Code Generation Within The Parser

- It is easy to augment the parser so that it generates SaM code. E.g., let's do this for evaluating arithmetic expressions.

2

```
PUSHIMM 2
STOP
```

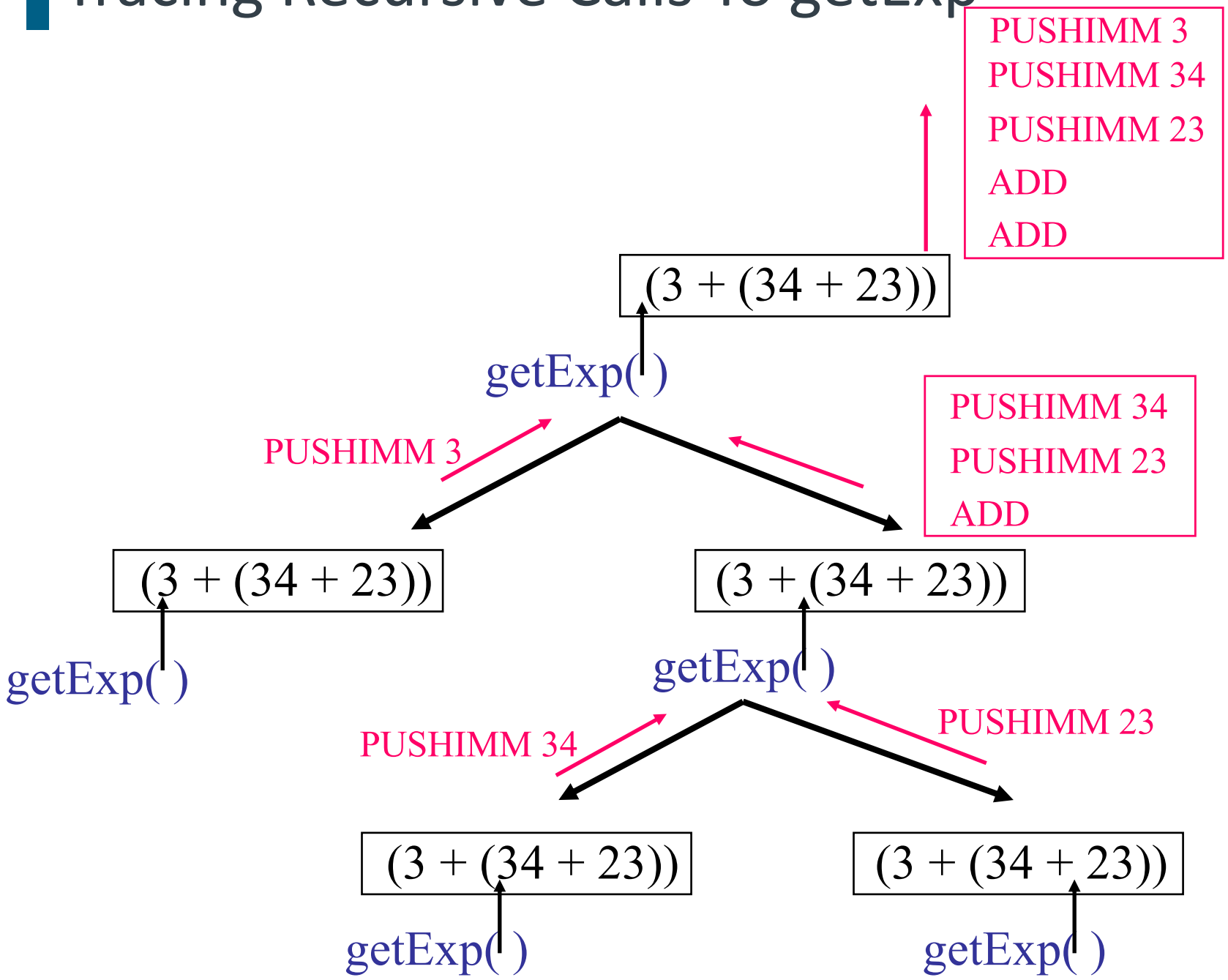
(2+3)

```
PUSHIMM 2
PUSHIMM 3
ADD
STOP
```

Augmenting The Parser

- Recursive method `getExp` should return a `String` containing SaM code for the (sub-)expression it has parsed.
- The top-level method `expParser` should tack on a `STOP` command after the code that it receives from `getExp`.
- Method `getExp` generates code in a recursive manner.
  - For integer k, it returns the `String` "PUSHIMM " + k + "\n".
  - For (E1 + E2):
    - The recursive calls to generate code for E1 and E2 return the `Strings` S1 and S2.
    - Method returns S1 + S2 + "ADD\n".

Tracing Recursive Calls To getExp



Shapes for Recursive Code Generation

Construct	Code
num	PUSHIMM val
x	PUSHOFF yy
(e1 + e2)	Code for e1 Code for e2 ADD
x = e;	Code for e STOREOFF yy
{S1 … Sn}	Code for S1 … Code for Sn