# *Pushdown Automata*

*A PDA is an FA together with a stack.*

# *Stacks*

A ***stack*** stores information on the ***last-in first-out*** principle.

Items are added on top by ***pushing***; items are removed from the top by ***popping***.

# A Pushdown Automaton

A ***pushdown automaton*** (PDA) has a fixed set of states (like FA), but it also has one unbounded stack for storage.

When symbol is read, depending on (a) state of automaton, (b) symbol on top of stack, and (c) symbol read, the automaton

1. updates its state, and
2. (optionally) pops or pushes a symbol.

The automaton may also pop or push without reading input.

# *Flowcharts*

We draw the ***program*** of a PDA as a ***flowchart*** (we will see FA-like diagram later). This uses:

- A single ***start*** state;
- A single ***halt-and-accept*** state;
- A ***reader*** box: read one symbol from input and based on that update state (as in FA);
- A ***pop*** box: pop one symbol from stack and based on that update state;
- A ***push*** box: add symbol to stack.

## *Notes*

There is no explicit reject state: if no legal continuation, then PDA halts and rejects.

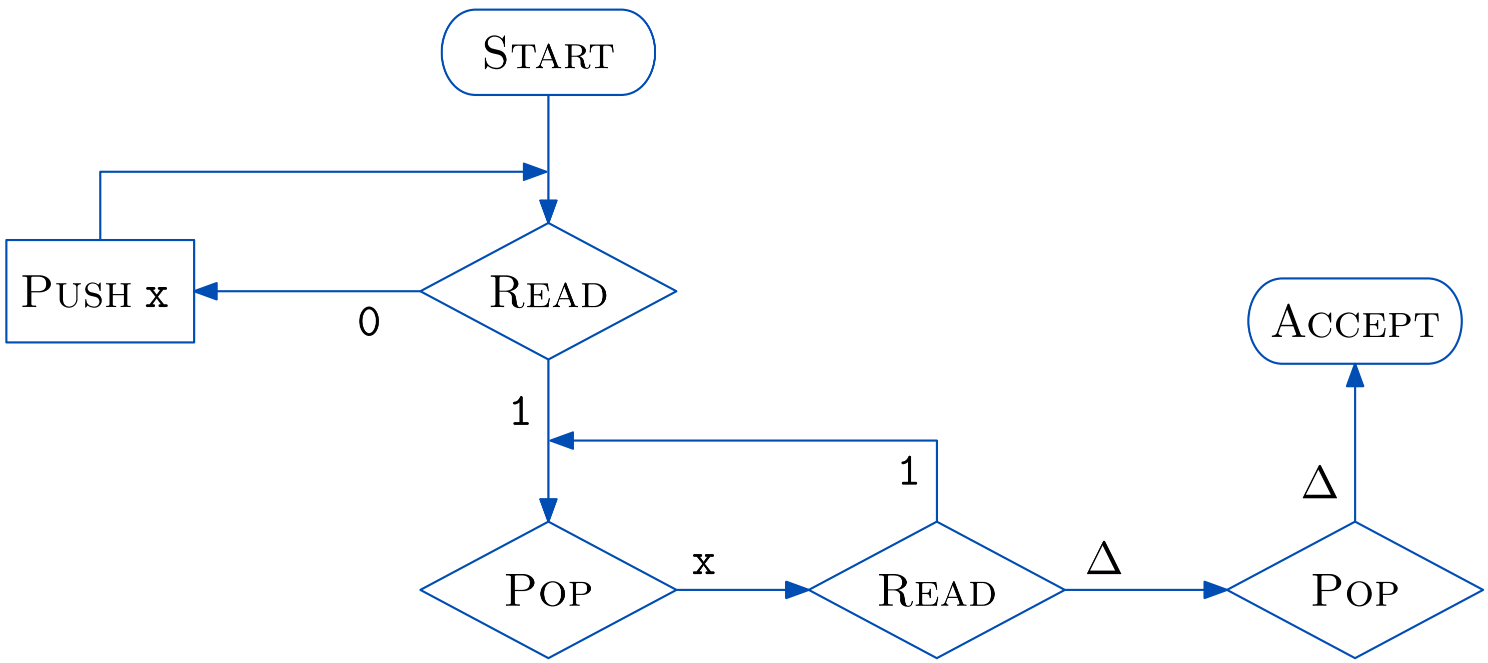
We use symbol $\triangle$ to indicate both the end of input, and the result of popping from an empty stack.

# *Example:* $\mathtt{0}^n\mathtt{1}^n$ *Again*

Consider a PDA for $\{\, \mathtt{0}^n\mathtt{1}^n : n > 0 \,\}$. The PDA uses its stack as counter.

For each `0` read, PDA pushes an `x` (say). When first `1` read, PDA enters new state. Now, it pops one symbol for each `1` read. If now `0` is read or pop from empty stack, it rejects. PDA accepts if and only if stack becomes empty as the input finishes. . .

# *Flowchart for* $0^n1^n$



## *Casualness*

There are traditional shapes for the different types of functions on flowcharts, but we don't worry about that.

Also, $\varepsilon$ often requires very special handling: from now on, however, we will simply *ignore the empty string*.

## *Balanced Brackets*

A string of left and right brackets is ***balanced*** if (a) reading from left to right, number of left brackets is always at least number of right brackets; and (b) total number of left brackets equals total number of right brackets.

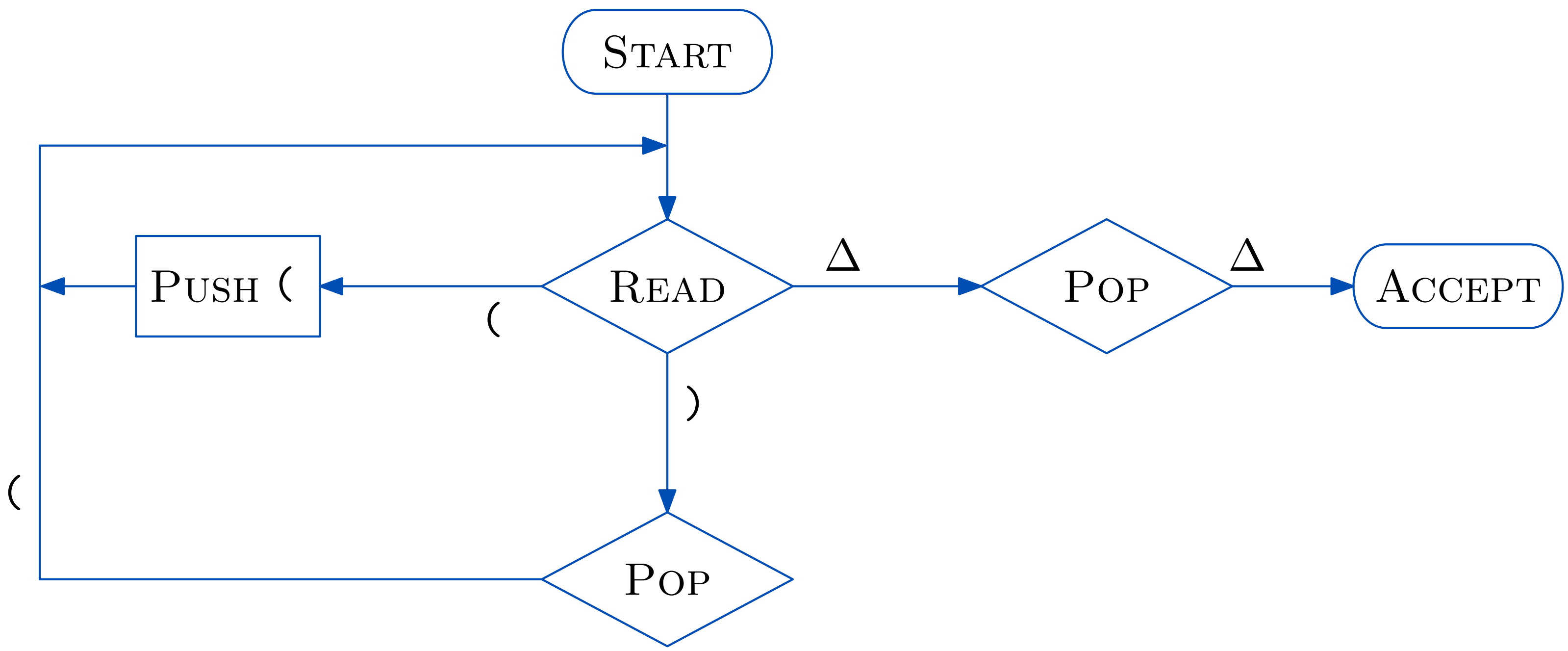
For example, ( ( ) ( ) ) ( ) is balanced; ( ( ) and ) ) ) ( are not.

Here is CFG:

$$S \to (S) \mid SS \mid \varepsilon$$

# *PDA for Balanced Brackets*

In PDA, each ( is pushed; each ) causes a matching ( to be popped.



# *Nondeterminism*

By definition, a PDA is nondeterministic. It accepts the input string if ***there exists*** a sequence of actions leading to the accept state.

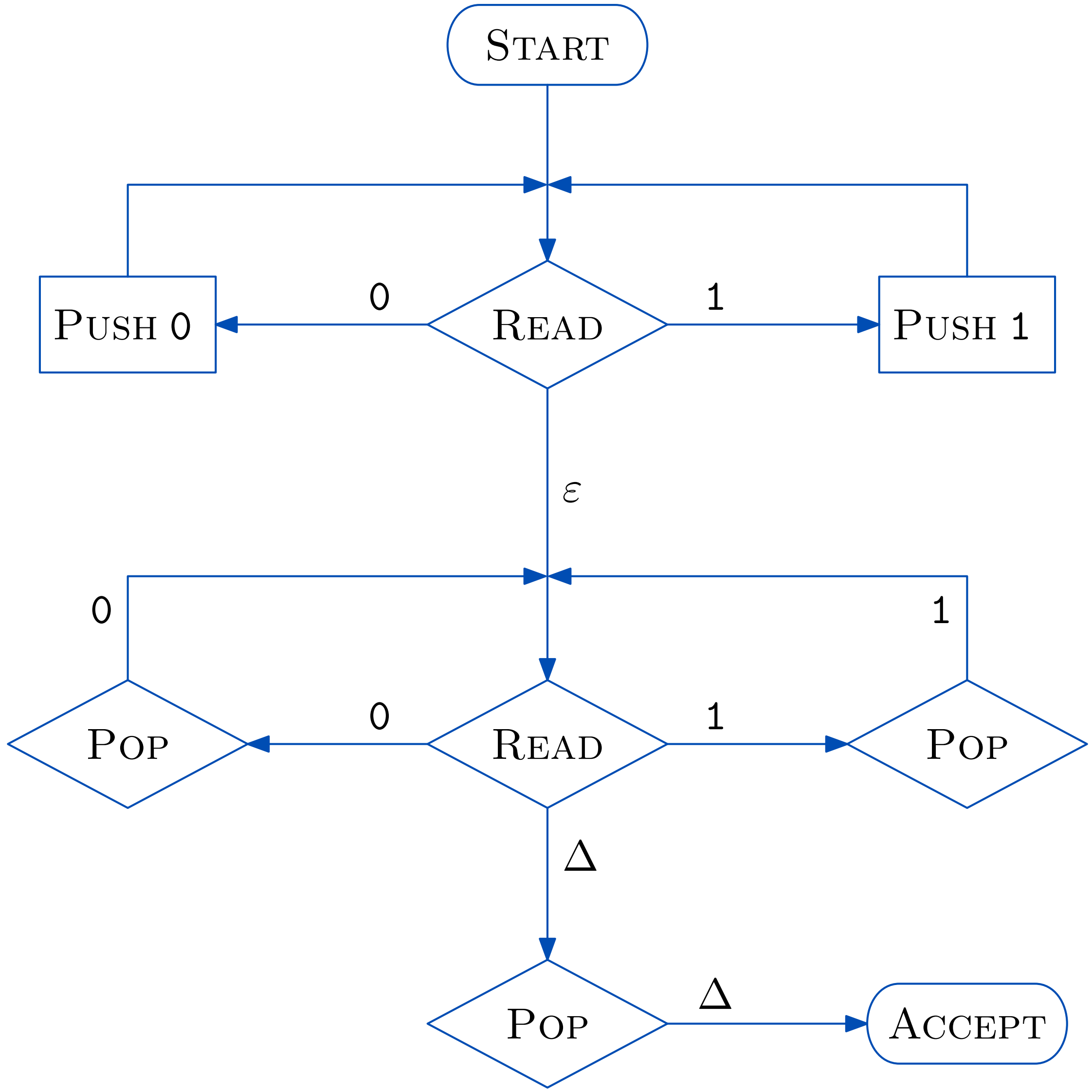
There are two ways to depict nondeterminism in the flowchart: two transitions with the same label, or a transition labeled with $\varepsilon$ (which does not consume an input symbol).

# *PDA for Palindromes*

The PDA for palindromes uses nondeterminism to guess the midpoint of the string; and the stack to compare the first and second halves.

Here is the PDA for even-length palindromes. . .

# *Even-Length Palindromes*



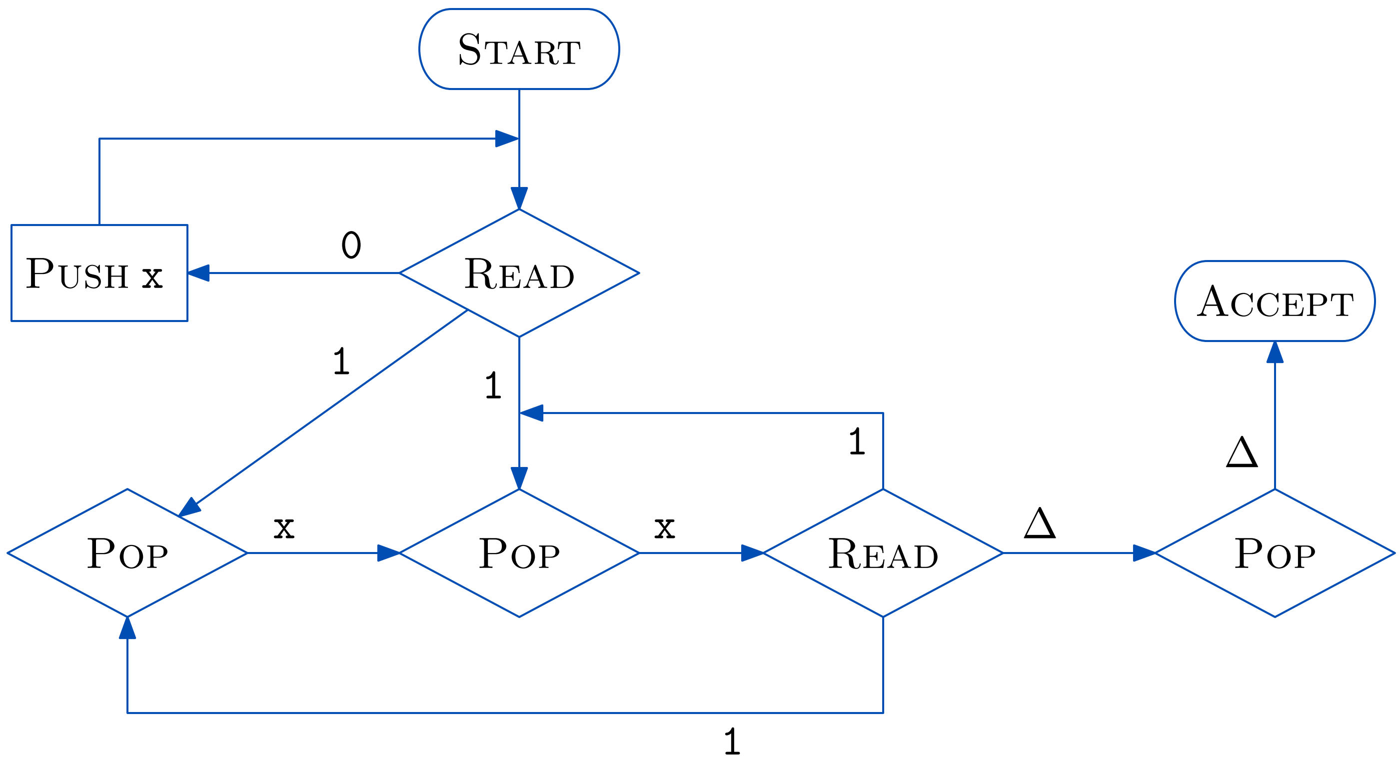
# *Another Example*

Consider the language $\{\, \mathtt{0}^m \mathtt{1}^n : n \leq m \leq 2n \,\}$.

## *Another Example*

Consider the language $\{\, \mathtt{0}^m \mathtt{1}^n : n \leq m \leq 2n \,\}$.

The PDA starts by counting the `0`'s, say using `x`. Then matches each `1` with either one or two `x`'s.

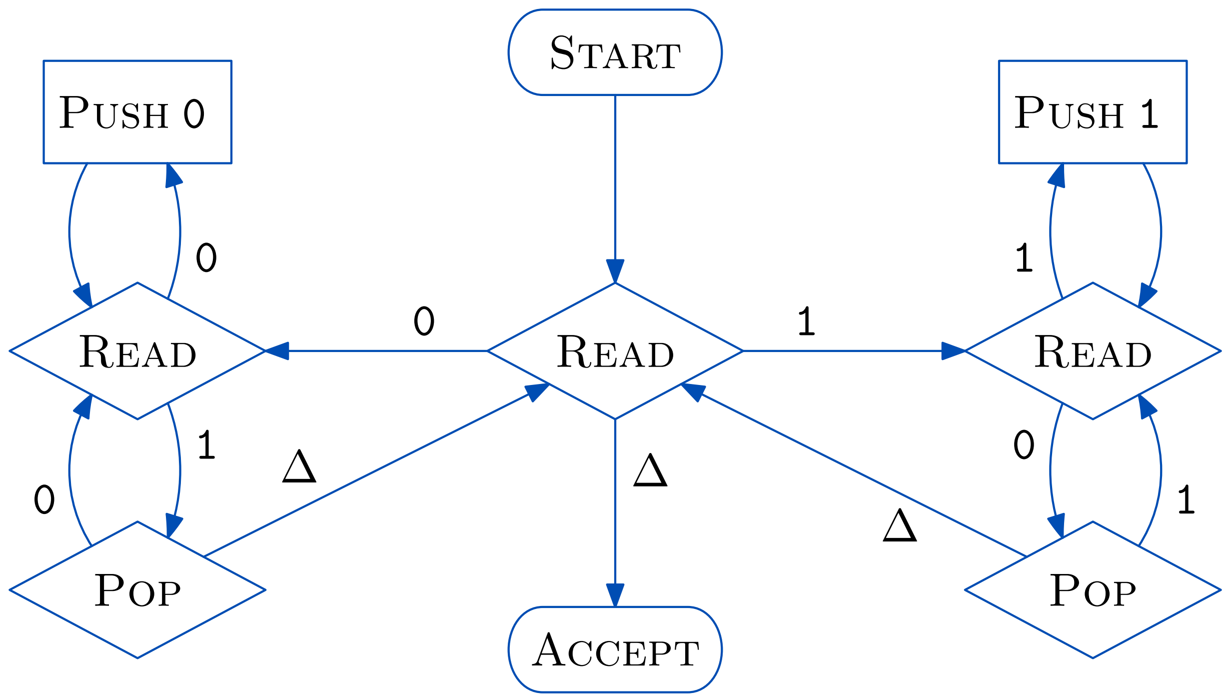


# *PDA for Equality*

Here is PDA for all binary strings with ***equal* `0`*'s and* `1`*'s***.

The PDA again uses stack as counter. Several approaches. One idea is to pair symbols off, storing the excess on the stack. The following PDA actually stores ***one less*** than the excess. . .

# *Flowchart for Equality*



## *Context-Free Languages*

**Theorem.** *A language is generated by a context-free grammar if and only if it is accepted by a pushdown automaton.*

We prove this later.

# *Applications of PDAs: Reverse Polish*

A compiler converts an arithmetic expression into code that can be evaluated using a stack.

For example,

$$1 + 5 * (3 + 2) + 4$$

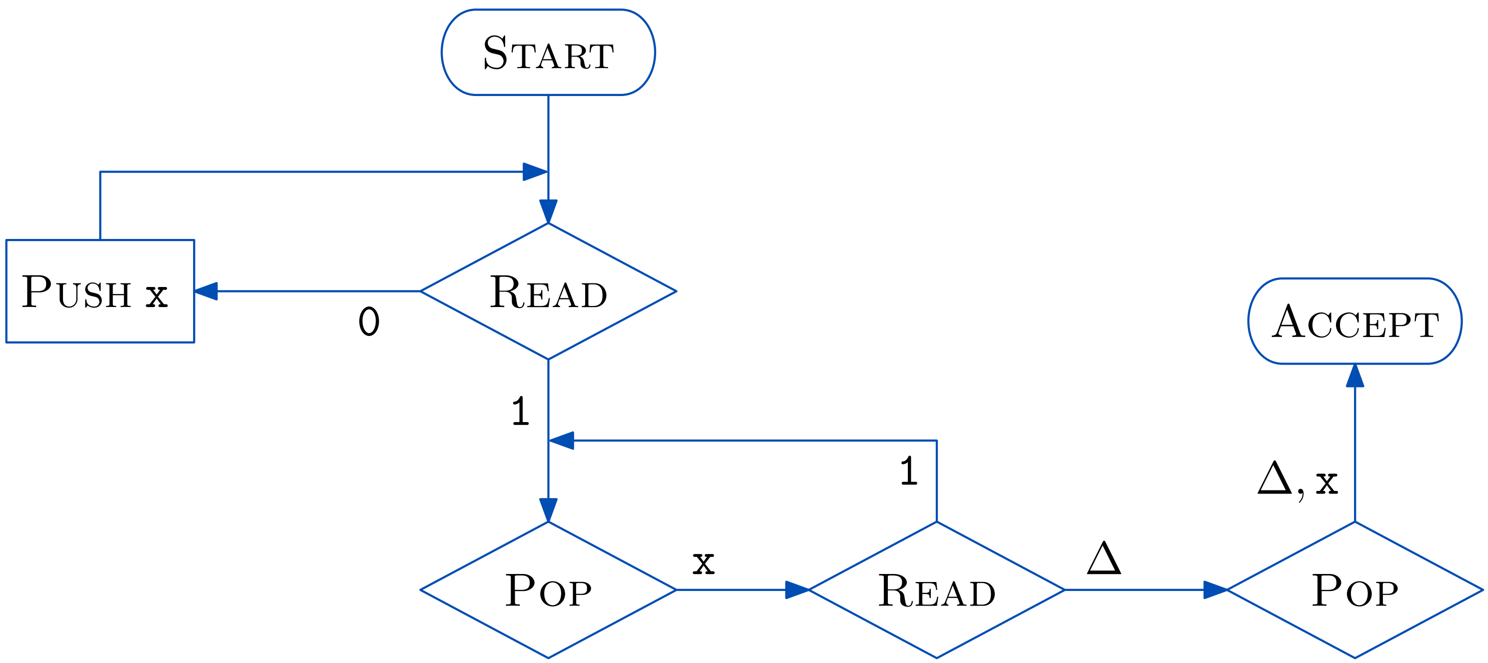
might become

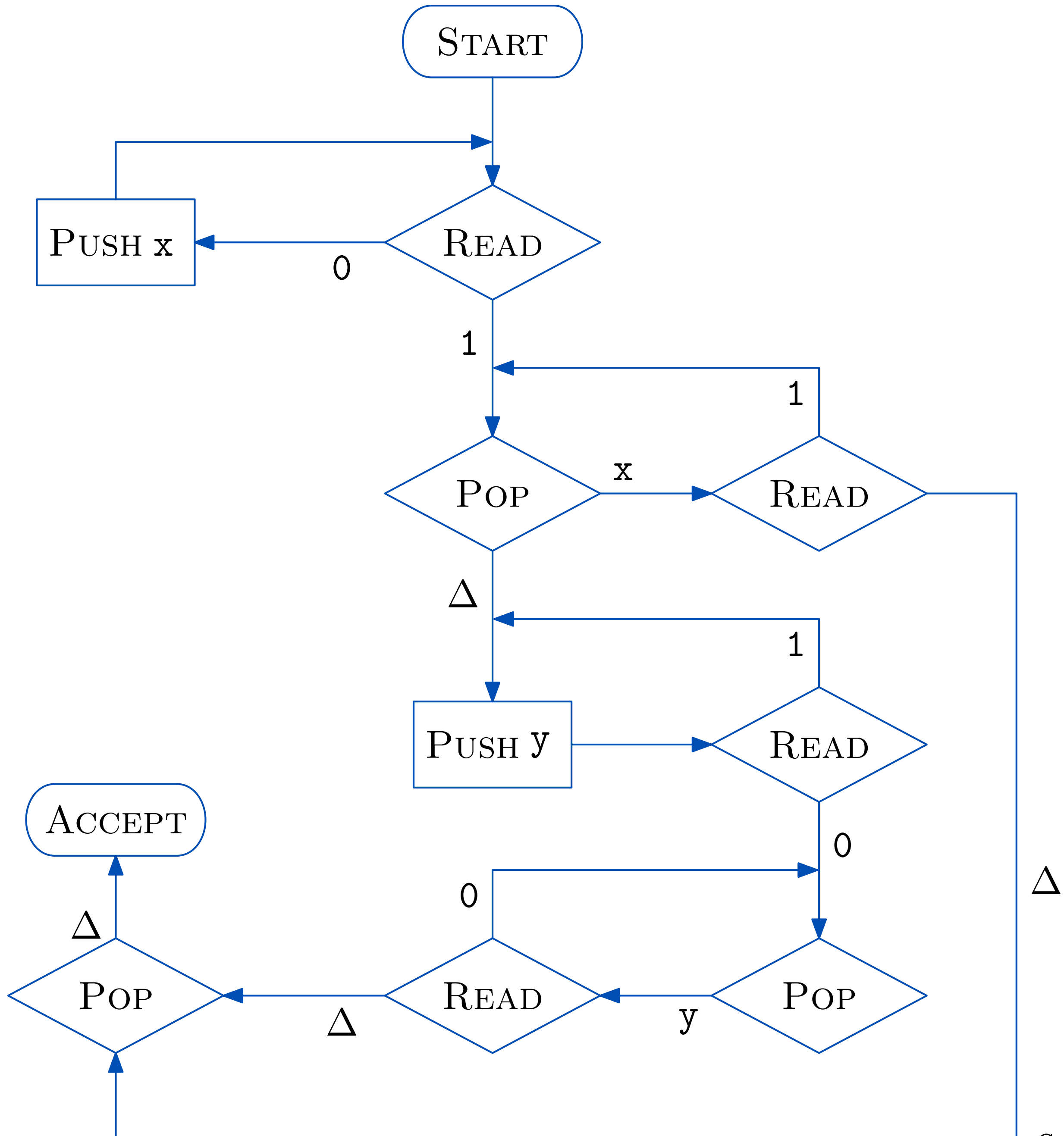
PUSH(1) PUSH(5) PUSH(3) PUSH(2) ADD MUL ADD PUSH(4) ADD

# *Practice*

1. Draw a PDA for the set of all strings of the form $\mathtt{0}^a\mathtt{1}^b$ such that $a \geq b$.
2. Draw a PDA for the set of all strings of the form $\mathtt{0}^a\mathtt{1}^b\mathtt{0}^c$ such that $a + c = b$.

# *Solutions to Practice*




START
READ
PUSH x
0
1
POP
x
READ
1
Δ
PUSH y
READ
1
0
ACCEPT
Δ
POP
READ
0
Δ
POP
y
Δ

# *Summary*

A pushdown automaton (PDA) is an FA with a stack added for storage. We choose to draw these as flowcharts where the character $\Delta$ indicates both empty stack and end-of-input. A PDA is nondeterministic by definition.