$L = \{a^n b^n : n \geq 0\}$ Using a direct construction.
Start state: s Final states: $\{t\}$
Meaning of states: s: read a's t: read b's

State	Read	Pop	Next State	Push	Comments
s	a	ε	s	B	Push B on stack for each a in first part of the string.
s	ε	ε	t	ε	Switch to reading in b's.
t	b	B	t	ε	Match each b with B on stack.

Some accepting computations:
1. $(s, \varepsilon, \varepsilon) \vdash (t, \varepsilon, \varepsilon)$

2. $(s, aabb, \varepsilon) \vdash (s, abb, B) \vdash (s, bb, BB) \vdash (t, bb, BB) \vdash (t, b, B) \vdash (t, \varepsilon, \varepsilon)$

Start state: s   Final states: {t}

State	Read	Pop	Next State	Push	Comments
s	a	ε	s	B	Push B on stack for each a in first part of the string.
s	ε	ε	t	ε	Switch to reading in b's.
t	b	B	t	ε	Match each b with B on stack.

Some non-accepting computations:

1. (s, a, ε) ⊢ (s, ε, B) ⊢ (t, ε, B)   Stack not empty.
2. (s, abb, ε) ⊢ (s, bb, B) ⊢ (t, bb, B) ⊢ (t, b, ε) Input not consumed.
3. (s, aaba, ε) ⊢ (s, aba, B) ⊢ (s, ba, BB) ⊢ (t, ba, BB) ⊢ (t, a, B) Stuck.

L= {$a^n b^n : n \geq 0$}   Using grammar: S → aSb, S → ε
Start state: s   Final states: {t}
s: Push start symbol   t: apply rules from grammar or read

State	Read	Pop	Next State	Push	Comments
s	ε	ε	t	S	Push start symbol on stack.
t	ε	S	t	aSb	Apply rule: S → aSb
t	ε	S	t	ε	Apply rule: S → ε
t	a	a	t	ε	Match a's in derivation with a's in the input.
t	b	b	t	ε	Match b's in derivation with b's in the input.

State	Read	Pop	Next State	Push	Comments
s	ε	ε	t	S	Push start symbol on stack.
t	ε	S	t	aSb	Apply rule: S → aSb
t	ε	S	t	ε	Apply rule: S → ε
t	a	a	t	ε	Match a's in derivation with input.
t	b	b	t	ε	Match b's in derivation with input.

A derivation: S ⇒ a S b ⇒ a a S b b ⇒ a a b b

The corresponding PDA computation that mimics this derivation from the grammar:

(s, aabb, ε) ⊢ (t, aabb, S) ⊢ (t, aabb, aSb) ⊢ (t, abb, Sb) ⊢ (t, abb, aSbb) ⊢ (t, bb, Sbb) ⊢ (t, bb, bb) ⊢ (t, b, b) ⊢ (t, ε, ε)

Design a PDA that accepts:

$L_1 = \{ a^p b^q c^r : p \neq q \} = L_2 \cup L_3$ where

$L_2 = \{ a^p b^q c^r : p < q \}$ $L_3 = \{ a^p b^q c^r : p > q \}$

Start state: s   Final states: ?

State	Read	Pop	Next State	Push	Comments
s	ε	ε	s1	ε	Use this for p < q.
s	ε	ε	s2	ε	Use this for p > q.

$L_2 = \{ a^p b^q c^r : p < q \}$ Start state: **s1**   Final states: {u1}

s1: read a's   **t1: read b's**   **u1: read c's**

State	Read	Pop	Next State	Push	Comments
s1	a	ε	s1	b	Match these a's with b's
s1	ε	ε	s1	b	Use for extra b's.
s1	ε	ε	t1	b	Switch to reading b's. Pushing b ensures at least one extra b.
t1	b	b	t1	ε	Match b's with b's on stack.
t1	ε	ε	u1	ε	Switch to reading c's.
u1	c	ε	u1	ε	Read in c's.

$L_3 = \{ a^p b^q c^r : p > q \}$ Start state: s2   Final states: {u2}

s2: read a's   t2: read b's   u2: read c's

State	Read	Pop	Next State	Push	Comments
s2	a	ε	s2	b	Match these a's with b's
s2	a	ε	s2	ε	Use for extra a's.
s2	a	ε	t2	ε	Switch to reading b's. Reading *a* ensures at least one extra *a*.
t2	b	b	t2	ε	Match b's with b's on stack.
t2	ε	ε	u2	ε	Switch to reading c's.
u2	c	ε	u2	ε	Read in c's.

Accepting computation for aabbbbcc:

State	Read	Pop	Next State	Push	Comments
s1	a	ε	s1	b	Match these a's with b's
s1	ε	ε	s1	b	Use for extra b's.
s1	ε	ε	t1	b	Switch to reading b's. Pushing b ensures at least one extra b.
t1	b	b	t1	ε	Match b's with b's on stack.
t1	ε	ε	u1	ε	Switch to reading c's.
u1	c	ε	u1	ε	Read in c's.

(s, aabbbbcc, ε) ⊢ (s1, aabbbbcc, ε) ⊢* (s1, bbbbcc, bb) ⊢
(s1, bbbbcc, bbb) ⊢ (t1, bbbbcc, bbbb) ⊢* (t1, cc, ε)
⊢ (u1, cc, ε) ⊢* (u1, ε, ε)

Accepting computation for aaaaabb:

State	Read	Pop	Next State	Push	Comments
s2	a	ε	s2	b	Match these a's with b's
s2	a	ε	s2	ε	Use for extra a's.
s2	a	ε	t2	ε	Switch to reading b's. Reading *a* ensures at least one extra *a*.
t2	b	b	t2	ε	Match b's with b's on stack.
t2	ε	ε	u2	ε	Switch to reading c's.
u2	c	ε	u2	ε	Read in c's.

(s, aaaaabb, ε) ⊢ (s2, aaaaabb, ε) ⊢* (s2, aaabb, bb) ⊢
(s2, abb, bbbb) ⊢ (t2, bb, bb) ⊢* (t2, ε, ε)
⊢ (u2, ε, ε)