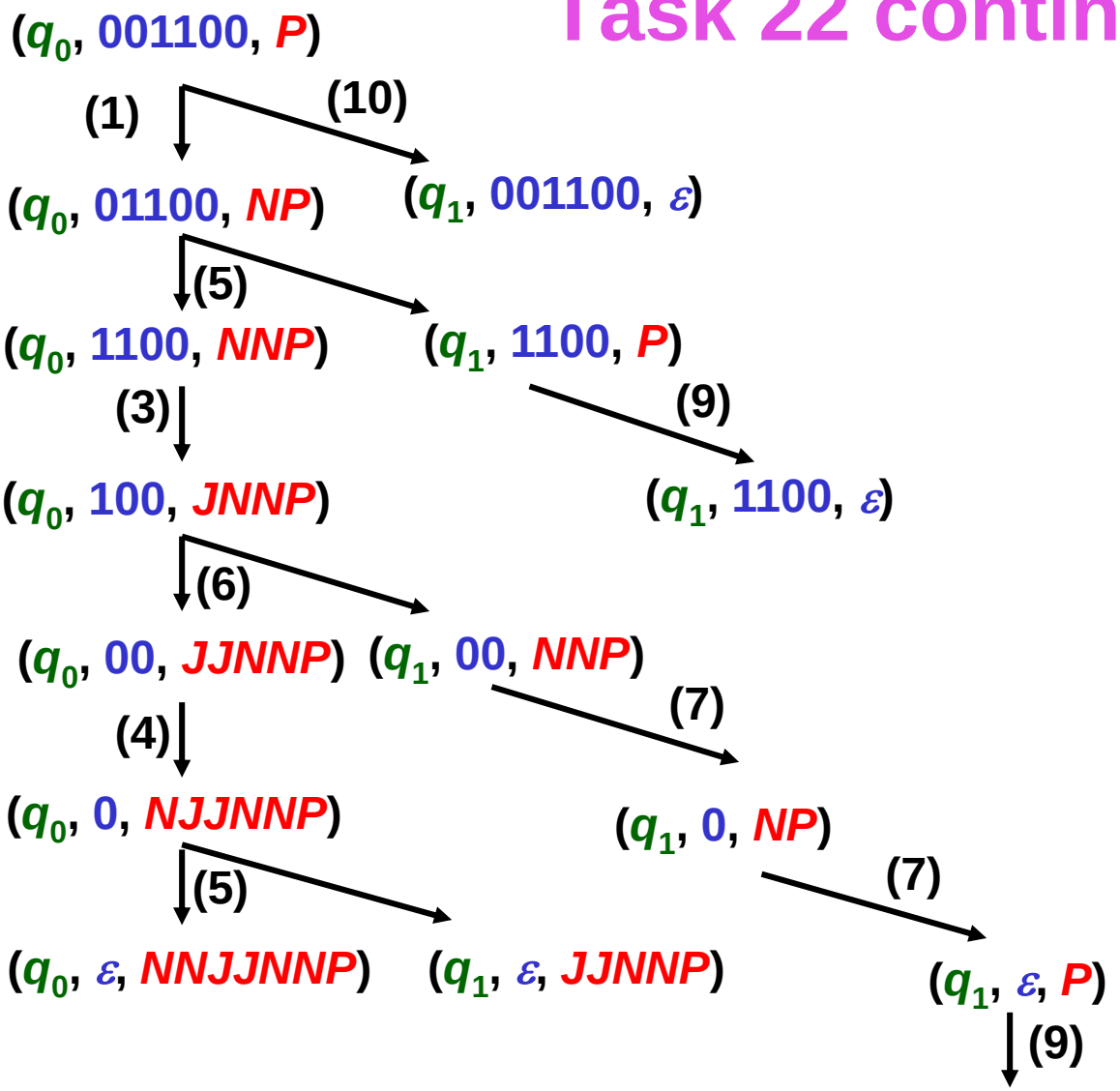
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

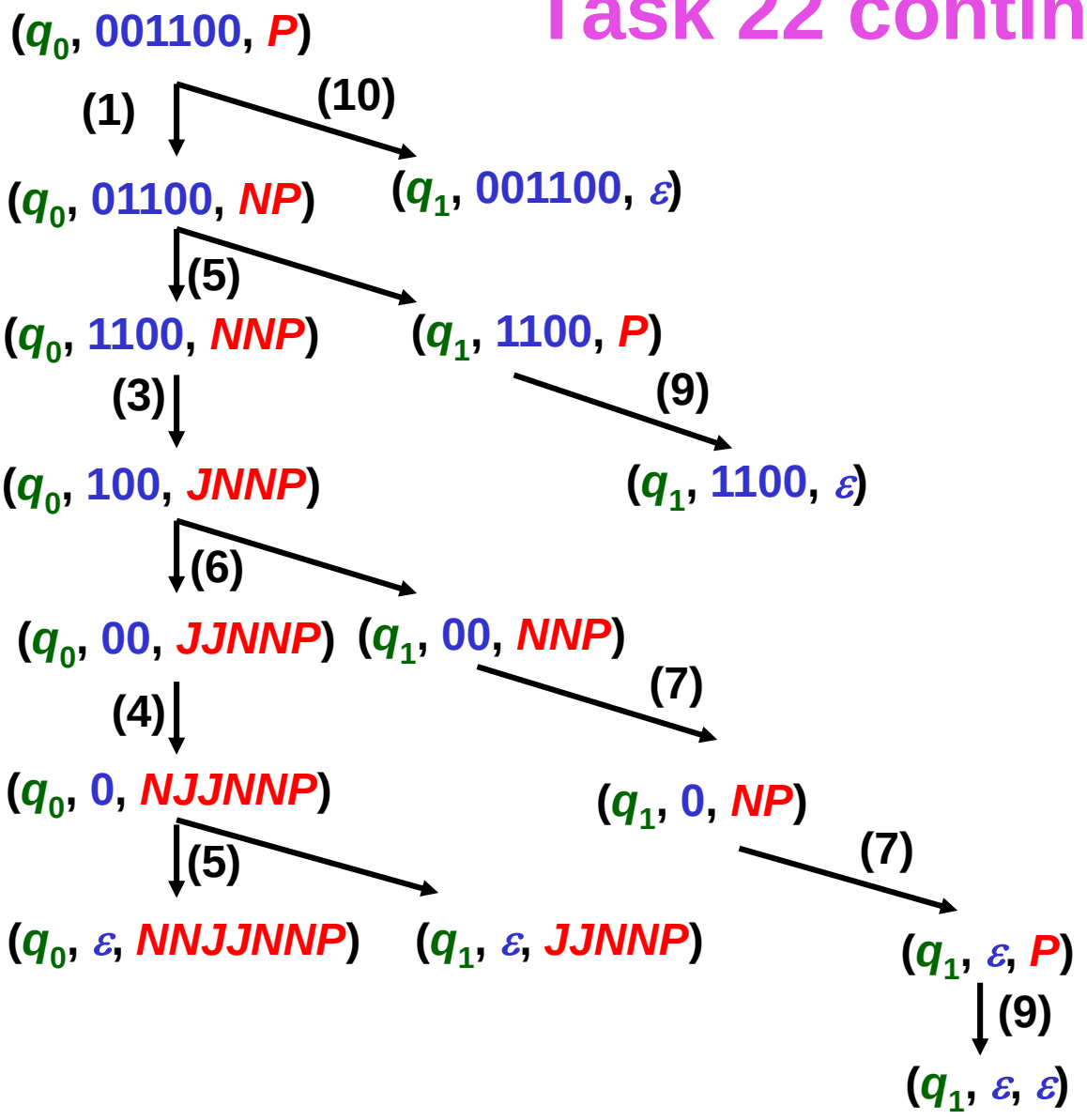
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

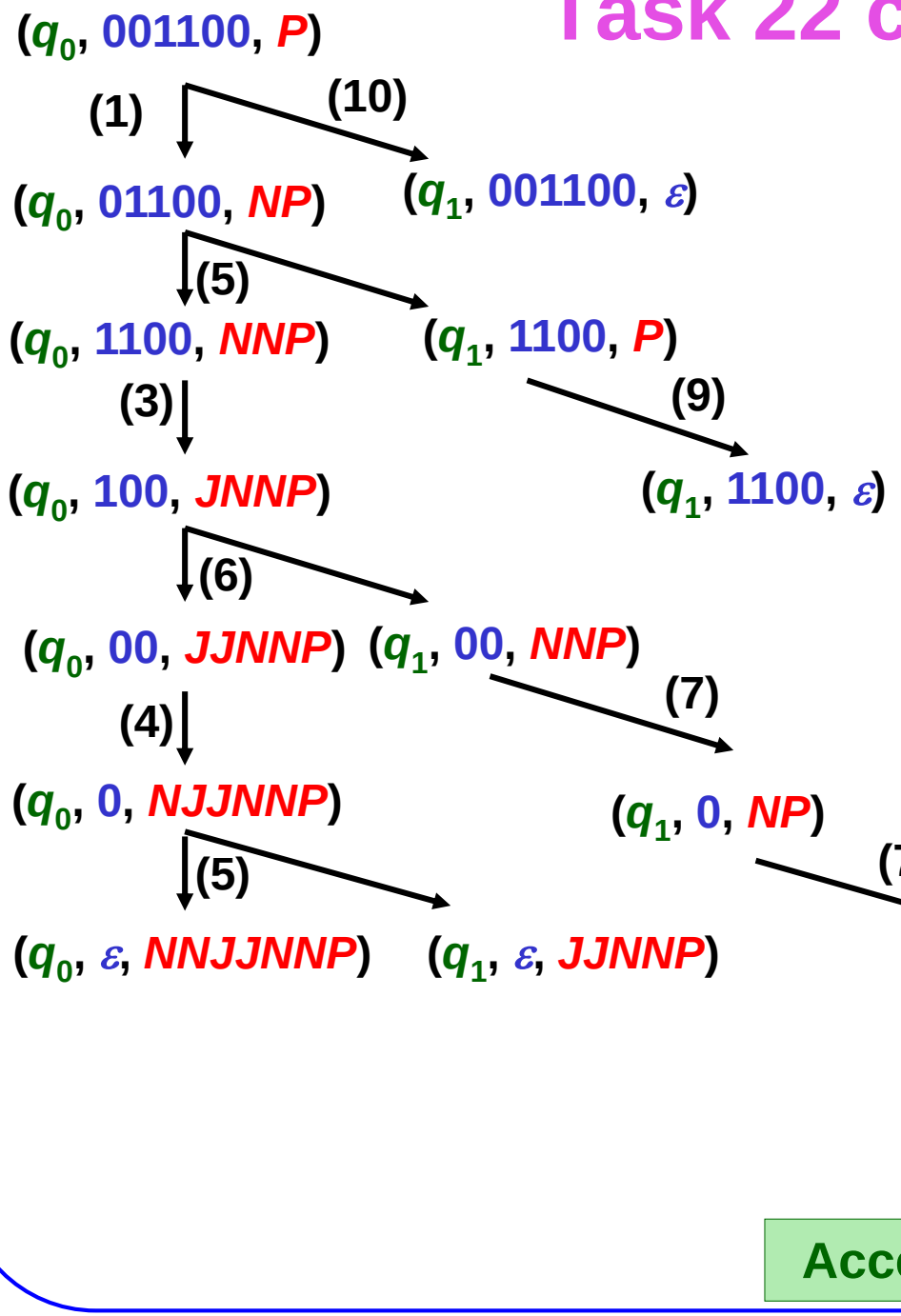
(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 22 continued



(1) $\delta(q_0,0,P)=(q_0,NP)$

(2) $\delta(q_0,1,P)=(q_0,JP)$

(3) $\delta(q_0,1,N)=(q_0,JN)$

(4) $\delta(q_0,0,J)=(q_0,NJ)$

(5) $\delta(q_0,0,N)=\{(q_0,NN),(q_1,\varepsilon)\}$

(6) $\delta(q_0,1,J)=\{(q_0,JJ),(q_1,\varepsilon)\}$

(7) $\delta(q_1,0,N)=(q_1,\varepsilon)$

(8) $\delta(q_1,1,J)=(q_1,\varepsilon)$

(9) $\delta(q_1,\varepsilon,P)=(q_1,\varepsilon)$

(10) $\delta(q_0,\varepsilon,P)=(q_0,\varepsilon)$

Task 23

- From a Pushdown Automata M1 that accepts by final (accepting) state, construct a Pushdown Automata M2 that accepts by empty stack.

$$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$$

$\delta(q_0,0,K)=(q_0,NK)$ $\quad$ $\delta(q_0,0,N)=(q_0,NN)$ $\quad$ $\delta(q_0,0,J)=(q_0,\varepsilon)$

$\delta(q_0,1,K)=(q_0,JK)$ $\quad$ $\delta(q_0,1,N)=(q_0,\varepsilon)$ $\quad$ $\delta(q_0,1,J)=(q_0,JJ)$

$\delta(q_0,2,K)=(q_0,K)$ $\quad$ $\delta(q_0,2,N)=(q_0,N)$ $\quad$ $\delta(q_0,2,J)=(q_0,J)$

$\delta(q_0,\varepsilon,K)=(q_1,K)$

Task 23

- From a Pushdown Automata M1 that accepts by final (accepting) state, construct a Pushdown Automata M2 that accepts by empty stack.

$$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$$

$\delta(q_0,0,K)=(q_0,NK)$ $\quad$ $\delta(q_0,0,N)=(q_0,NN)$ $\quad$ $\delta(q_0,0,J)=(q_0,\varepsilon)$

$\delta(q_0,1,K)=(q_0,JK)$ $\quad$ $\delta(q_0,1,N)=(q_0,\varepsilon)$ $\quad$ $\delta(q_0,1,J)=(q_0,JJ)$

$\delta(q_0,2,K)=(q_0,K)$ $\quad$ $\delta(q_0,2,N)=(q_0,N)$ $\quad$ $\delta(q_0,2,J)=(q_0,J)$

$\delta(q_0,\varepsilon,K)=(q_1,K)$

Constructing PA M_2 that accepts by empty stack:

Task 23

- From a Pushdown Automata M1 that accepts by final (accepting) state, construct a Pushdown Automata M2 that accepts by empty stack.

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta(q_0,0,K)=(q_0,NK)$ $\quad \delta(q_0,0,N)=(q_0,NN)$ $\quad \delta(q_0,0,J)=(q_0,\varepsilon)$

$\delta(q_0,1,K)=(q_0,JK)$ $\quad \delta(q_0,1,N)=(q_0,\varepsilon)$ $\quad \delta(q_0,1,J)=(q_0,JJ)$

$\delta(q_0,2,K)=(q_0,K)$ $\quad \delta(q_0,2,N)=(q_0,N)$ $\quad \delta(q_0,2,J)=(q_0,J)$

$\delta(q_0,\varepsilon,K)=(q_1,K)$

Constructing PA M_2 that accepts by empty stack:

a) PA M_2 simulates PA M_1

Task 23

- From a Pushdown Automata M1 that accepts by final (accepting) state, construct a Pushdown Automata M2 that accepts by empty stack.

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta(q_0,0,K)=(q_0,NK)$ $\quad$ $\delta(q_0,0,N)=(q_0,NN)$ $\quad$ $\delta(q_0,0,J)=(q_0,\varepsilon)$

$\delta(q_0,1,K)=(q_0,JK)$ $\quad$ $\delta(q_0,1,N)=(q_0,\varepsilon)$ $\quad$ $\delta(q_0,1,J)=(q_0,JJ)$

$\delta(q_0,2,K)=(q_0,K)$ $\quad$ $\delta(q_0,2,N)=(q_0,N)$ $\quad$ $\delta(q_0,2,J)=(q_0,J)$

$\delta(q_0,\varepsilon,K)=(q_1,K)$

Constructing PA M_2 that accepts by empty stack:

a) PA M_2 simulates PA M_1

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Task 23

- From a Pushdown Automata M1 that accepts by final (accepting) state, construct a Pushdown Automata M2 that accepts by empty stack.

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta(q_0,0,K)=(q_0,NK)$ $\quad \delta(q_0,0,N)=(q_0,NN)$ $\quad \delta(q_0,0,J)=(q_0,\varepsilon)$

$\delta(q_0,1,K)=(q_0,JK)$ $\quad \delta(q_0,1,N)=(q_0,\varepsilon)$ $\quad \delta(q_0,1,J)=(q_0,JJ)$

$\delta(q_0,2,K)=(q_0,K)$ $\quad \delta(q_0,2,N)=(q_0,N)$ $\quad \delta(q_0,2,J)=(q_0,J)$

$\delta(q_0,\varepsilon,K)=(q_1,K)$

Constructing PA M_2 that accepts by empty stack:

a) PA M_2 simulates PA M_1

b) If PA M_1 is in accepting state, PA M_2 empties the stack

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Task 23

Constructing PA M_2 that accepts by empty stack:

a) PA M_2 simulates PA M_1

Task 23

Constructing PA M_2 that accepts by empty stack:

a) PA M_2 simulates PA M_1

All PA M_1 transitions are added to set of transitions PA M_2

Task 23

Constructing PA M_2 that accepts by empty stack:

a) PA M_2 simulates PA M_1

All PA M_1 transitions are added to set of transitions PA M_2

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta(q_0,0,K)=(q_0,NK)$ $\quad$ $\delta(q_0,0,N)=(q_0,NN)$ $\quad$ $\delta(q_0,0,J)=(q_0,\varepsilon)$

$\delta(q_0,1,K)=(q_0,JK)$ $\quad$ $\delta(q_0,1,N)=(q_0,\varepsilon)$ $\quad$ $\delta(q_0,1,J)=(q_0,JJ)$

$\delta(q_0,2,K)=(q_0,K)$ $\quad$ $\delta(q_0,2,N)=(q_0,N)$ $\quad$ $\delta(q_0,2,J)=(q_0,J)$

$\delta(q_0,\varepsilon,K)=(q_1,K)$

Task 23

Constructing PA M_2 that accepts by empty stack:

a) PA M_2 simulates PA M_1

All PA M_1 transitions are added to set of transitions PA M_2

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta(q_0,0,K)=(q_0,NK)$	$\delta(q_0,0,N)=(q_0,NN)$	$\delta(q_0,0,J)=(q_0,\varepsilon)$
$\delta(q_0,1,K)=(q_0,JK)$	$\delta(q_0,1,N)=(q_0,\varepsilon)$	$\delta(q_0,1,J)=(q_0,JJ)$
$\delta(q_0,2,K)=(q_0,K)$	$\delta(q_0,2,N)=(q_0,N)$	$\delta(q_0,2,J)=(q_0,J)$
$\delta(q_0,\varepsilon,K)=(q_1,K)$		

$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_0,0,J)=(q_0,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,N)=(q_0,\varepsilon)$	$\delta'(q_0,1,J)=(q_0,JJ)$
$\delta'(q_0,2,K)=(q_0,K)$	$\delta'(q_0,2,N)=(q_0,N)$	$\delta'(q_0,2,J)=(q_0,J)$
$\delta'(q_0,\varepsilon,K)=(q_1,K)$		

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$

$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,K)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 that accepts by empty stack:

b) If PA M_1 is in accepting state, PA M_2 empties the stack

Set of states PA M_2 is extended by state q_E in which PA M_2 empties the stack without reading input symbols

PA M_2 enters state q_E without reading input symbols only if PA M_1 is in accepting state

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,K)=(q_E,\varepsilon)$

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Symbol Z is the start stack symbol for PA M_2

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Symbol Z is the start stack symbol for PA M_2

Set of states PA M_2 is extended by state q_P which enables the transition of

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Symbol Z is the start stack symbol for PA M_2

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Symbol Z is the start stack symbol for PA M_2

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_E\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \varnothing\ )$

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Symbol Z is the start stack symbol for PA M_2

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_E\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \varnothing\ )$

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Symbol Z is the start stack symbol for PA M_2

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_E\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \varnothing\ )$

Task 23

Constructing PA M_2 koji prihvaća praznim stogom:

c) If PA M_1 empties the stack, but is not in an accepting state, PA M_2 must not empty the stack

Set of stack symbols PA M_2 is extended by symbol Z which PA M_1 can not remove from the stack

Symbol Z is the start stack symbol for PA M_2

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \{q_1\}\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_E\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$

Task 23

PA M_2 that accepts with an empty stack:

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P, q_0, q_1, q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P,q_0,q_1,q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P, q_0, q_1, q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_0,0,J)=(q_0,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,N)=(q_0,\varepsilon)$	$\delta'(q_0,1,J)=(q_0,JJ)$
$\delta'(q_0,2,K)=(q_0,K)$	$\delta'(q_0,2,N)=(q_0,N)$	$\delta'(q_0,2,J)=(q_0,J)$
$\delta'(q_0,\varepsilon,K)=(q_1,K)$		

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P,q_0,q_1,q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_0,0,J)=(q_0,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,N)=(q_0,\varepsilon)$	$\delta'(q_0,1,J)=(q_0,JJ)$
$\delta'(q_0,2,K)=(q_0,K)$	$\delta'(q_0,2,N)=(q_0,N)$	$\delta'(q_0,2,J)=(q_0,J)$
$\delta'(q_0,\varepsilon,K)=(q_1,K)$		
$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$	
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$	
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,K)=(q_E,\varepsilon)$	

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P,q_0,q_1,q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_0,0,J)=(q_0,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,N)=(q_0,\varepsilon)$	$\delta'(q_0,1,J)=(q_0,JJ)$
$\delta'(q_0,2,K)=(q_0,K)$	$\delta'(q_0,2,N)=(q_0,N)$	$\delta'(q_0,2,J)=(q_0,J)$
$\delta'(q_0,\varepsilon,K)=(q_1,K)$		
$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$	
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$	
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,K)=(q_E,\varepsilon)$	

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P,q_0,q_1,q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_0,0,J)=(q_0,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,N)=(q_0,\varepsilon)$	$\delta'(q_0,1,J)=(q_0,JJ)$
$\delta'(q_0,2,K)=(q_0,K)$	$\delta'(q_0,2,N)=(q_0,N)$	$\delta'(q_0,2,J)=(q_0,J)$
$\delta'(q_0,\varepsilon,K)=(q_1,K)$		

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,K)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,Z)=(q_E,\varepsilon)$	

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P,q_0,q_1,q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_0,0,J)=(q_0,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,N)=(q_0,\varepsilon)$	$\delta'(q_0,1,J)=(q_0,JJ)$
$\delta'(q_0,2,K)=(q_0,K)$	$\delta'(q_0,2,N)=(q_0,N)$	$\delta'(q_0,2,J)=(q_0,J)$
$\delta'(q_0,\varepsilon,K)=(q_1,K)$		

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,K)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,Z)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,Z)=(q_E,\varepsilon)$

Task 23

PA M_2 that accepts with an empty stack:

$M_2 = (\ \{q_P,q_0,q_1,q_E\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \varnothing\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_0,0,J)=(q_0,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,N)=(q_0,\varepsilon)$	$\delta'(q_0,1,J)=(q_0,JJ)$
$\delta'(q_0,2,K)=(q_0,K)$	$\delta'(q_0,2,N)=(q_0,N)$	$\delta'(q_0,2,J)=(q_0,J)$
$\delta'(q_0,\varepsilon,K)=(q_1,K)$		

$\delta'(q_1,\varepsilon,N)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,N)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,J)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,J)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,K)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,K)=(q_E,\varepsilon)$
$\delta'(q_1,\varepsilon,Z)=(q_E,\varepsilon)$	$\delta'(q_E,\varepsilon,Z)=(q_E,\varepsilon)$

Task 24

- From Pushdown Automata M_1 that accepts by empty stack, construct a Pushdown Automata M_2 that accepts by . From Pushdown Automata that accepts by final (accepting) state.

$$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$$

$\delta(q_0,0,K)=(q_0,NK)$	$\delta(q_0,0,J)=(q_0,NJ)$	$\delta(q_0,0,N)=(q_0,NN)$	$\delta(q_1,0,N)=(q_1,\varepsilon)$
$\delta(q_0,1,K)=(q_0,JK)$	$\delta(q_0,1,J)=(q_0,JJ)$	$\delta(q_0,1,N)=(q_0,JN)$	$\delta(q_1,1,J)=(q_1,\varepsilon)$
$\delta(q_0,2,K)=(q_1,\varepsilon)$	$\delta(q_0,2,J)=(q_1,J)$	$\delta(q_0,2,N)=(q_1,N)$	$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

Task 24

- From Pushdown Automata M_1 that accepts by empty stack, construct a Pushdown Automata M_2 that accepts by . From Pushdown Automata that accepts by final (accepting) state.

$$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$$

$\delta(q_0,0,K)=(q_0,NK)$	$\delta(q_0,0,J)=(q_0,NJ)$	$\delta(q_0,0,N)=(q_0,NN)$	$\delta(q_1,0,N)=(q_1,\varepsilon)$
$\delta(q_0,1,K)=(q_0,JK)$	$\delta(q_0,1,J)=(q_0,JJ)$	$\delta(q_0,1,N)=(q_0,JN)$	$\delta(q_1,1,J)=(q_1,\varepsilon)$
$\delta(q_0,2,K)=(q_1,\varepsilon)$	$\delta(q_0,2,J)=(q_1,J)$	$\delta(q_0,2,N)=(q_1,N)$	$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

Constructing PA M_2 that accepts with final (accepting) state:

Task 24

- From Pushdown Automata M_1 that accepts by empty stack, construct a Pushdown Automata M_2 that accepts by . From Pushdown Automata that accepts by final (accepting) state.

$$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$$

$\delta(q_0,0,K)=(q_0,NK)$	$\delta(q_0,0,J)=(q_0,NJ)$	$\delta(q_0,0,N)=(q_0,NN)$	$\delta(q_1,0,N)=(q_1,\varepsilon)$
$\delta(q_0,1,K)=(q_0,JK)$	$\delta(q_0,1,J)=(q_0,JJ)$	$\delta(q_0,1,N)=(q_0,JN)$	$\delta(q_1,1,J)=(q_1,\varepsilon)$
$\delta(q_0,2,K)=(q_1,\varepsilon)$	$\delta(q_0,2,J)=(q_1,J)$	$\delta(q_0,2,N)=(q_1,N)$	$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

Constructing PA M_2 that accepts with final (accepting) state:

a) PA M_2 simulates PA M_1

Task 24

- From Pushdown Automata M_1 that accepts by empty stack, construct a Pushdown Automata M_2 that accepts by . From Pushdown Automata that accepts by final (accepting) state.

$$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$$

$\delta(q_0,0,K)=(q_0,NK)$	$\delta(q_0,0,J)=(q_0,NJ)$	$\delta(q_0,0,N)=(q_0,NN)$	$\delta(q_1,0,N)=(q_1,\varepsilon)$
$\delta(q_0,1,K)=(q_0,JK)$	$\delta(q_0,1,J)=(q_0,JJ)$	$\delta(q_0,1,N)=(q_0,JN)$	$\delta(q_1,1,J)=(q_1,\varepsilon)$
$\delta(q_0,2,K)=(q_1,\varepsilon)$	$\delta(q_0,2,J)=(q_1,J)$	$\delta(q_0,2,N)=(q_1,N)$	$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

Constructing PA M_2 that accepts with final (accepting) state:

a) PA M_2 simulates PA M_1

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

a) PA M_2 simulates PA M_1

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

a) PA M_2 simulates PA M_1

All PA M_1 transitions are added to the set of transitions PA M_2

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

a) PA M_2 simulates PA M_1

All PA M_1 transitions are added to the set of transitions PA M_2

$$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$$

$\delta(q_0,0,K)=(q_0,NK)$	$\delta(q_0,0,J)=(q_0,NJ)$	$\delta(q_0,0,N)=(q_0,NN)$	$\delta(q_1,0,N)=(q_1,\varepsilon)$
$\delta(q_0,1,K)=(q_0,JK)$	$\delta(q_0,1,J)=(q_0,JJ)$	$\delta(q_0,1,N)=(q_0,JN)$	$\delta(q_1,1,J)=(q_1,\varepsilon)$
$\delta(q_0,2,K)=(q_1,\varepsilon)$	$\delta(q_0,2,J)=(q_1,J)$	$\delta(q_0,2,N)=(q_1,N)$	$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

a) PA M_2 simulates PA M_1

All PA M_1 transitions are added to the set of transitions PA M_2

$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$

$\delta(q_0,0,K)=(q_0,NK)$	$\delta(q_0,0,J)=(q_0,NJ)$	$\delta(q_0,0,N)=(q_0,NN)$	$\delta(q_1,0,N)=(q_1,\varepsilon)$
$\delta(q_0,1,K)=(q_0,JK)$	$\delta(q_0,1,J)=(q_0,JJ)$	$\delta(q_0,1,N)=(q_0,JN)$	$\delta(q_1,1,J)=(q_1,\varepsilon)$
$\delta(q_0,2,K)=(q_1,\varepsilon)$	$\delta(q_0,2,J)=(q_1,J)$	$\delta(q_0,2,N)=(q_1,N)$	$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,J)=(q_0,NJ)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_1,0,N)=(q_1,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,J)=(q_0,JJ)$	$\delta'(q_0,1,N)=(q_0,JN)$	$\delta'(q_1,1,J)=(q_1,\varepsilon)$
$\delta'(q_0,2,K)=(q_1,\varepsilon)$	$\delta'(q_0,2,J)=(q_1,J)$	$\delta'(q_0,2,N)=(q_1,N)$	$\delta'(q_1,\varepsilon,K)=(q_1,\varepsilon)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1 has emptied the stack

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1 has emptied the stack

$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1 has emptied the stack

$M_1 = (\{q_0, q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1 has emptied the stack

$M_1 = (\{q_0, q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$

$\delta'(q_0, \varepsilon, Z) = (q_F, Z)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1 has emptied the stack

$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$

$\delta'(q_0,\varepsilon,Z)=(q_F,Z)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1 has emptied the stack

$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$

$\delta'(q_0,\varepsilon,Z)=(q_F,Z)$

$\delta'(q_1,\varepsilon,Z)=(q_F,Z)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters final (accepting) state

Set of states PA M_2 is extended by final (accepting) state q_F in which PA M_2 enters without reading input symbols only if PA M_1 empties the stack

Set of stack symbols PA M_2 is extended by symbol Z (the start stack symbol)

If symbol Z is on top of stack PA M_2 , that means that PA M_1 has emptied the stack

$M_1 = (\{q_0,q_1\}, \{0,1,2\}, \{N,J,K\}, q_0, \delta, K, \varnothing)$

$\delta'(q_0,\varepsilon,Z)=(q_F,Z)$

$\delta'(q_1,\varepsilon,Z)=(q_F,Z)$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters the final (accepting) state

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters the final (accepting) state

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters the final (accepting) state

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \varnothing\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_F\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \{q_F\}\ )$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters the final (accepting) state

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \varnothing\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_F\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \{q_F\}\ )$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters the final (accepting) state

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \varnothing\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_F\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \{q_F\}\ )$

Task 24

Constructing PA M_2 that accepts by final (accepting) state:

b) If PA M_1 empties the stack, PA M_2 enters the final (accepting) state

Set of states PA M_2 is extended by state q_P which enables the transition of PA M_2 into the initial configuration of PA M_1

$M_1 = (\ \{q_0,q_1\},\ \{0,1,2\},\ \{N,J,K\},\ q_0,\ \delta,\ K,\ \varnothing\ )$

$M_2 = (\ \{q_P,q_0,q_1,q_F\},\ \{0,1,2\},\ \{N,J,K,Z\},\ q_P,\ \delta',\ Z,\ \{q_F\}\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$

Task 24

PA M_2 that accepts by final (accepting) state:

Task 24

PA M_2 that accepts by final (accepting) state:

$M_2 = (\ \{q_P, q_0, q_1, q_F\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \{q_F\}\ )$

Task 24

PA M_2 that accepts by final (accepting) state:

$M_2 = (\ \{q_P, q_0, q_1, q_F\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \{q_F\}\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$		

Task 24

PA M_2 that accepts by final (accepting) state:

$M_2 = (\ \{q_P, q_0, q_1, q_F\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \{q_F\}\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$			
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,J)=(q_0,NJ)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_1,0,N)=(q_1,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,J)=(q_0,JJ)$	$\delta'(q_0,1,N)=(q_0,JN)$	$\delta'(q_1,1,J)=(q_1,\varepsilon)$
$\delta'(q_0,2,K)=(q_1,\varepsilon)$	$\delta'(q_0,2,J)=(q_1,J)$	$\delta'(q_0,2,N)=(q_1,N)$	$\delta'(q_1,\varepsilon,K)=(q_1,\varepsilon)$

Task 24

PA M_2 that accepts by final (accepting) state:

$M_2 = (\ \{q_P,q_0,q_1,q_F\}, \{0,1,2\}, \{N,J,K,Z\}, q_P, \delta', Z, \{q_F\}\ )$

$\delta'(q_P,\varepsilon,Z)=(q_0,KZ)$			
$\delta'(q_0,0,K)=(q_0,NK)$	$\delta'(q_0,0,J)=(q_0,NJ)$	$\delta'(q_0,0,N)=(q_0,NN)$	$\delta'(q_1,0,N)=(q_1,\varepsilon)$
$\delta'(q_0,1,K)=(q_0,JK)$	$\delta'(q_0,1,J)=(q_0,JJ)$	$\delta'(q_0,1,N)=(q_0,JN)$	$\delta'(q_1,1,J)=(q_1,\varepsilon)$
$\delta'(q_0,2,K)=(q_1,\varepsilon)$	$\delta'(q_0,2,J)=(q_1,J)$	$\delta'(q_0,2,N)=(q_1,N)$	$\delta'(q_1,\varepsilon,K)=(q_1,\varepsilon)$
$\delta'(q_0,\varepsilon,Z)=(q_F,Z)$			
$\delta'(q_1,\varepsilon,Z)=(q_F,Z)$			

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

$S \rightarrow xABy$	$B \rightarrow wC$	$C \rightarrow vB$
$A \rightarrow zwA$	$B \rightarrow v$	$C \rightarrow xA$
$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	$C \rightarrow \varepsilon$

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

$S \rightarrow xABy$	$B \rightarrow wC$	$C \rightarrow vB$
$A \rightarrow zwA$	$B \rightarrow v$	$C \rightarrow xA$
$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	$C \rightarrow \varepsilon$

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

$S \rightarrow xABy$	$B \rightarrow wC$	$C \rightarrow vB$
$A \rightarrow zwA$	$B \rightarrow v$	$C \rightarrow xA$
$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	$C \rightarrow \varepsilon$

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

$S \rightarrow xABy$

$A \rightarrow zwA$

$A \rightarrow \varepsilon$

$B \rightarrow wC$

$B \rightarrow v$

$B \rightarrow \varepsilon$

$C \rightarrow vB$

$C \rightarrow xA$

$C \rightarrow \varepsilon$

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

$S \rightarrow xABy$	$B \rightarrow wC$	$C \rightarrow vB$
$A \rightarrow zwA$	$B \rightarrow v$	$C \rightarrow xA$
$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	$C \rightarrow \varepsilon$

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

$S \rightarrow xABy$

$A \rightarrow zwA$

$A \rightarrow \varepsilon$

$B \rightarrow wC$

$B \rightarrow v$

$B \rightarrow \varepsilon$

$C \rightarrow vB$

$C \rightarrow xA$

~~$C \rightarrow \varepsilon$~~

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

$S \rightarrow xABy$	$B \rightarrow wC$	$C \rightarrow vB$
$A \rightarrow zwA$	$B \rightarrow v$	$C \rightarrow xA$
$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	$C \rightarrow \varepsilon$

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

$S \rightarrow xABy$	$S \rightarrow xABy$
$A \rightarrow zwA$	$A \rightarrow zwA$
$A \rightarrow \varepsilon$	$A \rightarrow \varepsilon$
$B \rightarrow wC$	**$B \rightarrow wC$**
$B \rightarrow v$	**$B \rightarrow w$**
$B \rightarrow \varepsilon$	$B \rightarrow v$
$C \rightarrow vB$	$B \rightarrow \varepsilon$
$C \rightarrow xA$	$C \rightarrow vB$
~~$C \rightarrow \varepsilon$~~	$C \rightarrow xA$

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

$S \rightarrow xABy$	$B \rightarrow wC$	$C \rightarrow vB$
$A \rightarrow zwA$	$B \rightarrow v$	$C \rightarrow xA$
$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	$C \rightarrow \varepsilon$

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

$S \rightarrow xABy$	$S \rightarrow xABy$
$A \rightarrow zwA$	$A \rightarrow zwA$
$A \rightarrow \varepsilon$	$A \rightarrow \varepsilon$
$B \rightarrow wC$	**$B \rightarrow wC$**
$B \rightarrow v$	**$B \rightarrow w$**
$B \rightarrow \varepsilon$	$B \rightarrow v$
$C \rightarrow vB$	~~$B \rightarrow \varepsilon$~~
$C \rightarrow xA$	$C \rightarrow vB$
~~$C \rightarrow \varepsilon$~~	$C \rightarrow xA$

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

S→xABy	B→wC	C→vB
A→zwA	B→v	C→xA
A→ε	B→ε	C→ε

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

S→xABy	S→xABy	**S→xABy** C→xA
A→zwA	A→zwA	**S→xAy**
A→ε	A→ε	A→zwA
B→wC	**B→wC**	A→ε
B→v	**B→w**	B→wC
B→ε	B→v	B→w
C→vB	~~B→ε~~	B→v
C→xA	C→vB	**C→vB**
~~C→ε~~	C→xA	C→v

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

S→xABy	B→wC	C→vB
A→zwA	B→v	C→xA
A→ε	B→ε	C→ε

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

S→xABy	S→xABy	**S→xABy**	C→xA
A→zwA	A→zwA	**S→xAy**	
A→ε	A→ε	A→zwA	
B→wC	**B→wC**	~~A→ε~~	
B→v	**B→w**	B→wC	
B→ε	B→v	B→w	
C→vB	~~B→ε~~	B→v	
C→xA	C→vB	**C→vB**	
~~C→ε~~	C→xA	C→v	

Task 25

- Construct a Pushdown Automata that accepts sequences generated by the following grammar:

S→xABy	B→wC	C→vB
A→zwA	B→v	C→xA
A→ε	B→ε	C→ε

Preparation step: Simplifying a grammar using Greibach normal form

First, eliminate all ε productions.

S→xABy	S→xABy	**S→xABy** C→xA	**S→xABy** C→vB
A→zwA	A→zwA	**S→xAy**	**S→xBy** C→v
A→ε	A→ε	A→zwA	**S→xAy** **C→xA**
B→wC	**B→wC**	~~A→ε~~	**S→xy** **C→x**
B→v	**B→w**	B→wC	**A→zwA**
B→ε	B→v	B→w	**A→zw**
C→vB	~~B→ε~~	B→v	B→wC
C→xA	C→vB	**C→vB**	B→w
~~C→ε~~	C→xA	C→v	B→v

Task 25

Replacing terminal symbols that are not the first on the right side of productions

Task 25

Replacing terminal symbols that are not the first on the right side of productions

$S \rightarrow xABy$

$S \rightarrow xBy$

$S \rightarrow xAy$

$S \rightarrow xy$

$A \rightarrow zwA$

$A \rightarrow zw$

$B \rightarrow wC$

$B \rightarrow w$

$B \rightarrow v$

$C \rightarrow vB$

$C \rightarrow v$

$C \rightarrow xA$

$C \rightarrow x$

Task 25

Replacing terminal symbols that are not the first on the right side of productions

$S \rightarrow xABy$

$S \rightarrow xBy$

$S \rightarrow xAy$

$S \rightarrow xy$

$A \rightarrow zwA$

$A \rightarrow zw$

$B \rightarrow wC$

$B \rightarrow w$

$B \rightarrow v$

$C \rightarrow vB$

$C \rightarrow v$

$C \rightarrow xA$

$C \rightarrow x$

Task 25

Replacing terminal symbols that are not the first on the right side of productions

$S \rightarrow xAB\mathbf{y}$	$S \rightarrow xABD \quad D \rightarrow y$
$S \rightarrow xB\mathbf{y}$	$S \rightarrow xBD$
$S \rightarrow xA\mathbf{y}$	$S \rightarrow xAD$
$S \rightarrow x\mathbf{y}$	$S \rightarrow xD$
$A \rightarrow zwA$	$A \rightarrow zwA$
$A \rightarrow zw$	$A \rightarrow zw$
$B \rightarrow wC$	$B \rightarrow wC$
$B \rightarrow w$	$B \rightarrow w$
$B \rightarrow v$	$B \rightarrow v$
$C \rightarrow vB$	$C \rightarrow vB$
$C \rightarrow v$	$C \rightarrow v$
$C \rightarrow xA$	$C \rightarrow xA$
$C \rightarrow x$	$C \rightarrow x$

Task 25

Replacing terminal symbols that are not the first on the right side of productions

S→xABy	S→xABD D→y
S→xBy	S→xBD
S→xAy	S→xAD
S→xy	S→xD
A→zwA	A→zwA
A→zw	A→zw
B→wC	B→wC
B→w	B→w
B→v	B→v
C→vB	C→vB
C→v	C→v
C→xA	C→xA
C→x	C→x

Task 25

Replacing terminal symbols that are not the first on the right side of productions

S→xAB**y**	S→xABD D→y	S→xABD D→y
S→x**B**y	S→xBD	S→xBD E→w
S→x**A**y	S→xAD	S→xAD
S→x**y**	S→xD	S→xD
A→zwA	A→z**w**A	A→zEA
A→zw	A→z**w**	A→zE
B→wC	B→wC	B→wC
B→w	B→w	B→w
B→v	B→v	B→v
C→vB	C→vB	C→vB
C→v	C→v	C→v
C→xA	C→xA	C→xA
C→x	C→x	C→x

Task 25

Constructing Pushdown Automata:

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S)

$S \rightarrow xABD$

$S \rightarrow xBD$

$S \rightarrow xAD$

$S \rightarrow xD$

$A \rightarrow zEA$

$A \rightarrow zE$

$B \rightarrow wC$

$B \rightarrow w$

$B \rightarrow v$

$C \rightarrow vB$

$C \rightarrow v$

$C \rightarrow xA$

$C \rightarrow x$

$D \rightarrow y$

$E \rightarrow w$

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD

S→xBD

S→xAD

S→xD

A→zEA

A→zE

B→wC

B→w

B→v

C→vB

C→v

C→xA

C→x

D→y

E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD

S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD

S→xD

A→zEA

A→zE

B→wC

B→w

B→v

C→vB

C→v

C→xA

C→x

D→y

E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD

S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD

S→xD

δ(q,z,A)={(q,EA),(q,E)}

A→zEA

A→zE

B→wC

B→w

B→v

C→vB

C→v

C→xA

C→x

D→y

E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD

S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD

S→xD

δ(q,z,A)={(q,EA),(q,E)}

A→zEA

A→zE

δ(q,w,B)={(q,C),(q,ε)}

B→wC

B→w

B→v

C→vB

C→v

C→xA

C→x

D→y

E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD

S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD

S→xD

δ(q,z,A)={(q,EA),(q,E)}

A→zEA

A→zE

δ(q,w,B)={(q,C),(q,ε)}

B→wC

δ(q,v,B)=(q,ε)

B→w

B→v

C→vB

C→v

C→xA

C→x

D→y

E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD

S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD

S→xD

δ(q,z,A)={(q,EA),(q,E)}

A→zEA

A→zE

δ(q,w,B)={(q,C),(q,ε)}

B→wC

δ(q,v,B)=(q,ε)

B→w

B→v

δ(q,v,C)={(q,B),(q,ε)}

C→vB

C→v

C→xA

C→x

D→y

E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD
S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD
S→xD

δ(q,z,A)={(q,EA),(q,E)}

A→zEA
A→zE

δ(q,w,B)={(q,C),(q,ε)}

B→wC

δ(q,v,B)=(q,ε)

B→w
B→v

δ(q,v,C)={(q,B),(q,ε)}

C→vB
C→v

δ(q,x,C)={(q,A),(q,ε)}

C→xA
C→x
D→y
E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD
S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD
S→xD

δ(q,z,A)={(q,EA),(q,E)}

A→zEA
A→zE

δ(q,w,B)={(q,C),(q,ε)}

B→wC

δ(q,v,B)=(q,ε)

B→w
B→v

δ(q,v,C)={(q,B),(q,ε)}

C→vB
C→v

δ(q,x,C)={(q,A),(q,ε)}

C→xA

δ(q,y,D)=(q,ε)

C→x
D→y
E→w

Task 25

Constructing Pushdown Automata:

G=(V,T,P,S) **M=({q},Σ,Γ,δ,q,S,∅)**

Σ=T, Γ=V te za A→bγ ⇒ δ(q,b,A)=(q,γ)

S→xABD
S→xBD

δ(q,x,S)={(q,ABD),(q,BD),(q,AD),(q,D)}

S→xAD
S→xD

δ(q,z,A)={(q,EA),(q,E)}

A→zEA
A→zE

δ(q,w,B)={(q,C),(q,ε)}

B→wC

δ(q,v,B)=(q,ε)

B→w
B→v

δ(q,v,C)={(q,B),(q,ε)}

C→vB
C→v

δ(q,x,C)={(q,A),(q,ε)}

C→xA

δ(q,y,D)=(q,ε)

C→x

δ(q,w,E)=(q,ε)

D→y
E→w

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q,\ A\in\Gamma\}$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[\ q_0$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q,\ A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[\ q_0$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[\ q_0\ \ K$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[\ q_0\ \ K$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[\ q_0\ \ K\ \ q_1\ ]$

Task 26

- Construct context-free grammar that generates sequences accepted by the following Pushdown Automata M:

$M=(\{q_0,q_1\},\ \{a,b,c\},\ \{A,K\},\ \delta,\ q_0,\ K,\ \varnothing)$

$\delta(q_0,b,K)=(q_0,AK)$

$\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$

$\delta(q_0,a,A)=(q_1,A)$

$\delta(q_0,b,A)=(q_0,AA)$

$\delta(q_1,c,A)=(q_1,\varepsilon)$

In order to construct this grammar, a PA must accept by empty stack:

$G=(V,T,P,S)$, where $T=\Sigma$, a $V=\{S\}\cup\{[q_iAq_j] \mid q_i,q_j\in Q, A\in\Gamma\}$

Introducing start productions from the start nonterminal symbol S:

$[\ q_0\ K\ q_i\ ],\ \forall q_i\in Q$

$S\rightarrow[\ q_0\ \ K\ \ q_0\ ]$

$S\rightarrow[\ q_0\ \ K\ \ q_1\ ]$

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

For transition $\delta(q_j,a,X)=(q_k,ABC...Z)$ the following nonterminal symbols and productions are introduced:

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

For transition $\delta(q_j,a,X)=(q_k,ABC...Z)$ the following nonterminal symbols and productions are introduced:

$[q_jXq_e] \rightarrow a[q_kAq_f][q_fBq_g][q_gCq_h]...[q_iZq_e]$. $\forall q_e,q_f,q_g,q_h,q_i \in Q$

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

For transition $\delta(q_j,a,X)=(q_k,ABC...Z)$ the following nonterminal symbols and productions are introduced:

$[q_jXq_e] \rightarrow a[q_kAq_f][q_fBq_g][q_gCq_h]...[q_iZq_e]$. $\forall q_e,q_f,q_g,q_h,q_i \in Q$

If $|Q|=n$ i $|ABC...Z|=m$ then we get n^m productions from one transition.

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

For transition $\delta(q_j,a,X)=(q_k,ABC...Z)$ the following nonterminal symbols and productions are introduced:

$[q_jXq_e] \rightarrow a[q_kAq_f][q_fBq_g][q_gCq_h]...[q_iZq_e]$. $\forall q_e,q_f,q_g,q_h,q_i \in Q$

If $|Q|=n$ i $|ABC...Z|=m$ then we get n^m productions from one transition.

Special case if $m=0$:

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

For transition $\delta(q_j,a,X)=(q_k,ABC...Z)$ the following nonterminal symbols and productions are introduced:

$[q_jXq_e] \rightarrow a[q_kAq_f][q_fBq_g][q_gCq_h]...[q_iZq_e]$. $\forall q_e,q_f,q_g,q_h,q_i \in Q$

If $|Q|=n$ i $|ABC...Z|=m$ then we get n^m productions from one transition.

Special case if m=0:

$\delta(q_j,a,X)=(q_k,\varepsilon) \Rightarrow [q_jXq_k] \rightarrow a,\ a \in T \cup \{\varepsilon\}$

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

For transition $\delta(q_j,a,X)=(q_k,ABC...Z)$ the following nonterminal symbols and productions are introduced:

$[q_jXq_e]\rightarrow a[q_kAq_f][q_fBq_g][q_gCq_h]...[q_iZq_e]$. $\forall q_e,q_f,q_g,q_h,q_i\in Q$

If $|Q|=n$ i $|ABC...Z|=m$ then we get n^m productions from one transition.

Special case if m=0:

$\delta(q_j,a,X)=(q_k,\varepsilon) \Rightarrow [q_jXq_k]\rightarrow a,\ a\in T\cup\{\varepsilon\}$

For transition $\delta(q_0,b,K)=(q_0,AK)$ we include:

$[q_0Kq_0]\rightarrow b[q_0Aq_0][q_0Kq_0]$

$[q_0Kq_0]\rightarrow b[q_0Aq_1][q_1Kq_0]$

$[q_0Kq_1]\rightarrow b[q_0Aq_0][q_0Kq_1]$

$[q_0Kq_1]\rightarrow b[q_0Aq_1][q_1Kq_1]$

Task 26

Other nonterminal symbols and productions of grammar are built based on PA transitions and the following rule:

For transition $\delta(q_j,a,X)=(q_k,ABC...Z)$ the following nonterminal symbols and productions are introduced:

$[q_jXq_e]\rightarrow a[q_kAq_f][q_fBq_g][q_gCq_h]...[q_iZq_e]$. $\forall q_e,q_f,q_g,q_h,q_i\in Q$

If $|Q|=n$ i $|ABC...Z|=m$ then we get n^m productions from one transition.

Special case if m=0:

$\delta(q_j,a,X)=(q_k,\varepsilon) \Rightarrow [q_jXq_k]\rightarrow a,\ a\in T\cup\{\varepsilon\}$

For transition $\delta(q_0,b,K)=(q_0,AK)$ we include:

$[q_0Kq_0]\rightarrow b[q_0Aq_0][q_0Kq_0]$

$[q_0Kq_0]\rightarrow b[q_0Aq_1][q_1Kq_0]$

$[q_0Kq_1]\rightarrow b[q_0Aq_0][q_0Kq_1]$

$[q_0Kq_1]\rightarrow b[q_0Aq_1][q_1Kq_1]$

For transition $\delta(q_0,a,A)=(q_1,A)$ we include:

$[q_0Aq_0]\rightarrow a[q_1Aq_0]$

$[q_0Aq_1]\rightarrow a[q_1Aq_1]$

Task 26

For transition δ(q0,b,A)=(q0,AA) we include:

$[q_0Aq_0]\rightarrow b[q_0Aq_0][q_0Aq_0]$

$[q_0Aq_0]\rightarrow b[q_0Aq_1][q_1Aq_0]$

$[q_0Aq_1]\rightarrow b[q_0Aq_0][q_0Aq_1]$

$[q_0Aq_1]\rightarrow b[q_0Aq_1][q_1Aq_1]$

Task 26

For transition $\delta(q0,b,A)=(q0,AA)$ we include:

$[q_0Aq_0]\rightarrow b[q_0Aq_0][q_0Aq_0]$

$[q_0Aq_0]\rightarrow b[q_0Aq_1][q_1Aq_0]$

$[q_0Aq_1]\rightarrow b[q_0Aq_0][q_0Aq_1]$

$[q_0Aq_1]\rightarrow b[q_0Aq_1][q_1Aq_1]$

For transition $\delta(q_1,c,A)=(q_1,\varepsilon)$ we include:

$[q_1Aq_1]\rightarrow c$

Task 26

For transition $\delta(q0,b,A)=(q0,AA)$ we include:

$[q_0Aq_0]\rightarrow b[q_0Aq_0][q_0Aq_0]$ $\quad$ $[q_0Aq_1]\rightarrow b[q_0Aq_0][q_0Aq_1]$

$[q_0Aq_0]\rightarrow b[q_0Aq_1][q_1Aq_0]$ $\quad$ $[q_0Aq_1]\rightarrow b[q_0Aq_1][q_1Aq_1]$

For transition $\delta(q_1,c,A)=(q_1,\varepsilon)$ we include:

$[q_1Aq_1]\rightarrow c$

For transition $\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$ we include:

$[q_1Kq_1]\rightarrow \varepsilon$

Task 26

For transition $\delta(q0,b,A)=(q0,AA)$ we include:

$[q_0Aq_0]\rightarrow b[q_0Aq_0][q_0Aq_0]$
$[q_0Aq_0]\rightarrow b[q_0Aq_1][q_1Aq_0]$
$[q_0Aq_1]\rightarrow b[q_0Aq_0][q_0Aq_1]$
$[q_0Aq_1]\rightarrow b[q_0Aq_1][q_1Aq_1]$

For transition $\delta(q_1,c,A)=(q_1,\varepsilon)$ we include:

$[q_1Aq_1]\rightarrow c$

For transition $\delta(q_1,\varepsilon,K)=(q_1,\varepsilon)$ we include:

$[q_1Kq_1]\rightarrow \varepsilon$

Constructed grammar:

$S\rightarrow[q_0Kq_0]$
$S\rightarrow[q_0Kq_1]$
$[q_0Kq_0]\rightarrow b[q_0Aq_0][q_0Kq_0]$
$[q_0Kq_0]\rightarrow b[q_0Aq_1][q_1Kq_0]$
$[q_0Kq_0]\rightarrow b[q_0Aq_0][q_0Kq_0]$
$[q_0Kq_0]\rightarrow b[q_0Aq_1][q_1Kq_0]$
$[q_0Aq_0]\rightarrow a[q_1Aq_0]$
$[q_0Aq_1]\rightarrow a[q_1Aq_1]$
$[q_0Aq_0]\rightarrow b[q_0Aq_0][q_0Aq_0]$
$[q_0Aq_0]\rightarrow b[q_0Aq_1][q_1Aq_0]$
$[q_0Aq_1]\rightarrow b[q_0Aq_0][q_0Aq_1]$
$[q_0Aq_1]\rightarrow b[q_0Aq_1][q_1Aq_1]$
$[q_1Aq_1]\rightarrow c$
$[q_1Kq_1]\rightarrow \varepsilon$

Task 26

The constructed grammar can have dead and unreachable nonterminal symbols. After eliminating dead and unreachable symbols, the grammar contains productions:

$S \rightarrow [q_0 K q_1]$

$[q_0 K q_1] \rightarrow b[q_0 A q_1][q_1 K q_1]$

$[q_0 A q_1] \rightarrow b[q_0 A q_1][q_1 A q_1]$

$[q_0 A q_1] \rightarrow a[q_1 A q_1]$

$[q_1 A q_1] \rightarrow c$

$[q_1 K q_1] \rightarrow \varepsilon$

Task 26

The constructed grammar can have dead and unreachable nonterminal symbols. After eliminating dead and unreachable symbols, the grammar contains productions:

$S \rightarrow [q_0Kq_1]$

$[q_0Kq_1] \rightarrow b[q_0Aq_1][q_1Kq_1]$

$[q_0Aq_1] \rightarrow b[q_0Aq_1][q_1Aq_1]$

$[q_0Aq_1] \rightarrow a[q_1Aq_1]$

$[q_1Aq_1] \rightarrow c$

$[q_1Kq_1] \rightarrow \varepsilon$

The constructed grammar is more readable if nonterminal symbols are renamed:

$[q_0Kq_1] \rightarrow A$ **$[q_0Aq_1] \rightarrow B$** **$[q_1Aq_1] \rightarrow C$** **$[q_1Kq_1] \rightarrow D$**

$S \rightarrow A$ $B \rightarrow aC$

$A \rightarrow bBD$ $C \rightarrow c$

$B \rightarrow bBC$ $D \rightarrow \varepsilon$