Lecture 14: Recursive Backtracking II

07/27/22

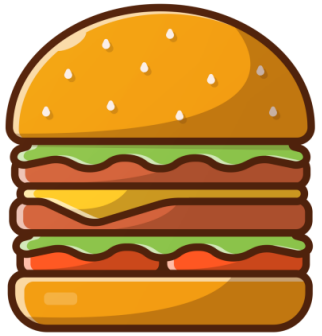
Reminders

- A3 Resubmission due Wednesday 7/27 @ 11:59pm
- A5 due Thursday 7/28 @ 11:59pm

Spend all your Husky card money!

Balance: $3

$1



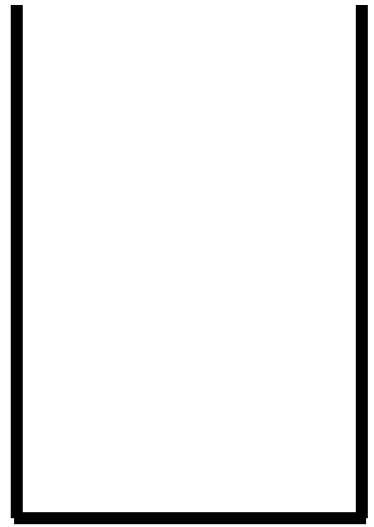
$2

$3

$5

chosen: 📞
dollarsLeft: 3

Output:



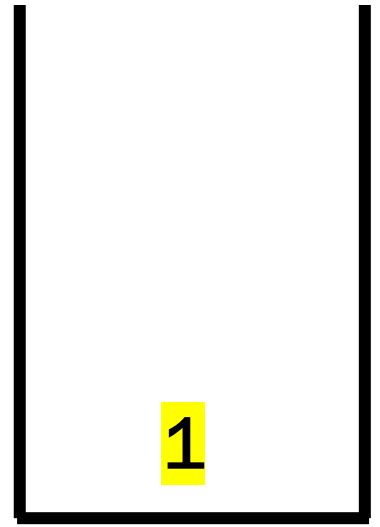
chosen

Phone Number: 425-123-4567

chosen: 📞
dollarsLeft: 3

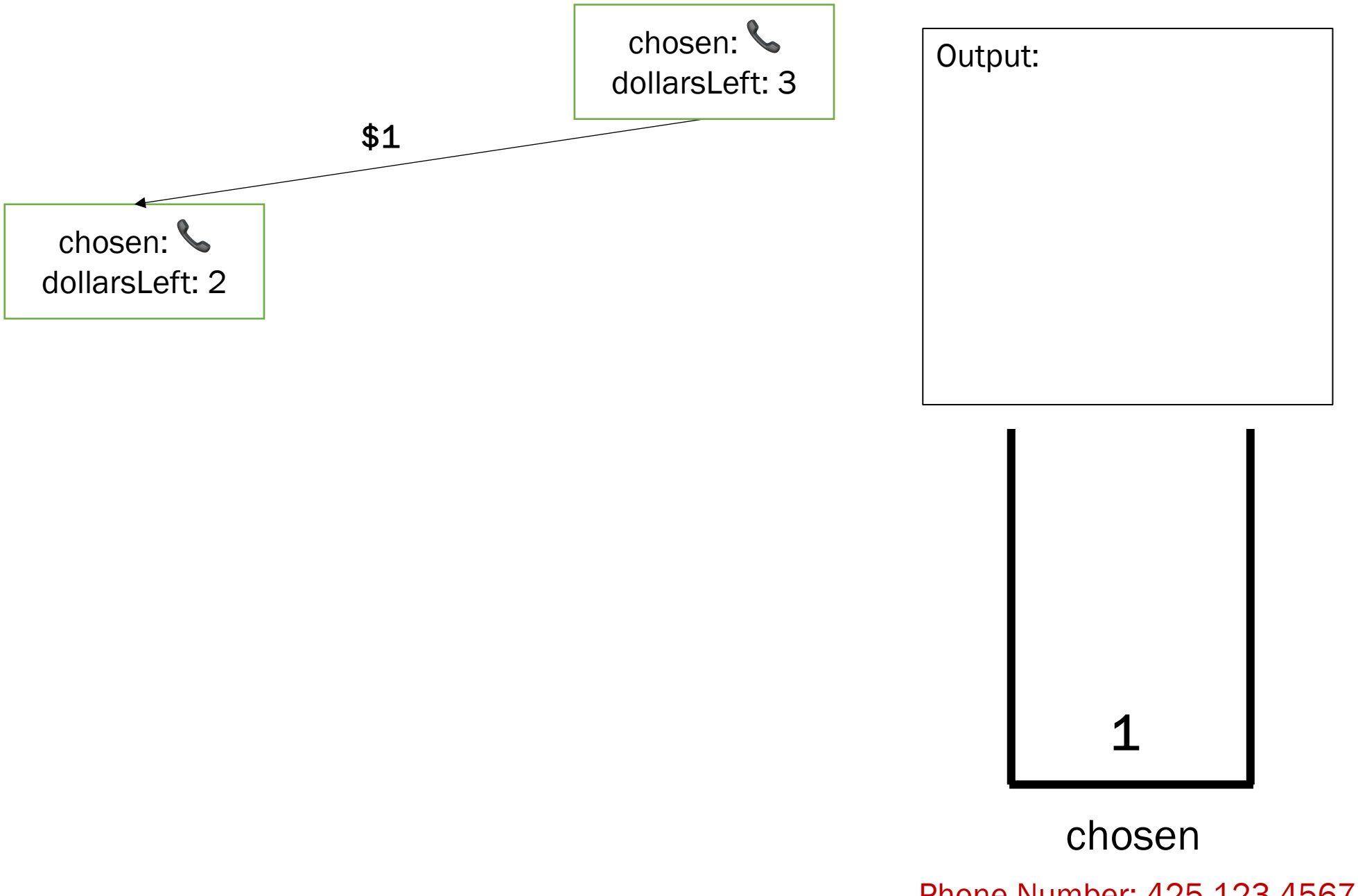
`$1`

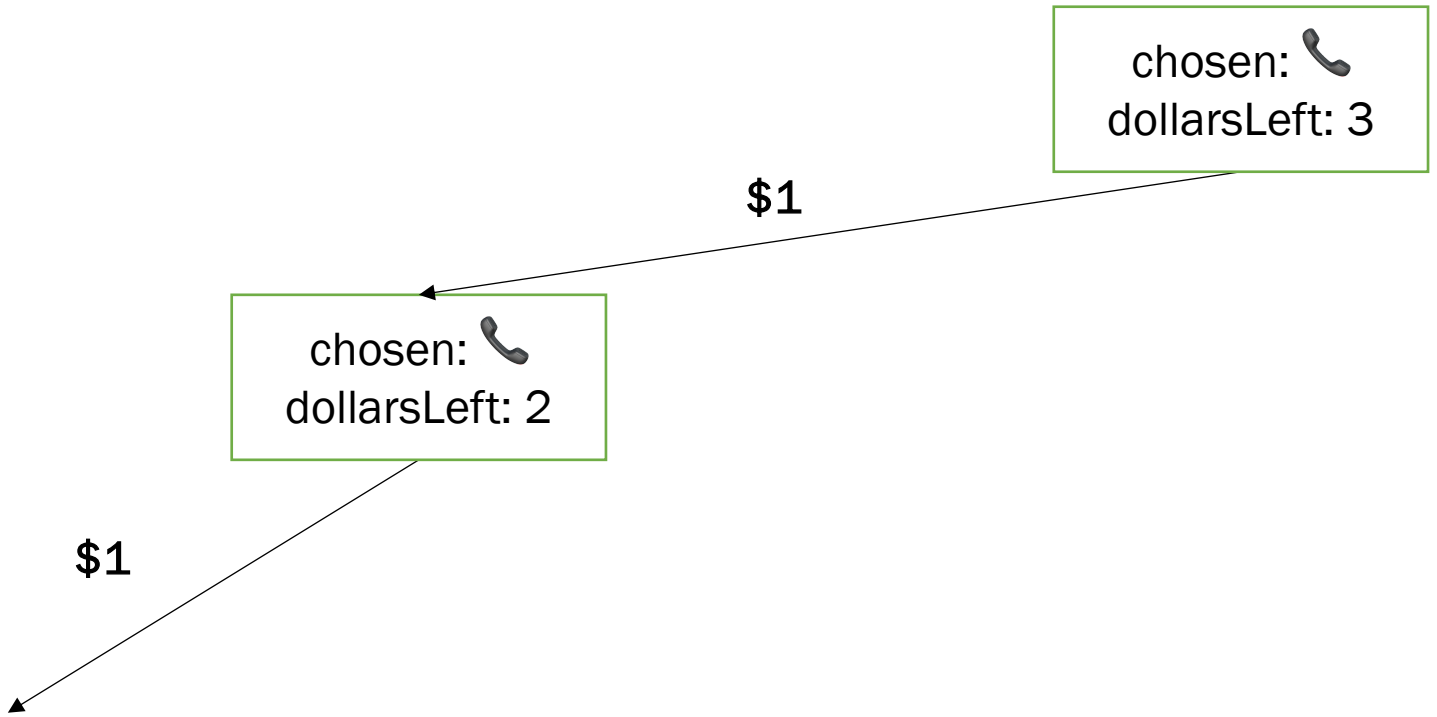
Output:



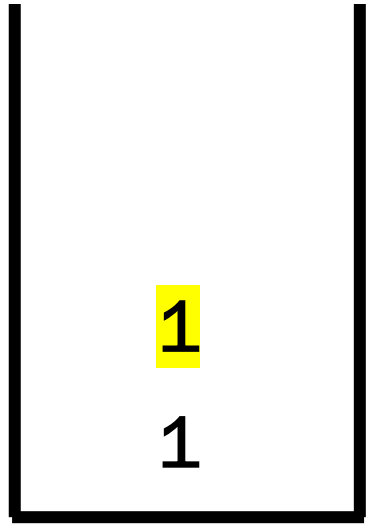
chosen

Phone Number: 425-123-4567


chosen:
dollarsLeft: 3
$1
chosen:
dollarsLeft: 2
Output:
1
chosen
Phone Number: 425-123-4567

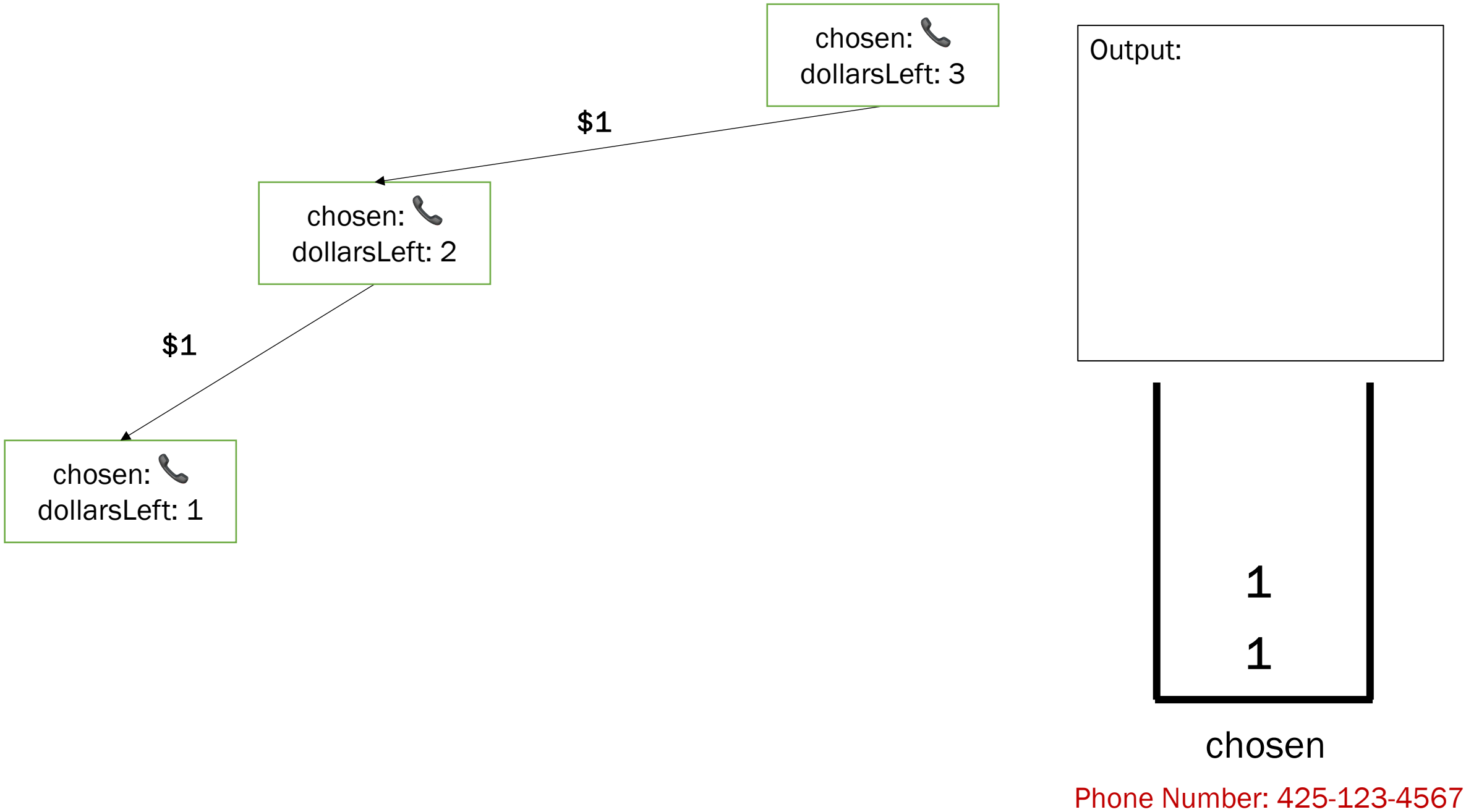


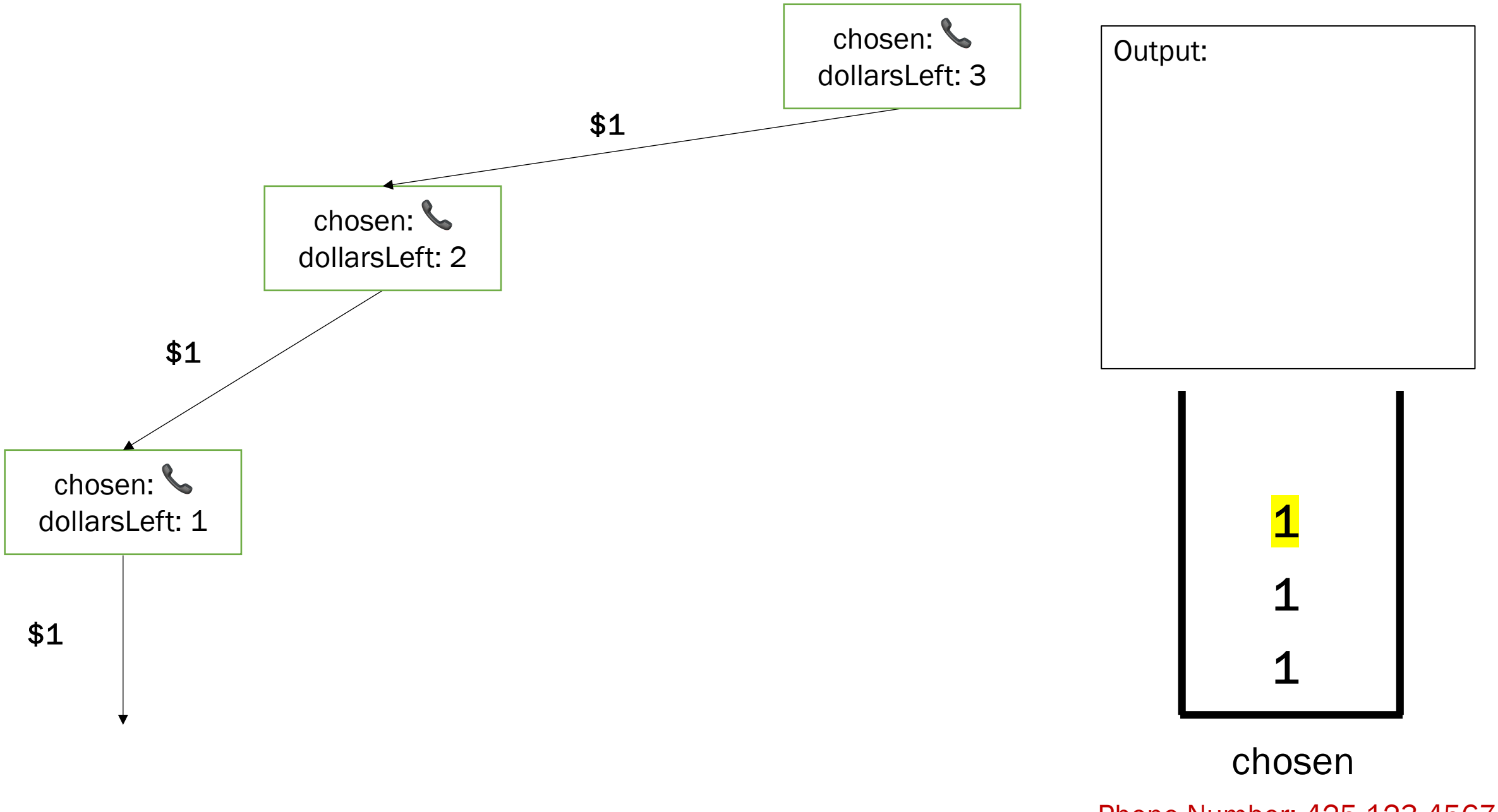
Output:



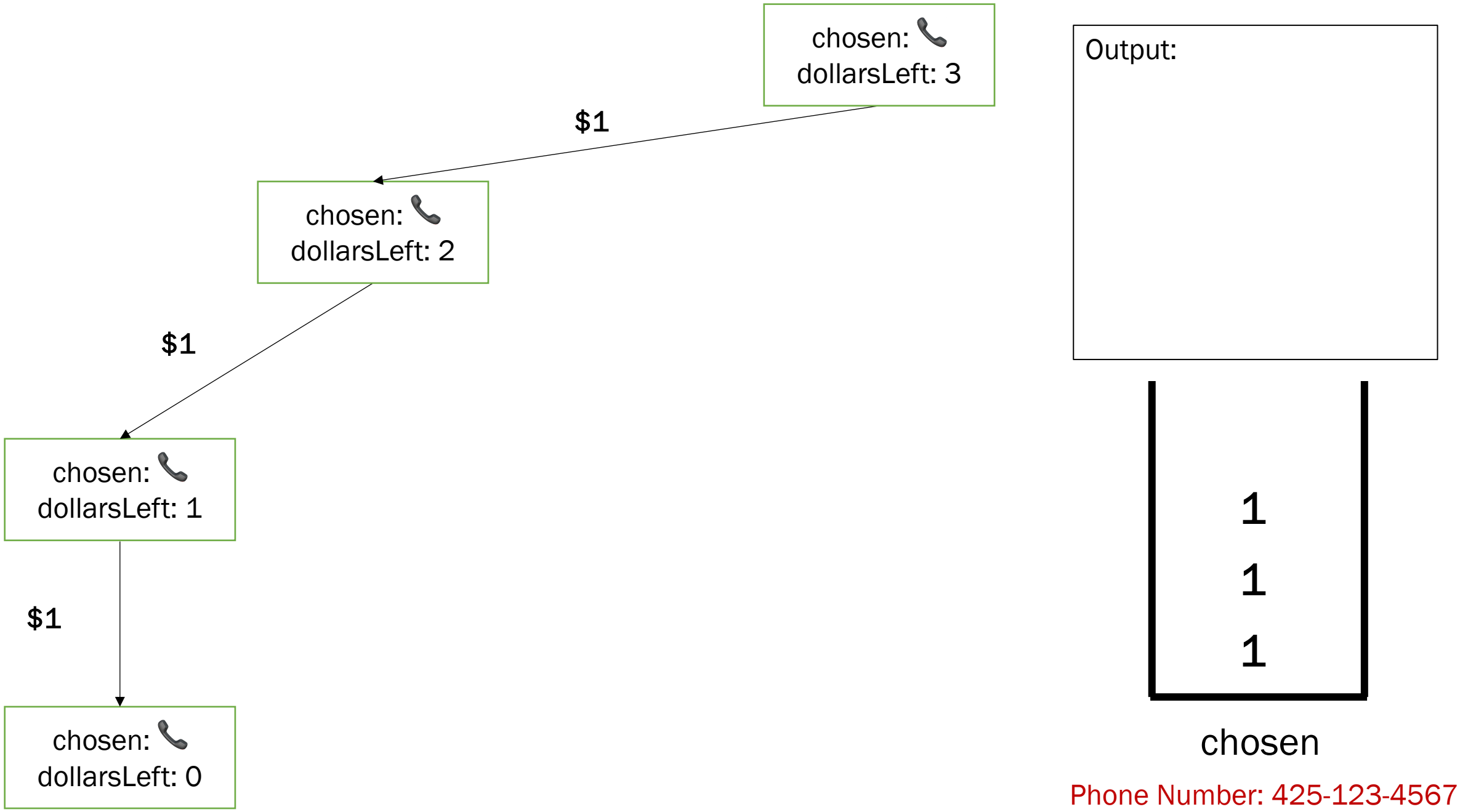
chosen

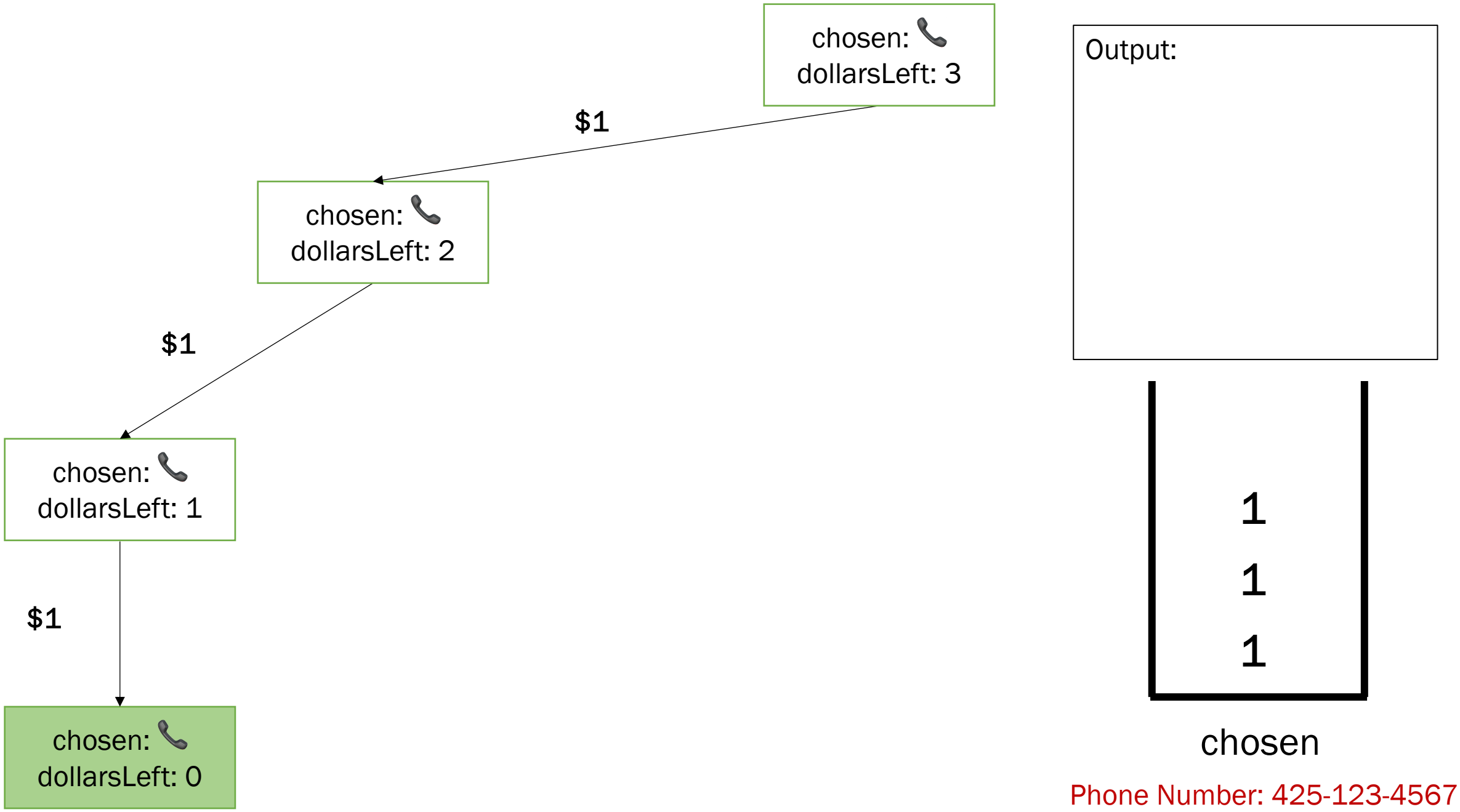
Phone Number: 425-123-4567

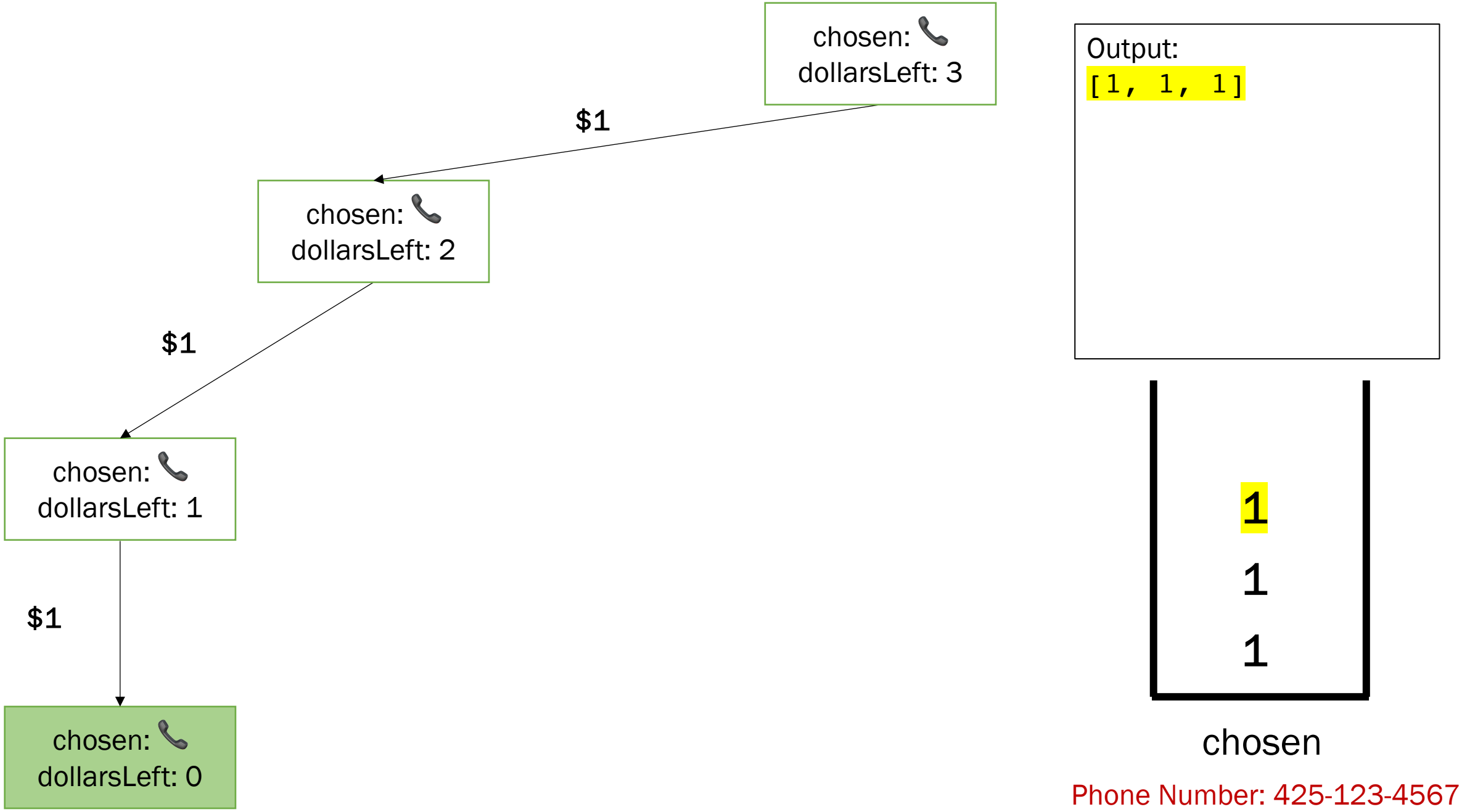

chosen: 📞
dollarsLeft: 3
$1
chosen: 📞
dollarsLeft: 2
$1
chosen: 📞
dollarsLeft: 1
Output:
1
1
chosen
Phone Number: 425-123-4567

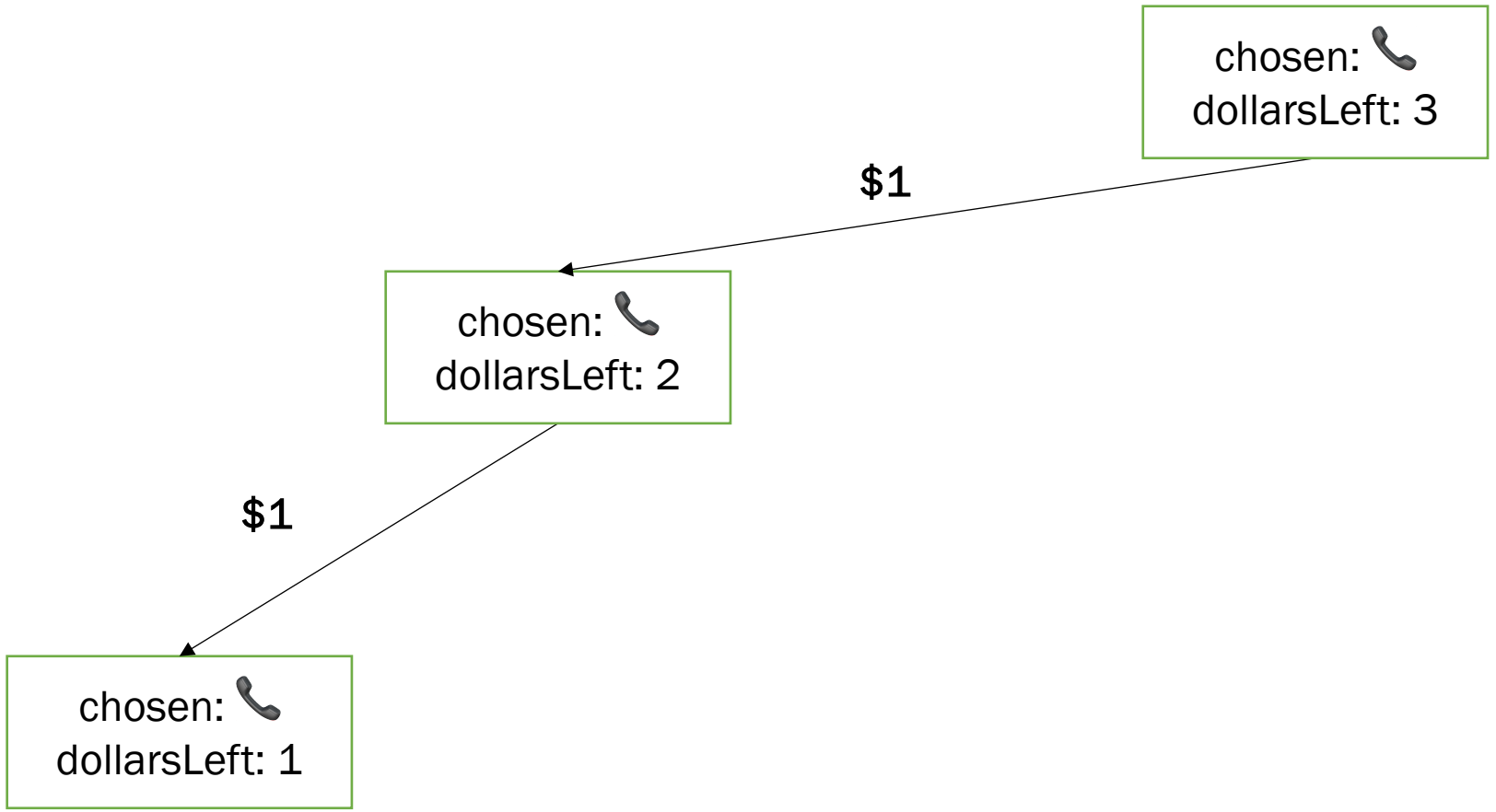


Phone Number: 425-123-4567


chosen: 📞
dollarsLeft: 3
$1
chosen: 📞
dollarsLeft: 2
$1
chosen: 📞
dollarsLeft: 1
$1
chosen: 📞
dollarsLeft: 0
Output:
1
1
1
chosen
Phone Number: 425-123-4567

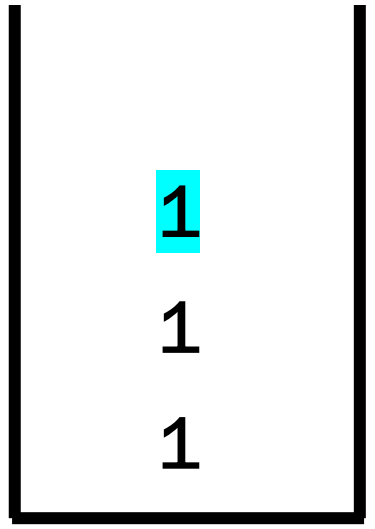

chosen: 📞
dollarsLeft: 3
$1
chosen: 📞
dollarsLeft: 2
$1
chosen: 📞
dollarsLeft: 1
$1
chosen: 📞
dollarsLeft: 0
Output:
1
1
1
chosen
Phone Number: 425-123-4567


chosen:
dollarsLeft: 3
$1
chosen:
dollarsLeft: 2
$1
chosen:
dollarsLeft: 1
$1
chosen:
dollarsLeft: 0
Output:
[1, 1, 1]
1
1
1
chosen
Phone Number: 425-123-4567



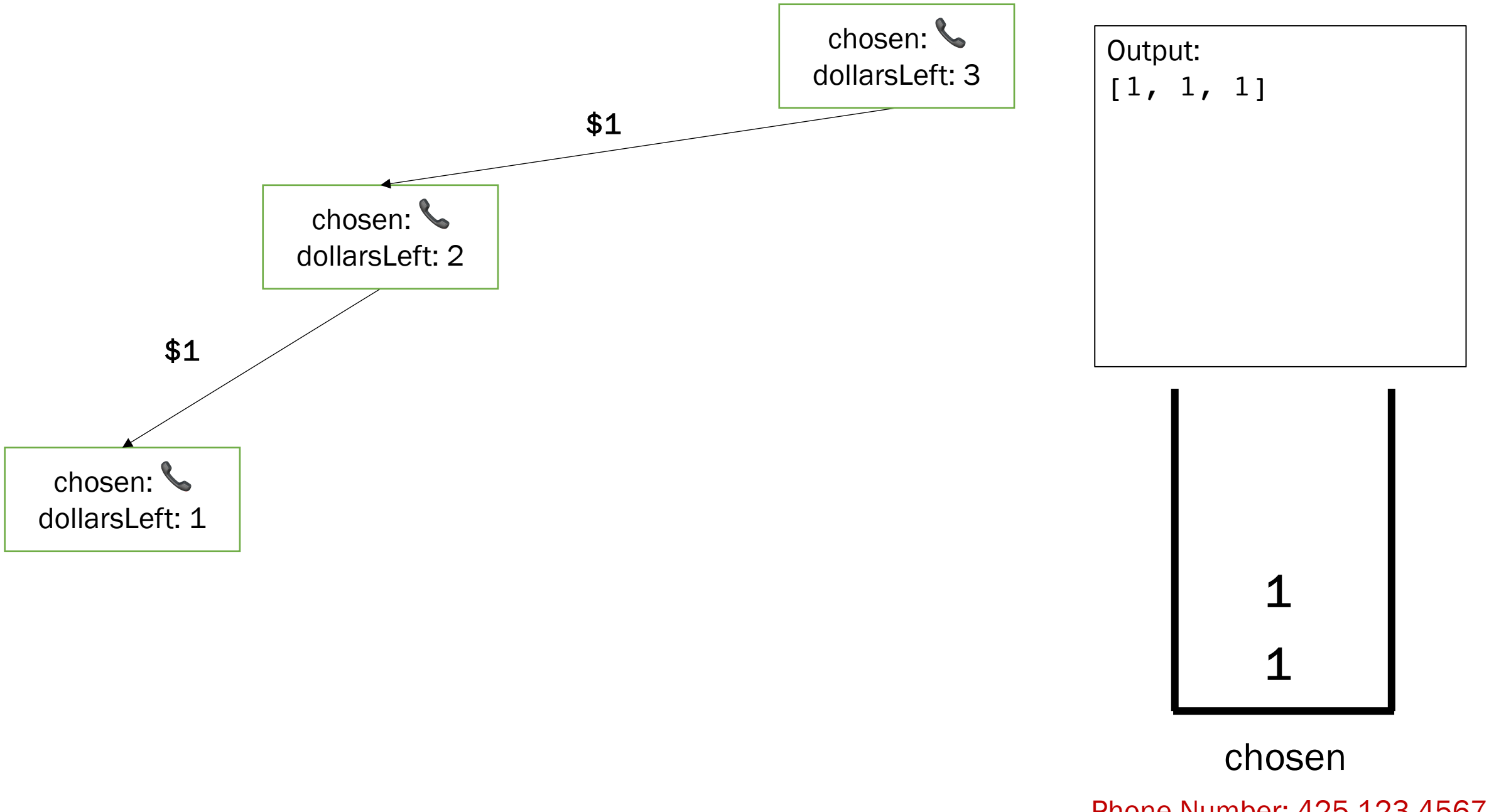
Output:

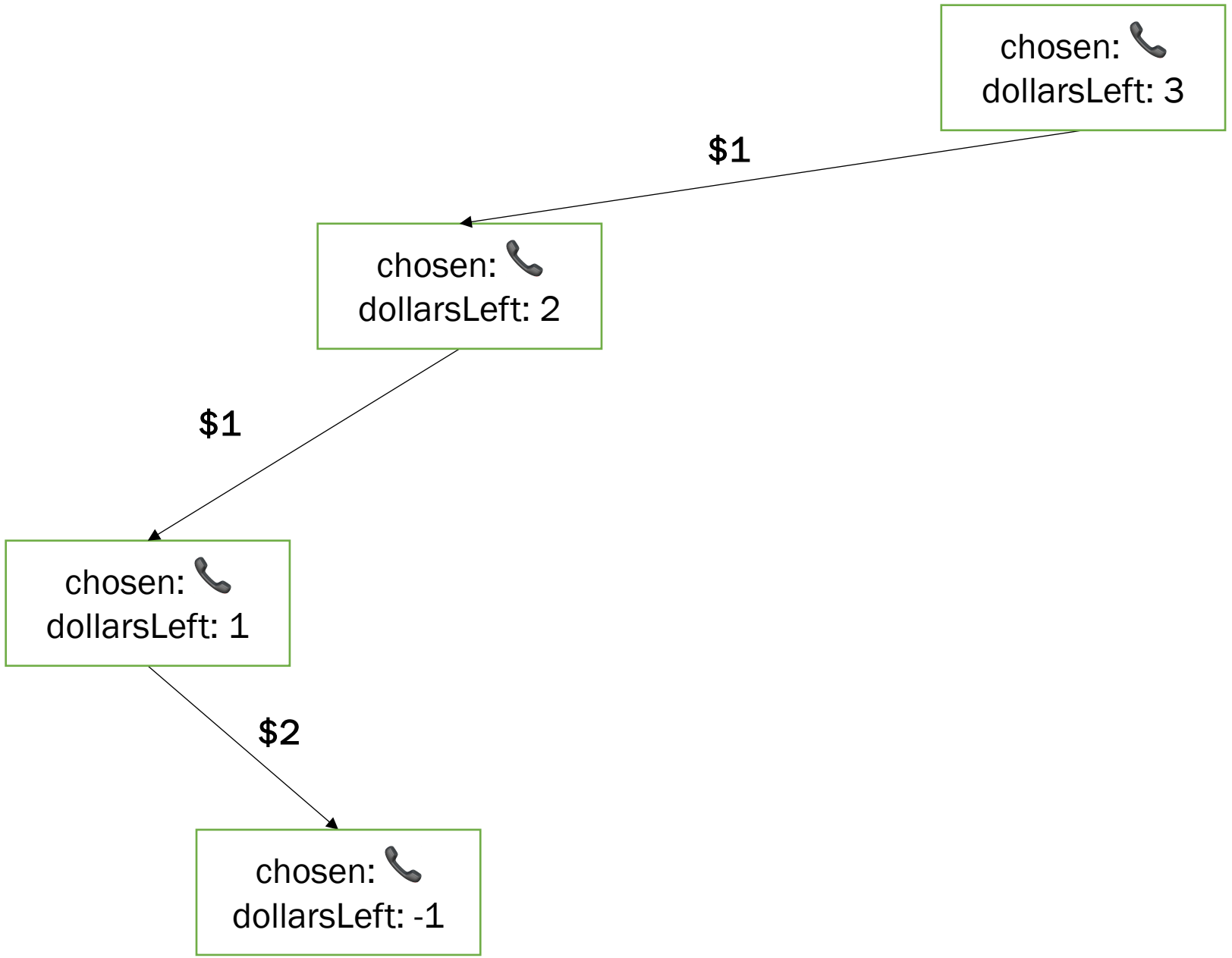
```
[1, 1, 1]
```



chosen

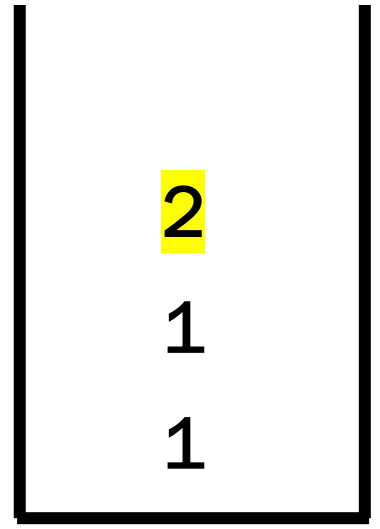
Phone Number: 425-123-4567


chosen: 📞
dollarsLeft: 3
$1
chosen: 📞
dollarsLeft: 2
$1
chosen: 📞
dollarsLeft: 1
Output:
[1, 1, 1]
1
1
chosen
Phone Number: 425-123-4567



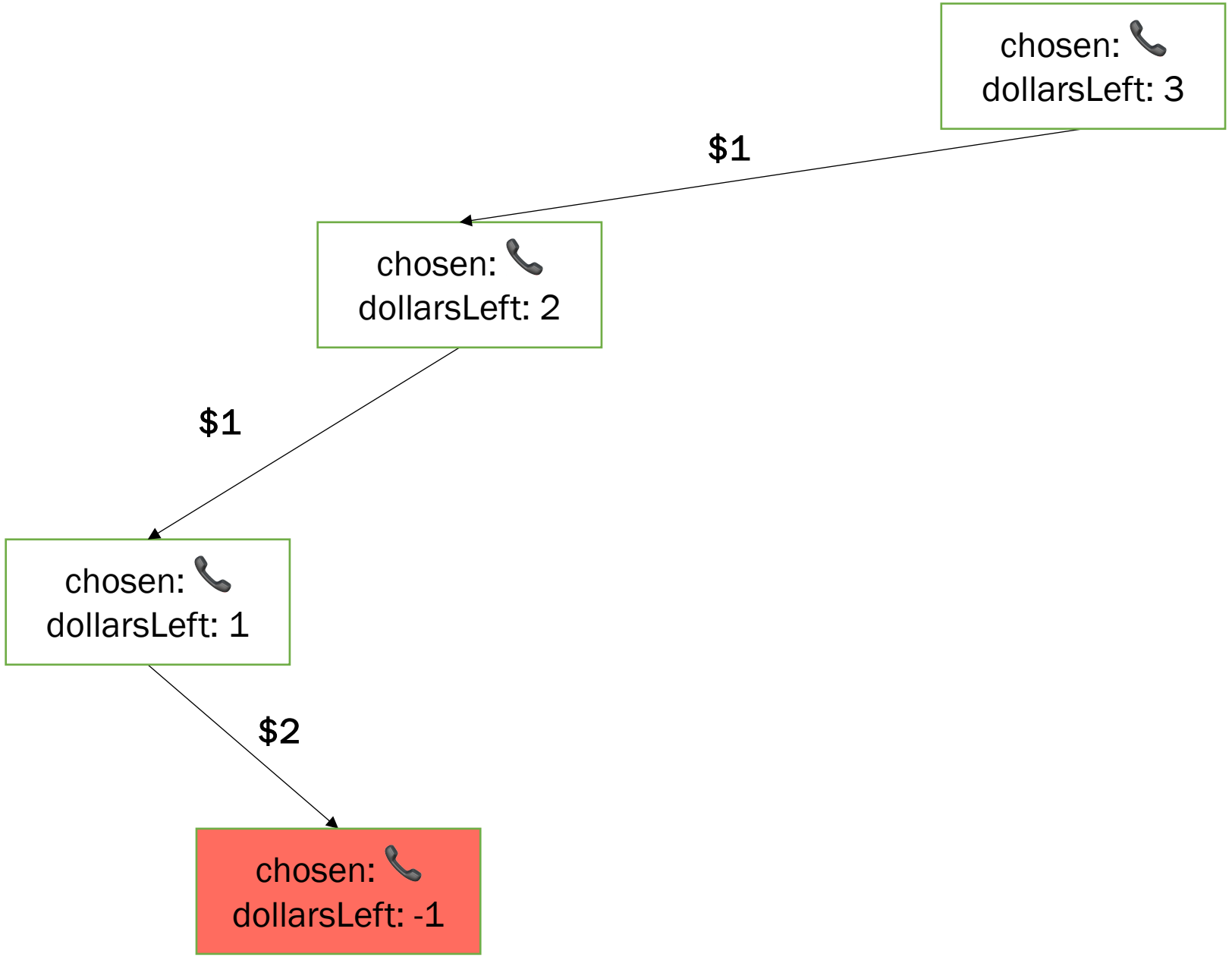
Output:

```
[1, 1, 1]
```



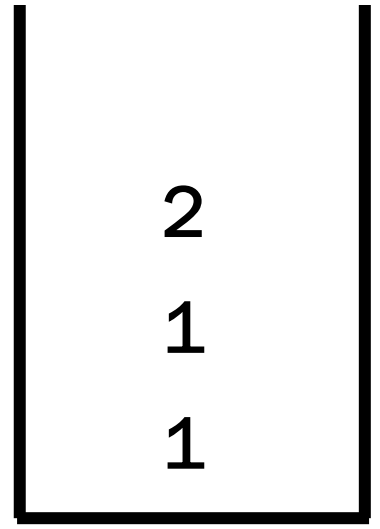
chosen

Phone Number: 425-123-4567



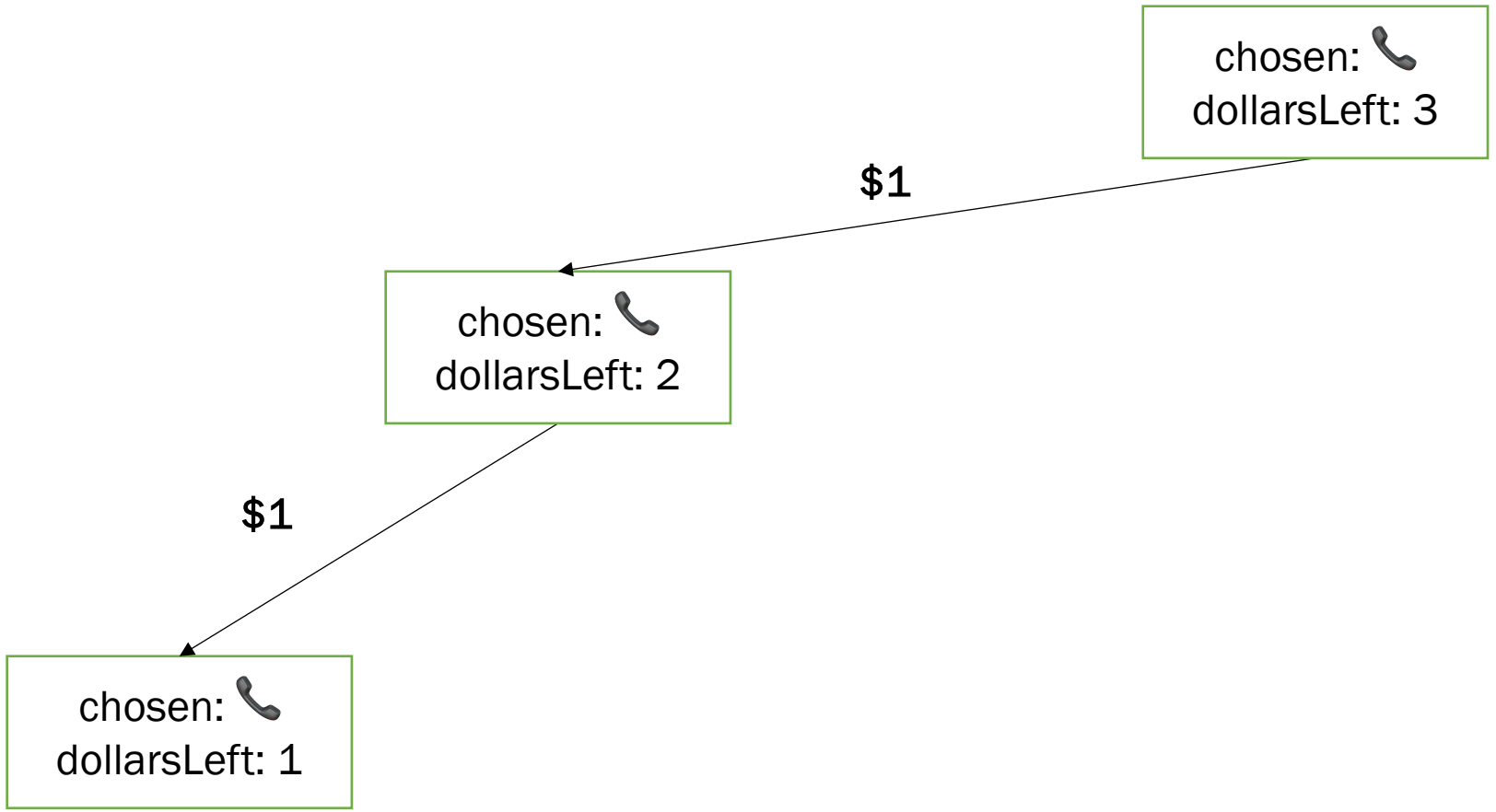
Output:

```
[1, 1, 1]
```



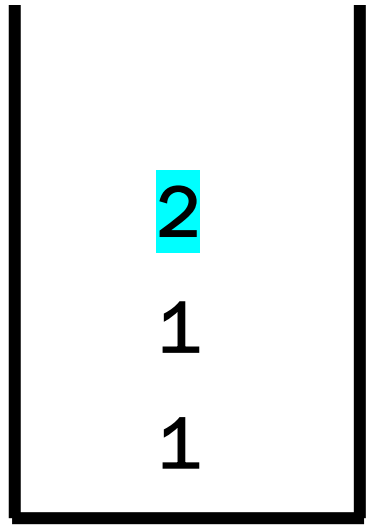
chosen

Phone Number: 425-123-4567



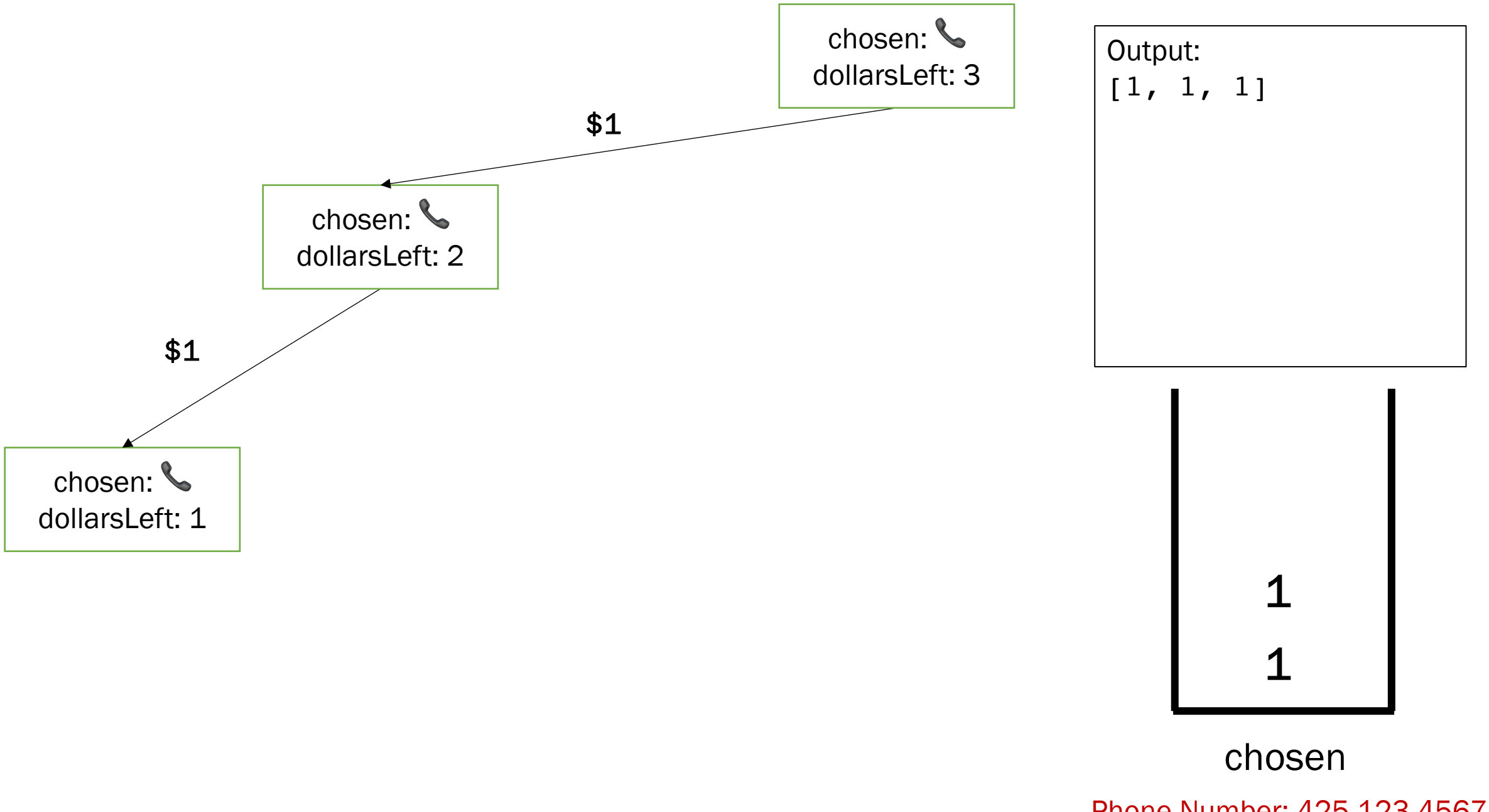
Output:

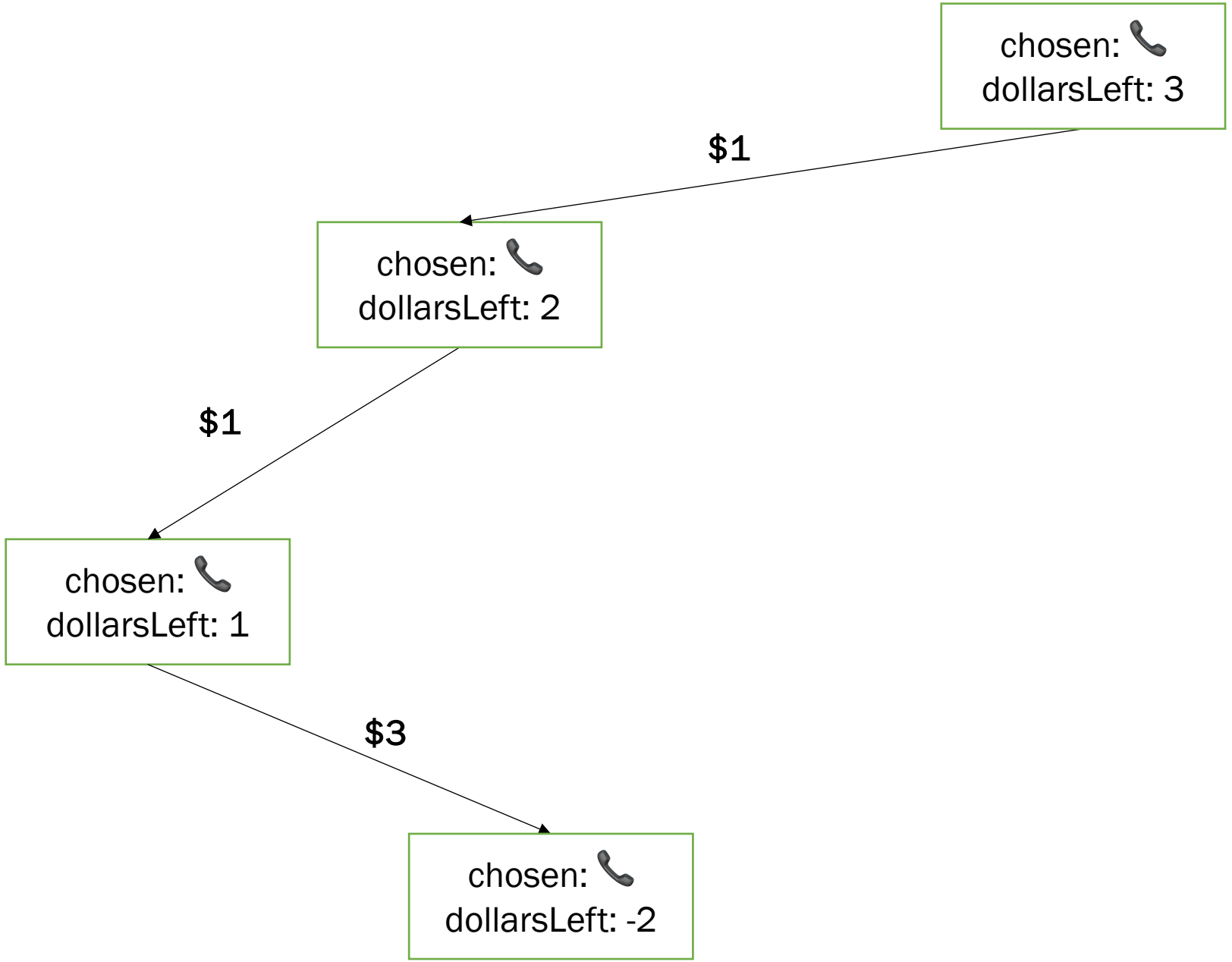
```
[1, 1, 1]
```



chosen

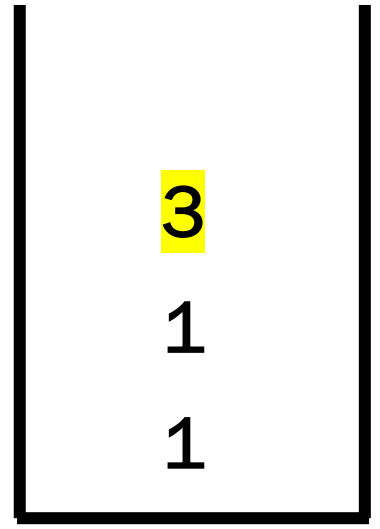
Phone Number: 425-123-4567


chosen: 📞
dollarsLeft: 3
$1
chosen: 📞
dollarsLeft: 2
$1
chosen: 📞
dollarsLeft: 1
Output:
[1, 1, 1]
1
1
chosen
Phone Number: 425-123-4567



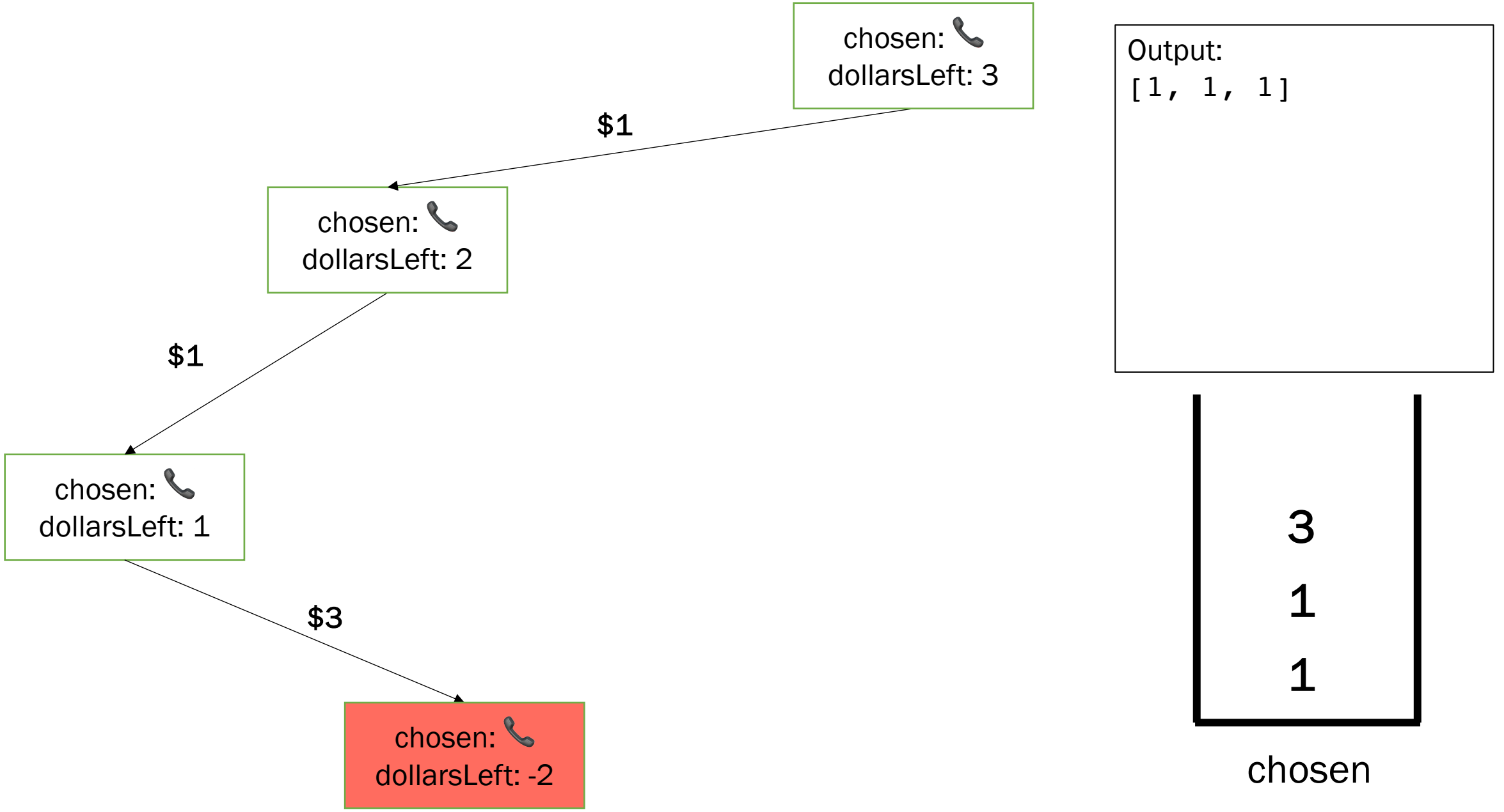
Output:

```
[1, 1, 1]
```

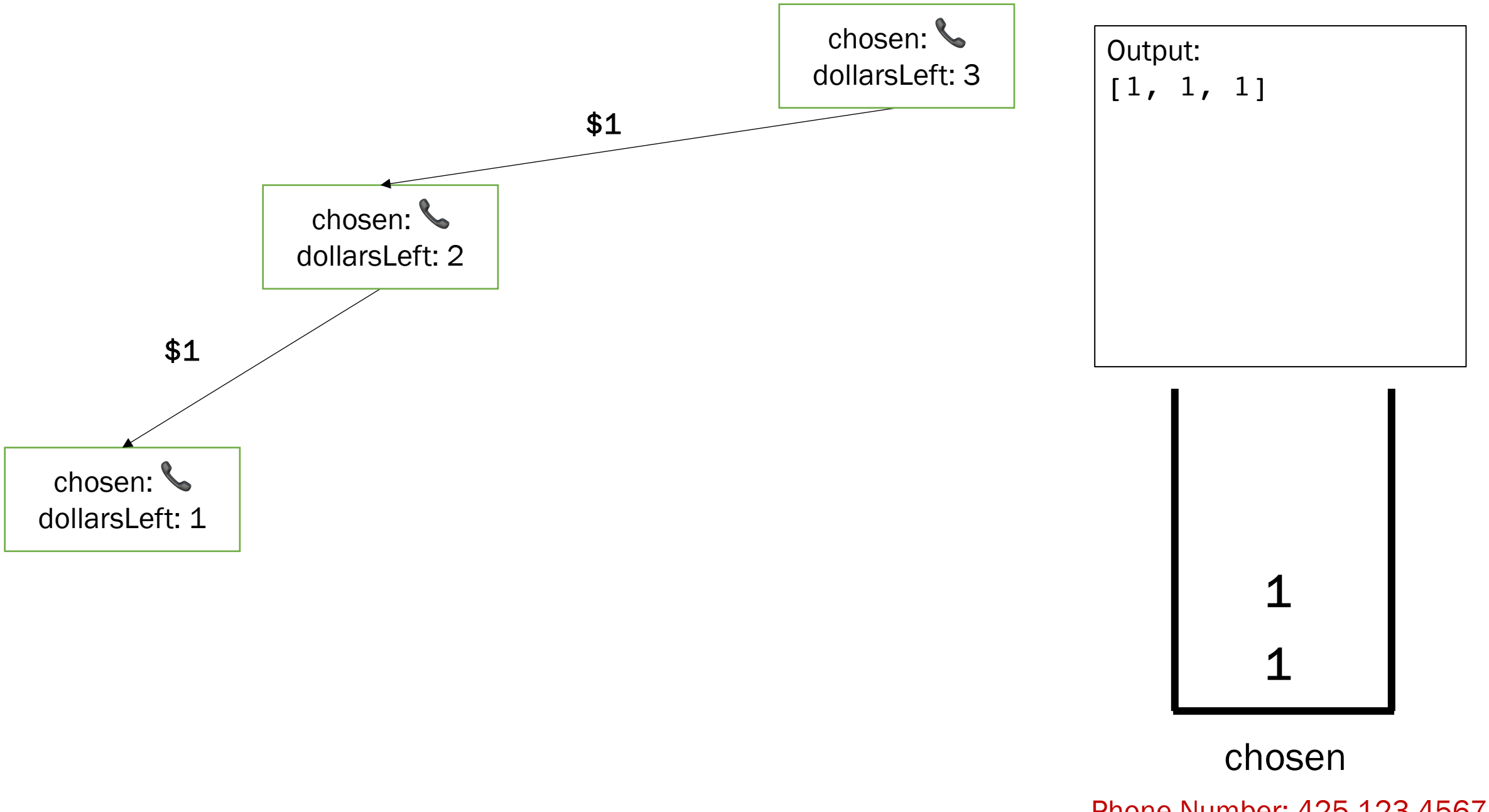


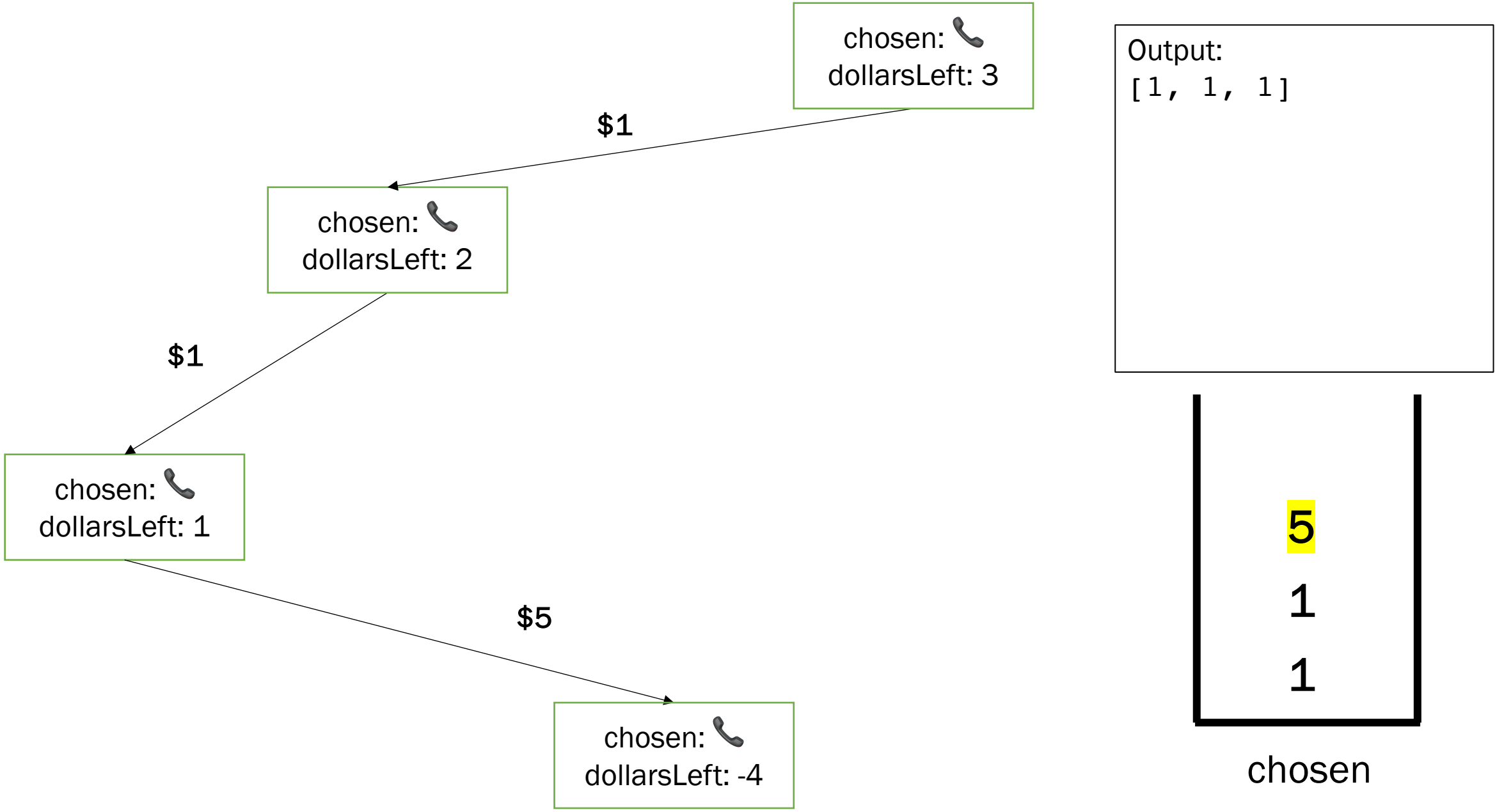
chosen

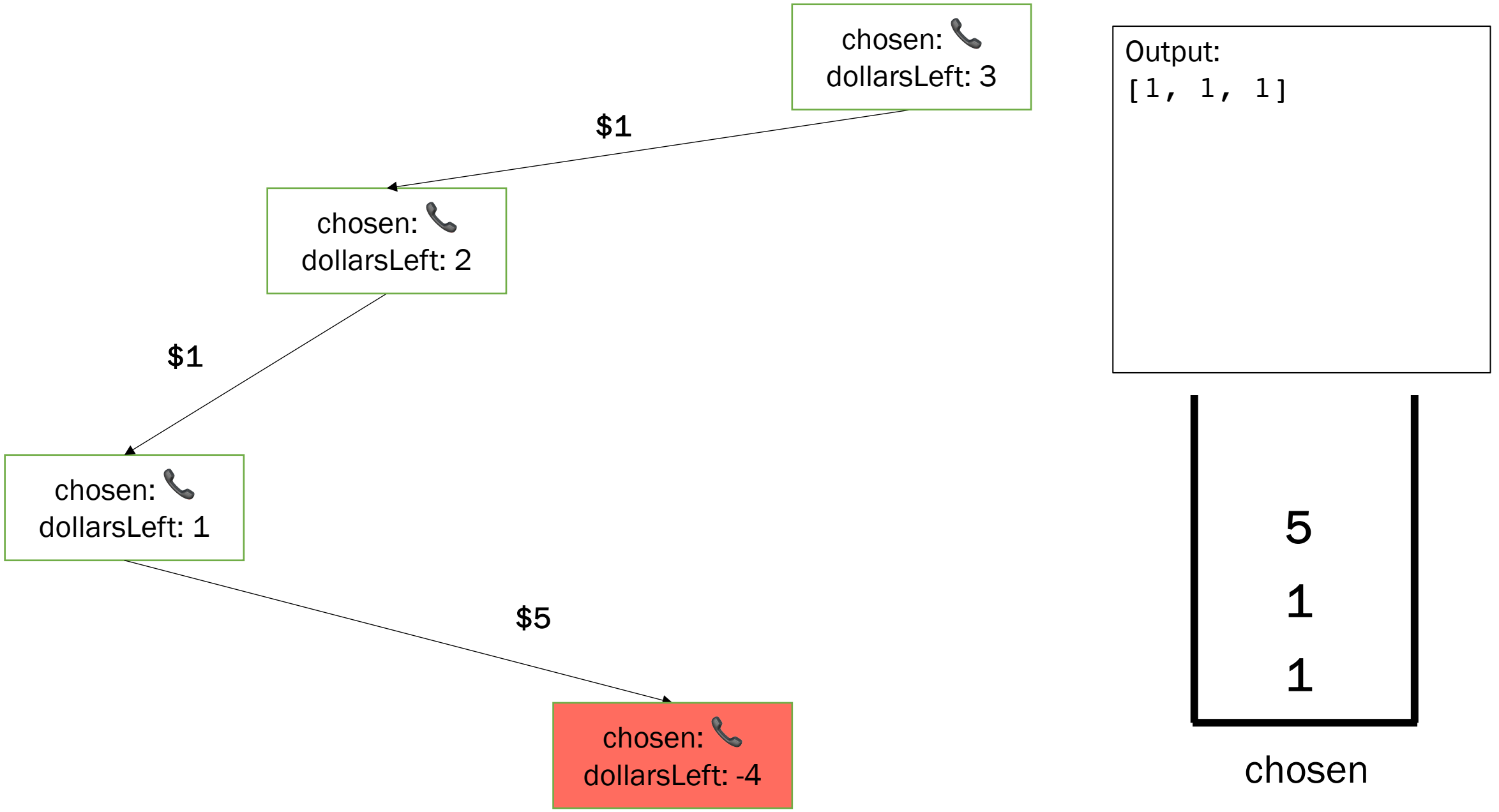
Phone Number: 425-123-4567

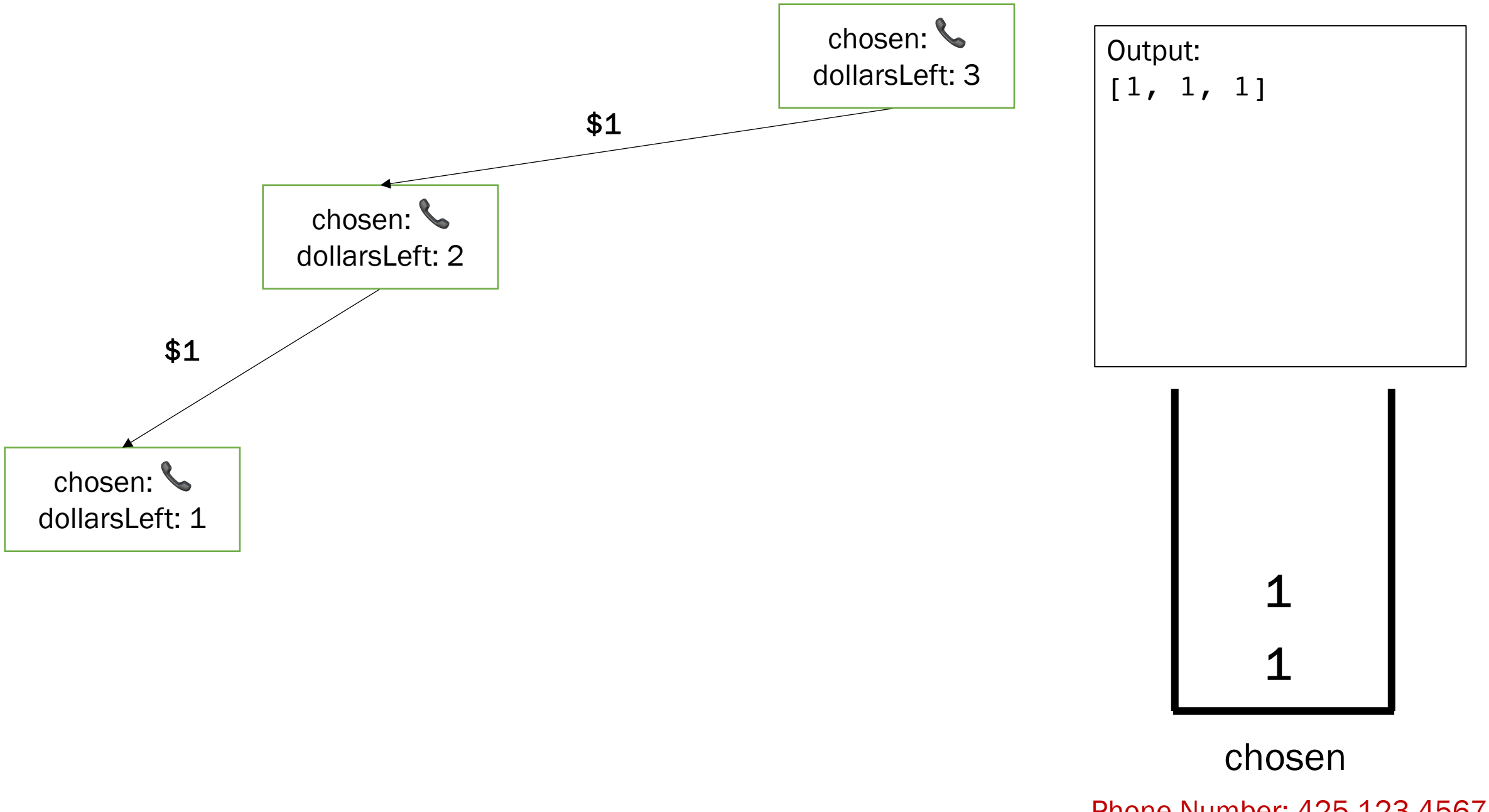


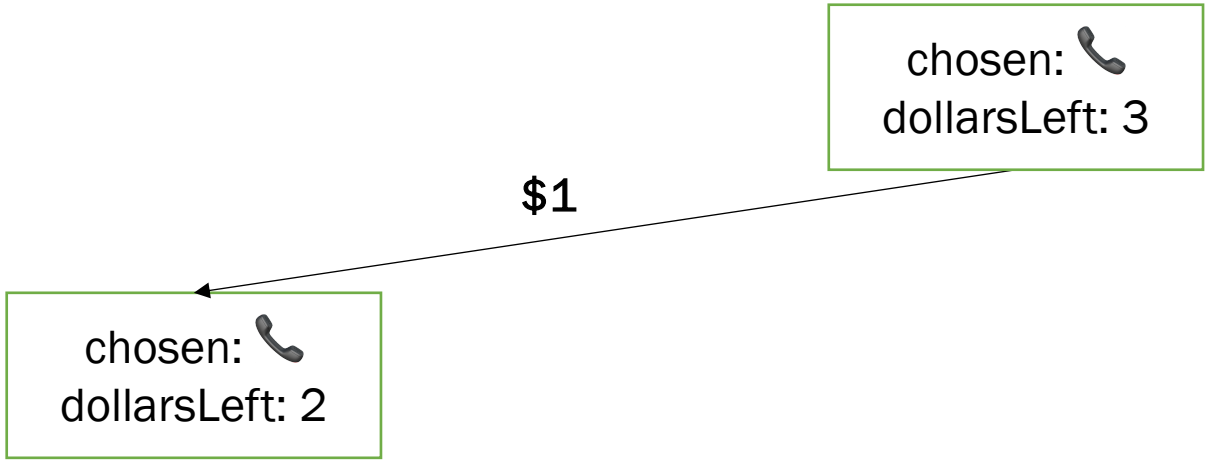
Phone Number: 425-123-4567


chosen: 📞
dollarsLeft: 3
$1
chosen: 📞
dollarsLeft: 2
$1
chosen: 📞
dollarsLeft: 1
Output:
[1, 1, 1]
1
1
chosen
Phone Number: 425-123-4567

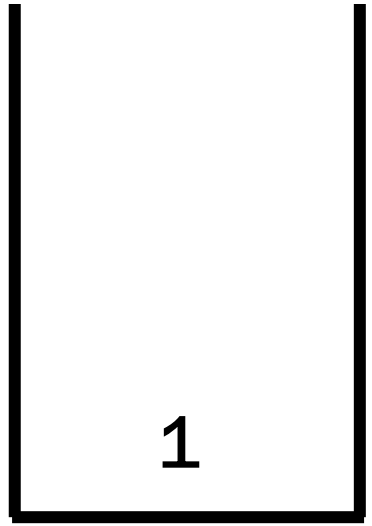

chosen:
dollarsLeft: 3
$1
chosen:
dollarsLeft: 2
$1
chosen:
dollarsLeft: 1
$5
chosen:
dollarsLeft: -4
Output:
[1, 1, 1]
5
1
1
chosen
Phone Number: 425-123-4567


chosen:
dollarsLeft: 3
$1
chosen:
dollarsLeft: 2
$1
chosen:
dollarsLeft: 1
$5
chosen:
dollarsLeft: -4
Output:
[1, 1, 1]
5
1
1
chosen
Phone Number: 425-123-4567


chosen: 📞
dollarsLeft: 3
$1
chosen: 📞
dollarsLeft: 2
$1
chosen: 📞
dollarsLeft: 1
Output:
[1, 1, 1]
1
1
chosen
Phone Number: 425-123-4567

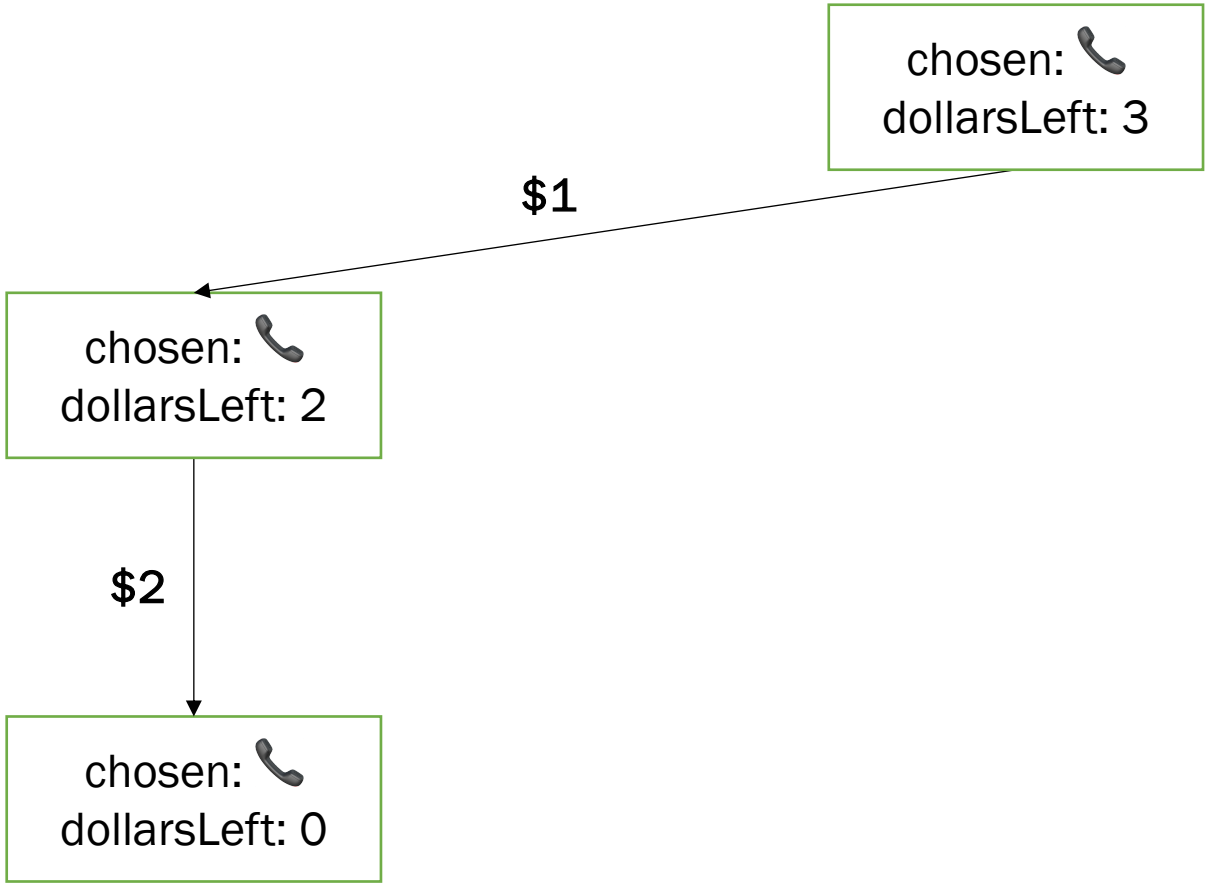


Output:
`[1, 1, 1]`



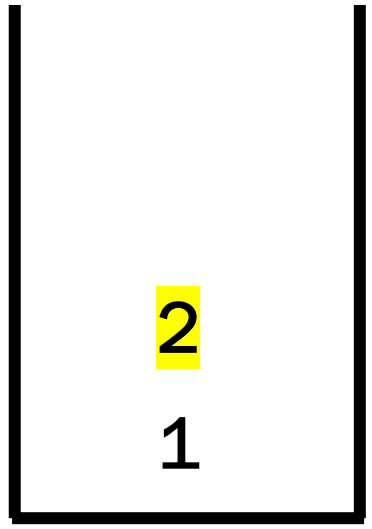
chosen

Phone Number: 425-123-4567



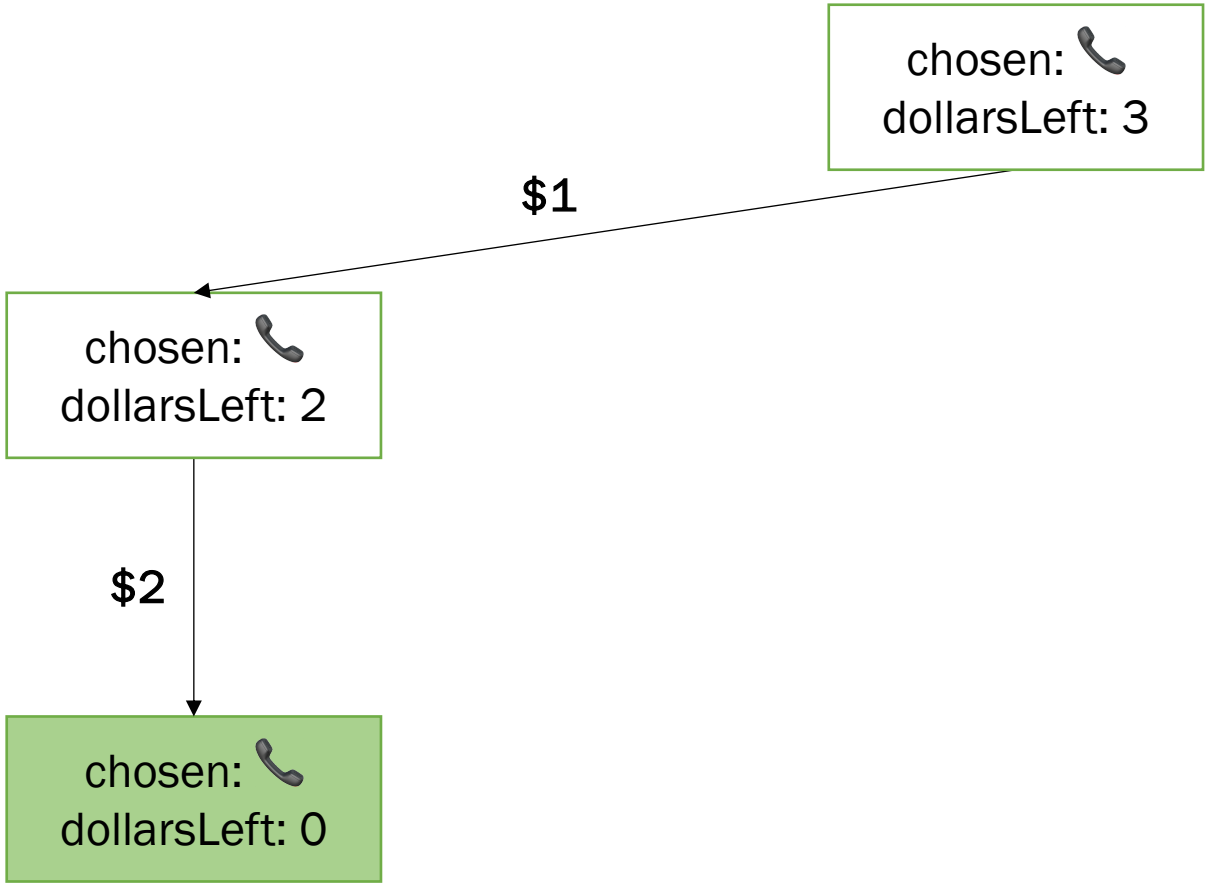
Output:

```
[1, 1, 1]
```



chosen

Phone Number: 425-123-4567



Output:

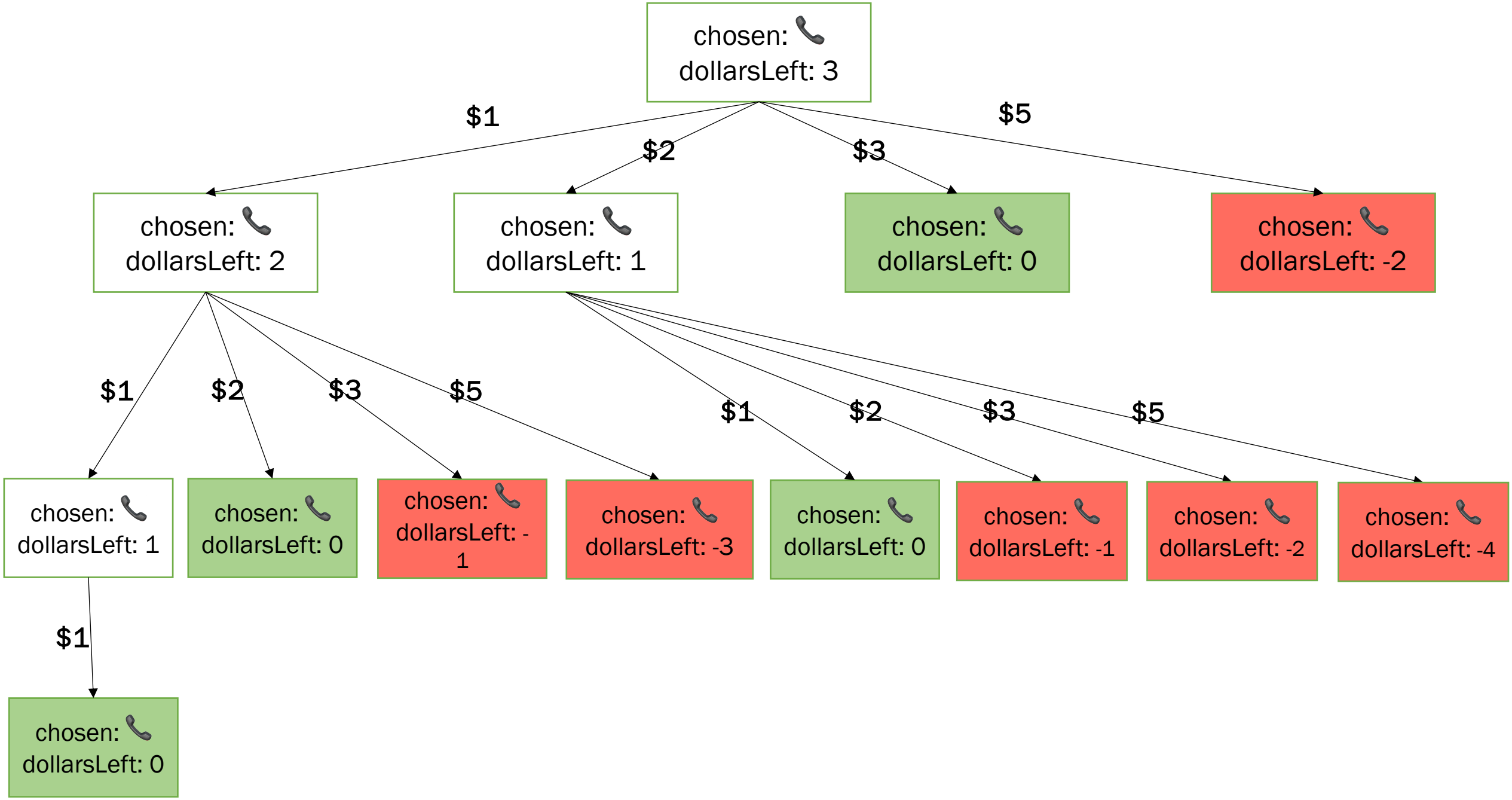
```
[1, 1, 1]
[1, 2]
```

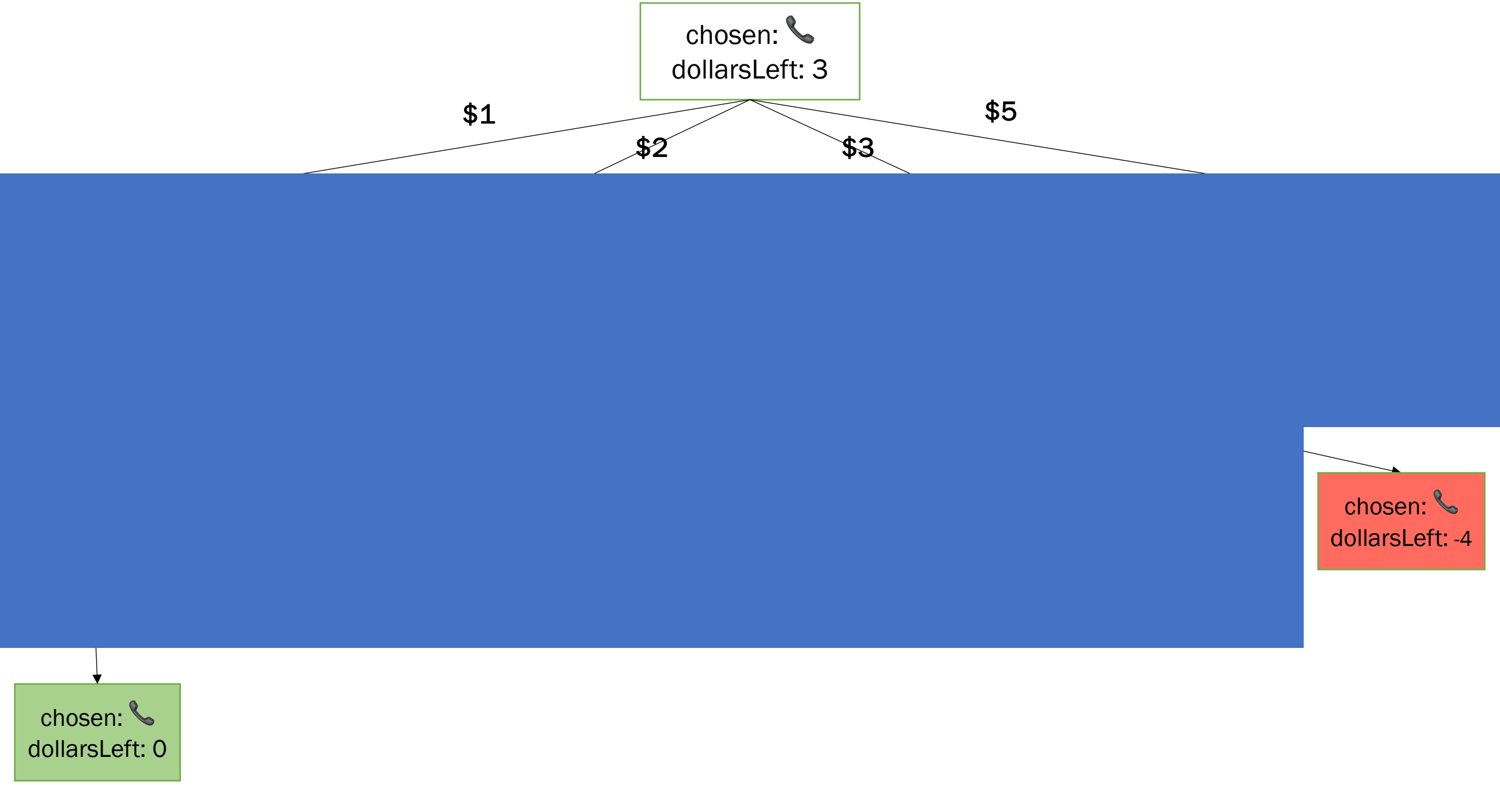
2

1

chosen

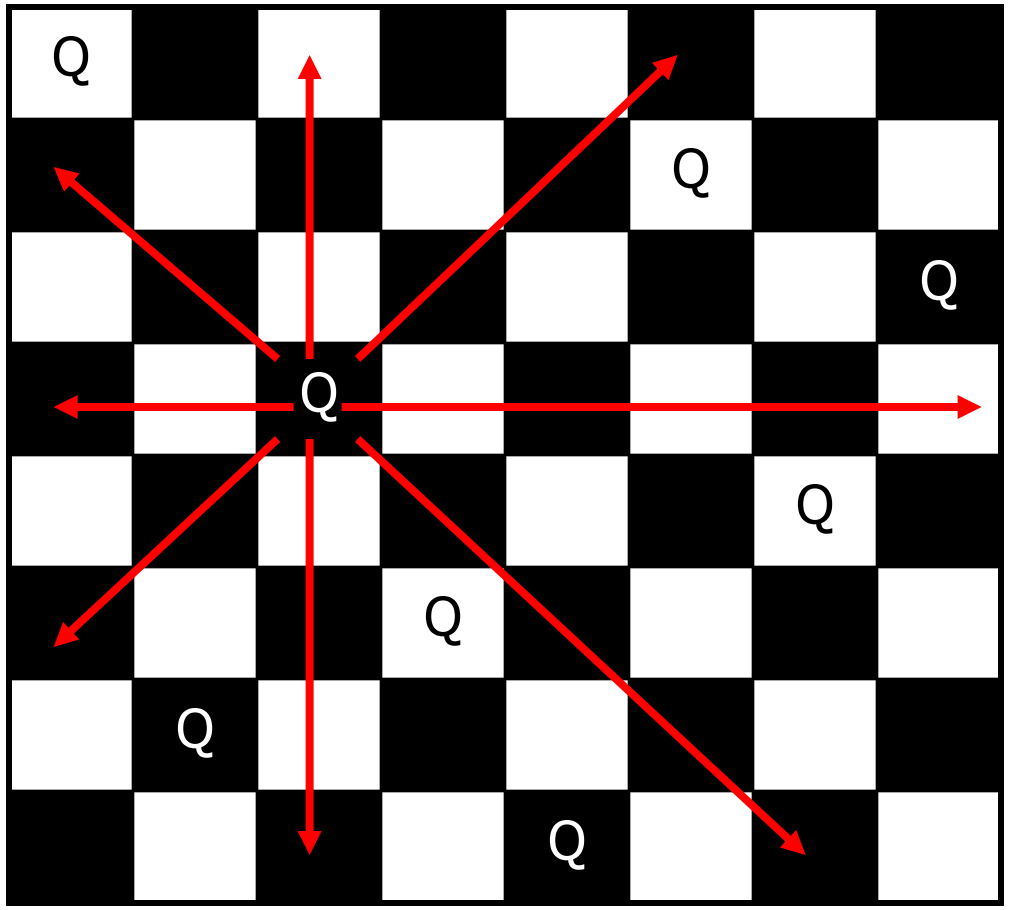
Phone Number: 425-123-4567


chosen:
dollarsLeft: 3
$1
$2
$3
$5
chosen:
dollarsLeft: 2
chosen:
dollarsLeft: 1
chosen:
dollarsLeft: 0
chosen:
dollarsLeft: -2
$1
$2
$3
$5
$1
$2
$3
$5
chosen:
dollarsLeft: 1
chosen:
dollarsLeft: 0
chosen:
dollarsLeft: -1
chosen:
dollarsLeft: -3
chosen:
dollarsLeft: 0
chosen:
dollarsLeft: -1
chosen:
dollarsLeft: -2
chosen:
dollarsLeft: -4
$1
chosen:
dollarsLeft: 0


chosen: 📞
dollarsLeft: 3
$1
$2
$3
$5
chosen: 📞
dollarsLeft: -4
chosen: 📞
dollarsLeft: 0

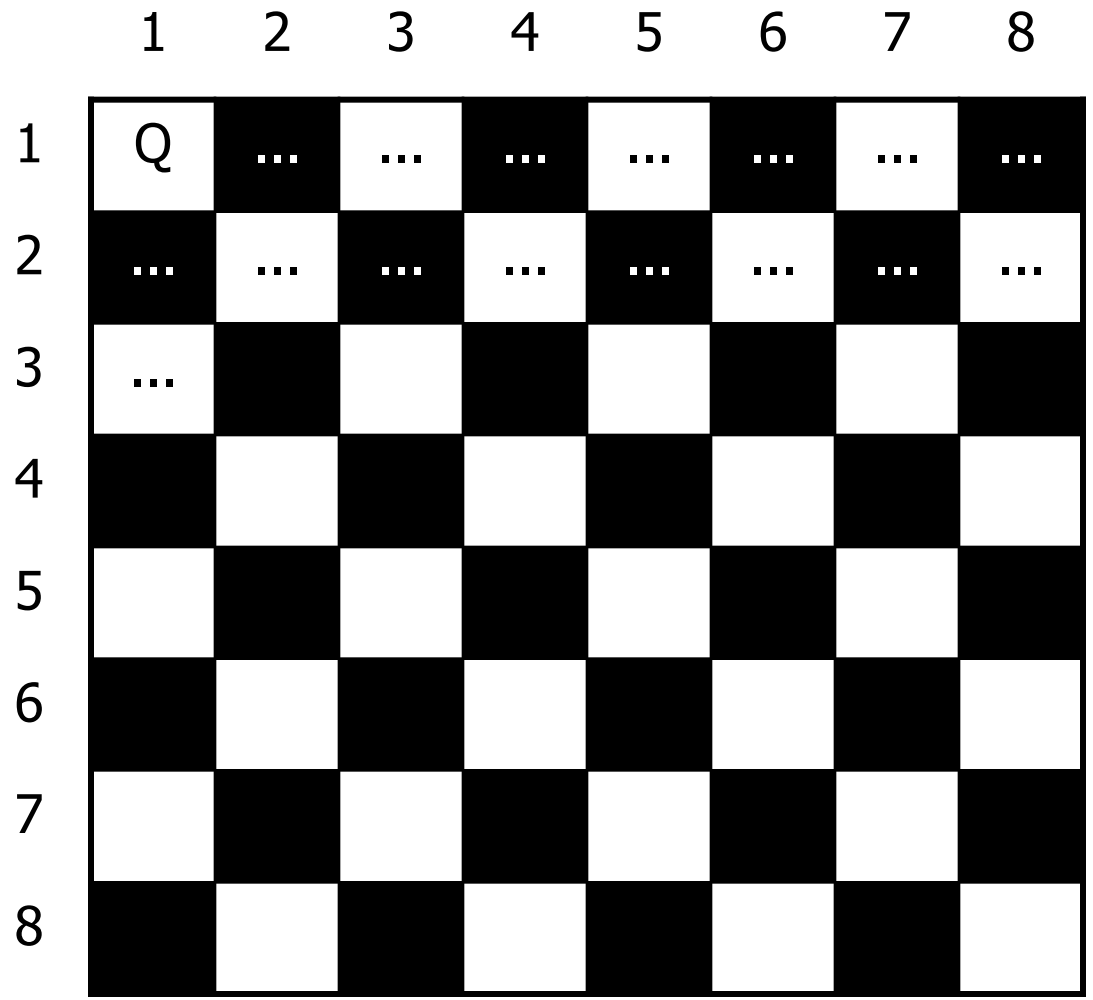
The "8 Queens" problem

- Consider the problem of trying to place 8 queens on a chess board such that no queen can attack another queen.
  - What are the "choices"?
  - How do we "make" or "un-make" a choice?
  - How do we know when to stop?



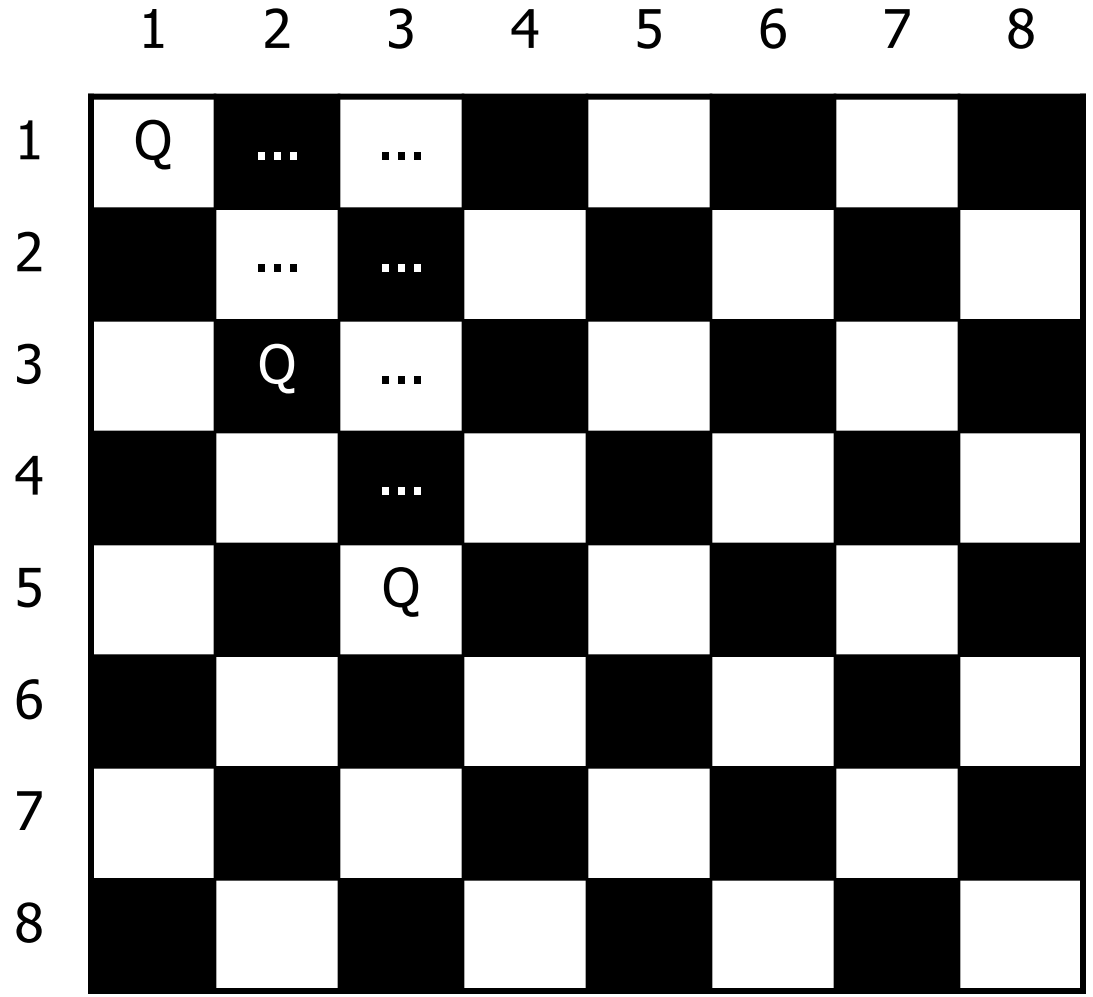
Naive algorithm

- for (each square on board):
  - Place a queen there.
  - Try to place the rest of the queens.
  - Un-place the queen.



Better algorithm idea

- Observation: In a working solution, exactly 1 queen must appear in each column.
  - Redefine a "choice" to be valid placement of a queen in a particular **column**.



`solve`

- Write a method `solve` that accepts a `Board` as a parameter and prints out all the ways to place `n` queens on it.

`Board` class