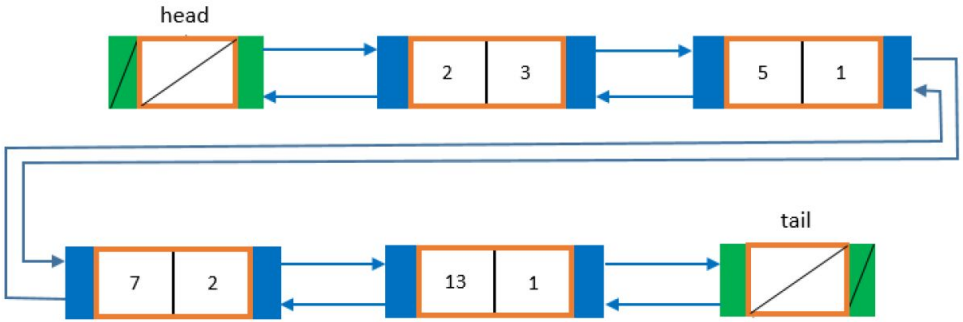
Week 8

Array Collections, Iterators

Project 3 is out!



Prime Factor List

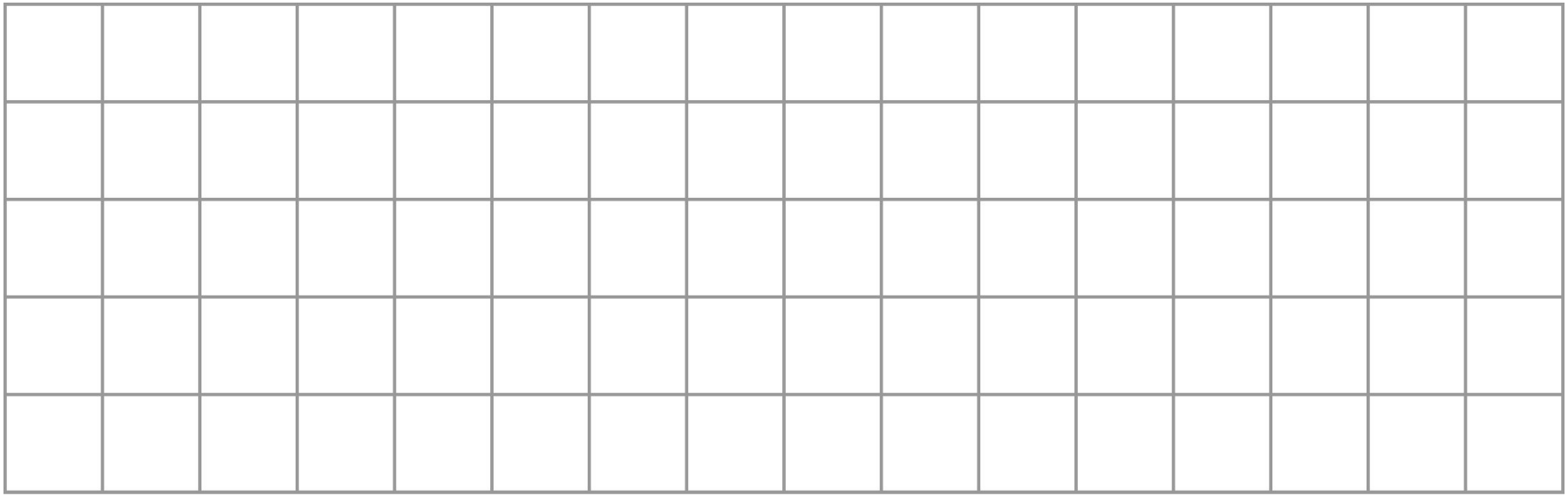
Applied the concepts of:

- Linked Lists
- Iterators
- Math

START EARLY! Only 20 submissions were submitted a day before the due date of project 2

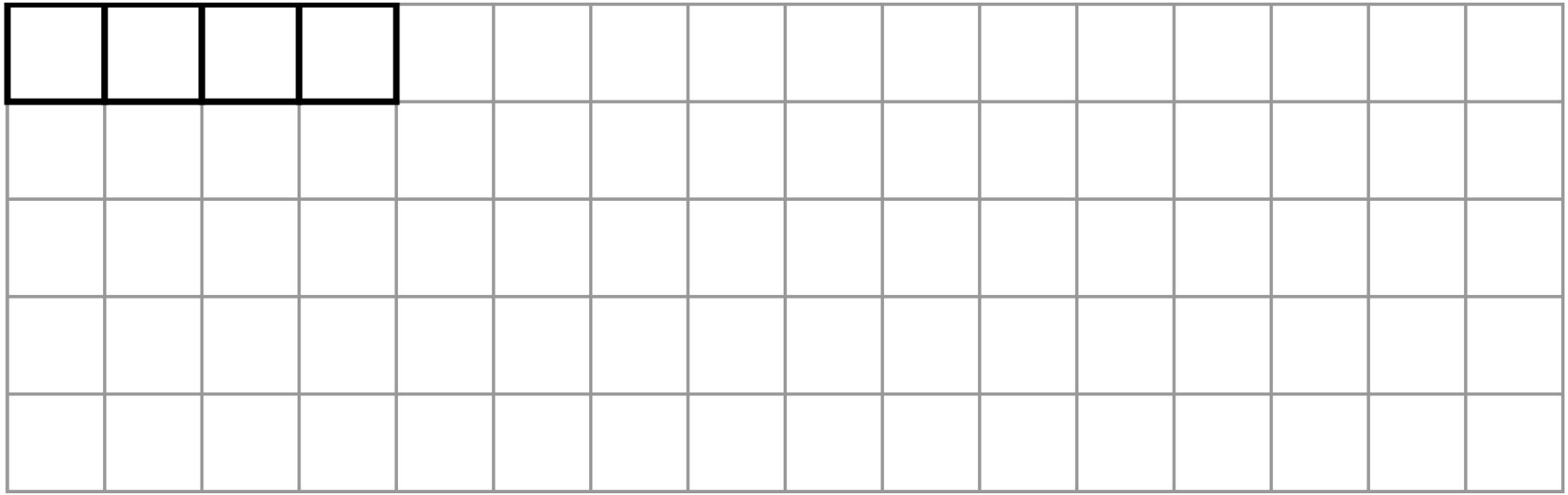
Array-based Collections

How do we store elements in an array?



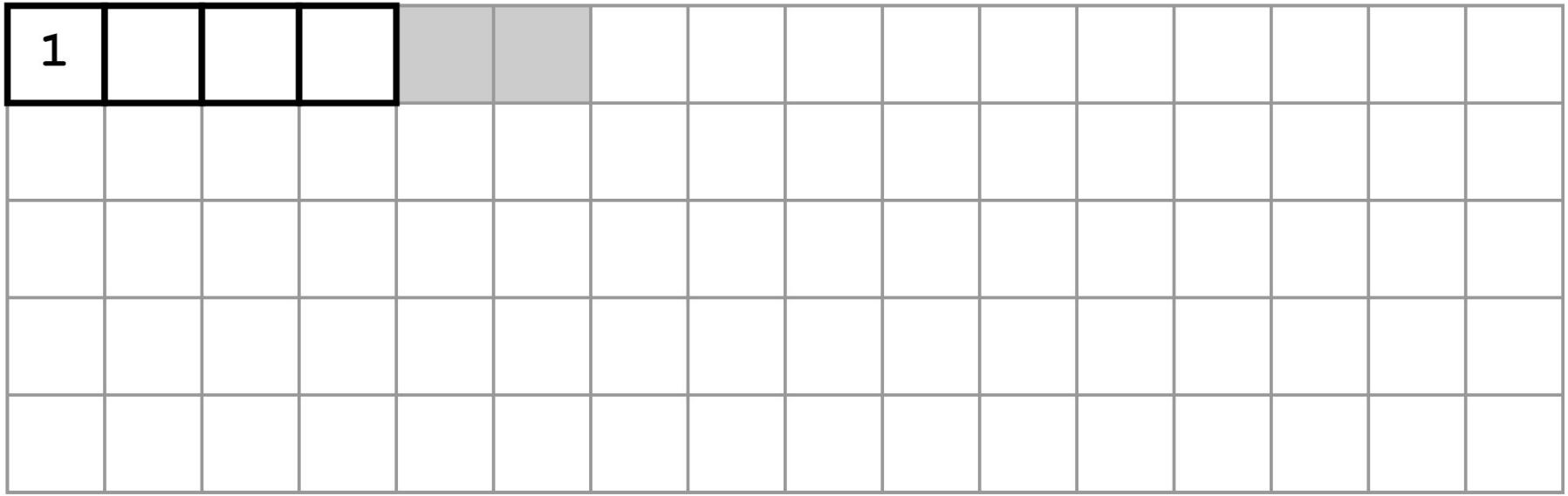
Program Memory

```
Integer[] arr = new Integer[4];
```



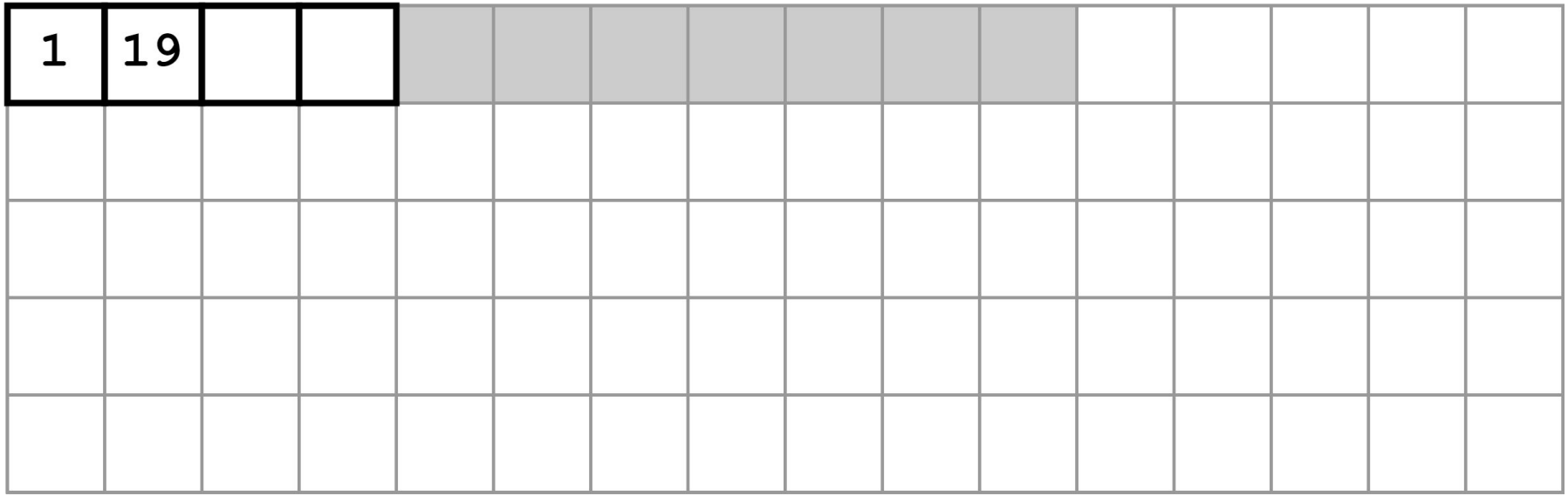
Program Memory

```
arr[0] = 1;
```



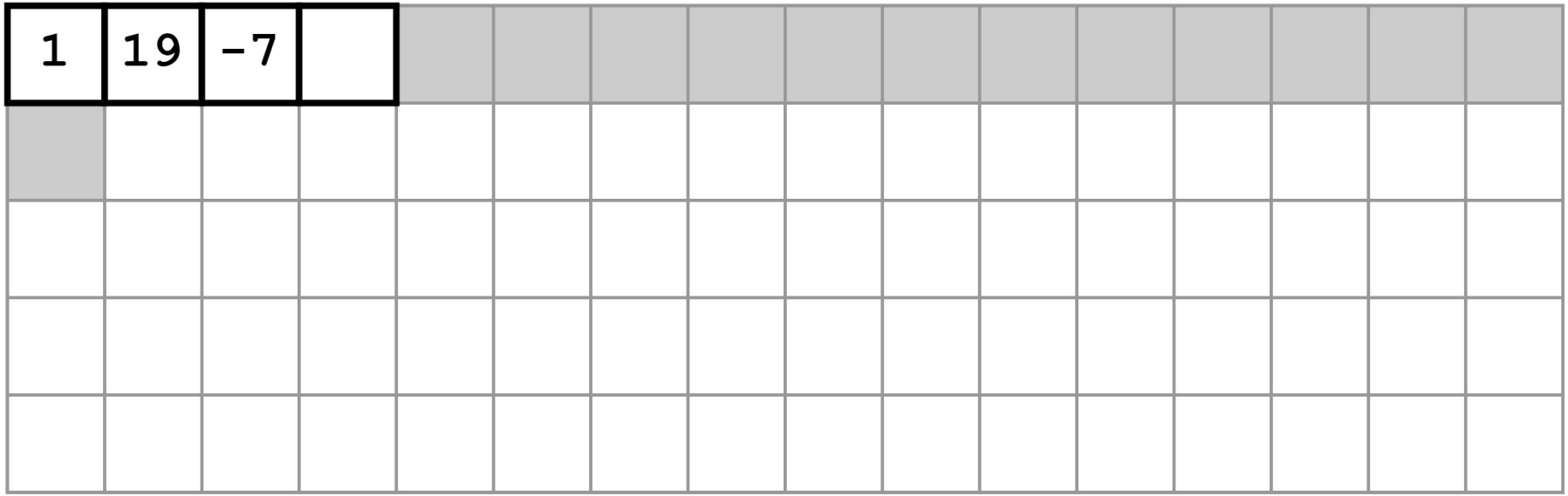
Program Memory

```
arr[1] = 19;
```



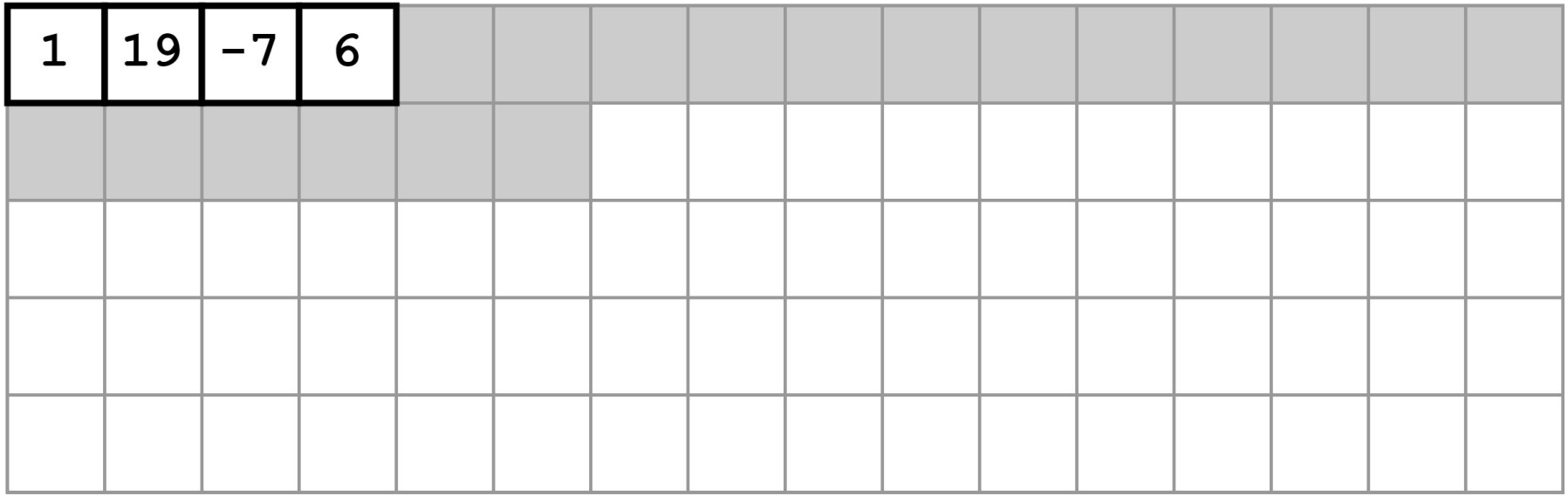
Program Memory

```
arr[1] = -7;
```



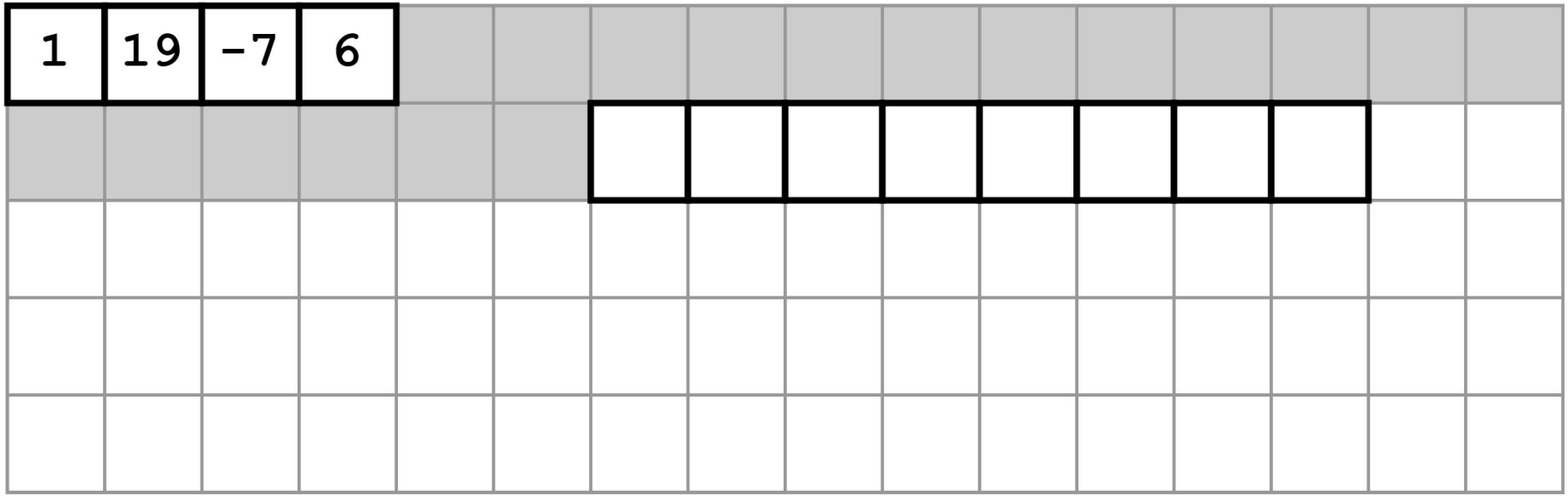
Program Memory

```
arr[1] = 6;
```



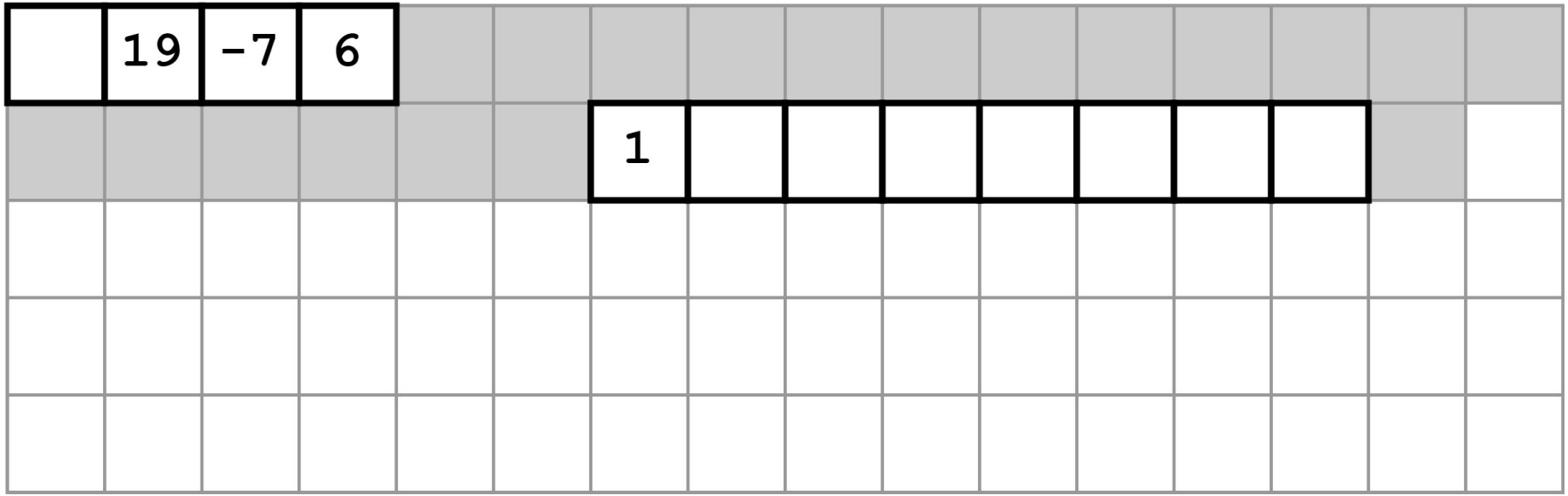
Program Memory

```
Integer[] temp = new Integer[8];
```

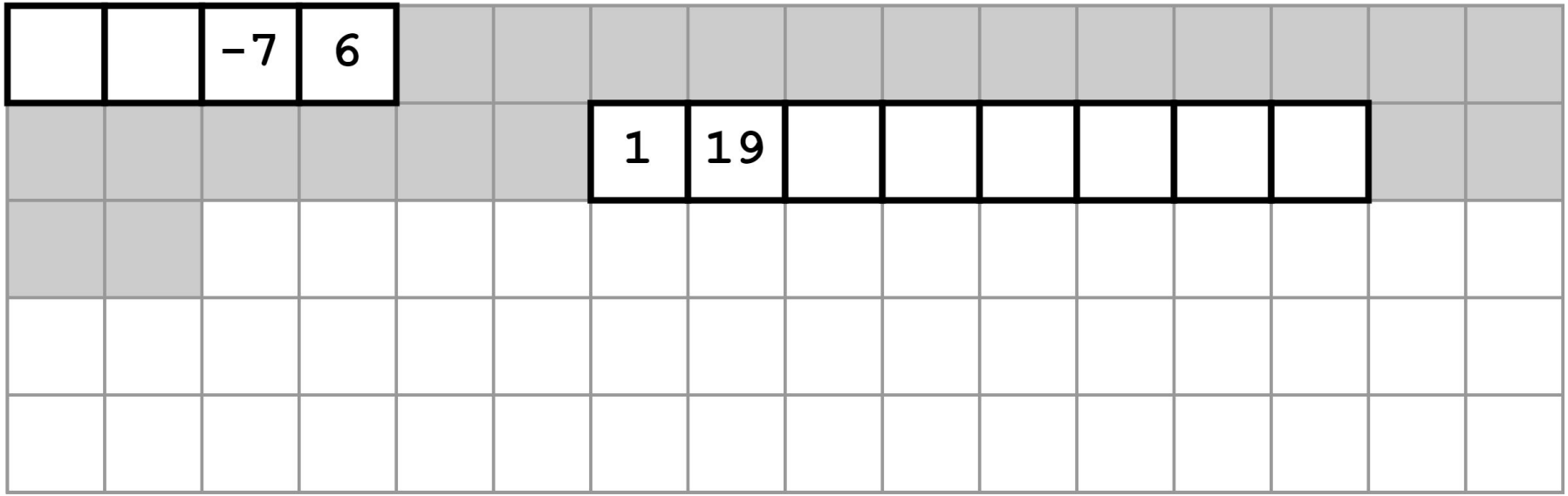


Program Memory

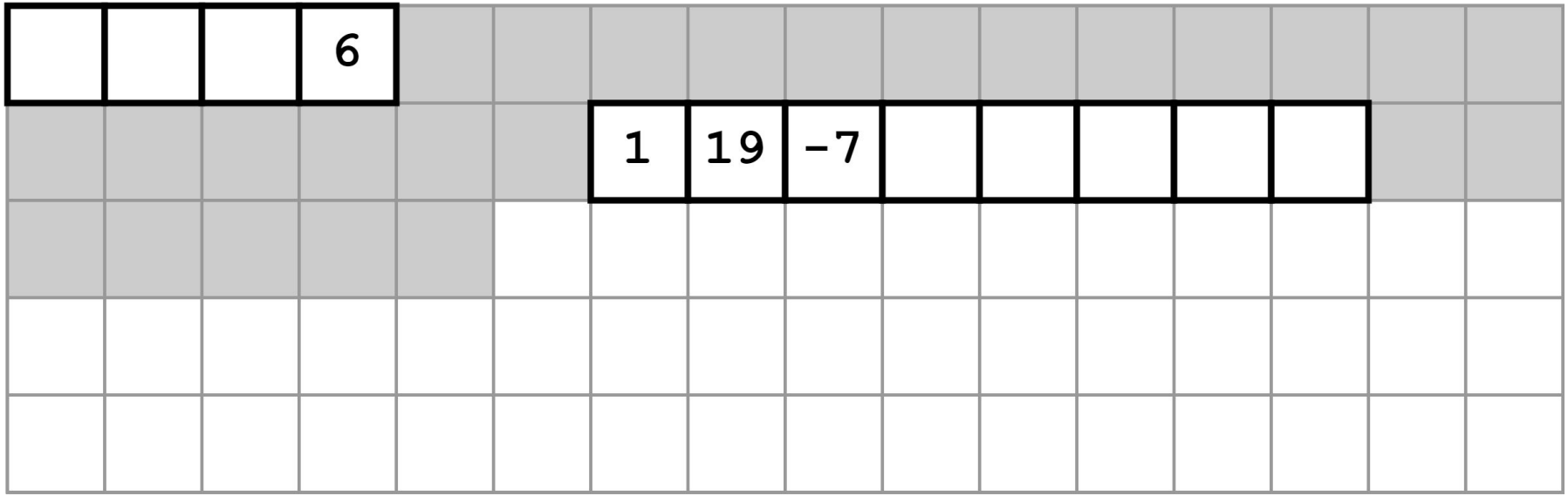
```
temp[0] = arr[0];
```



```
temp[1] = arr[1];
```

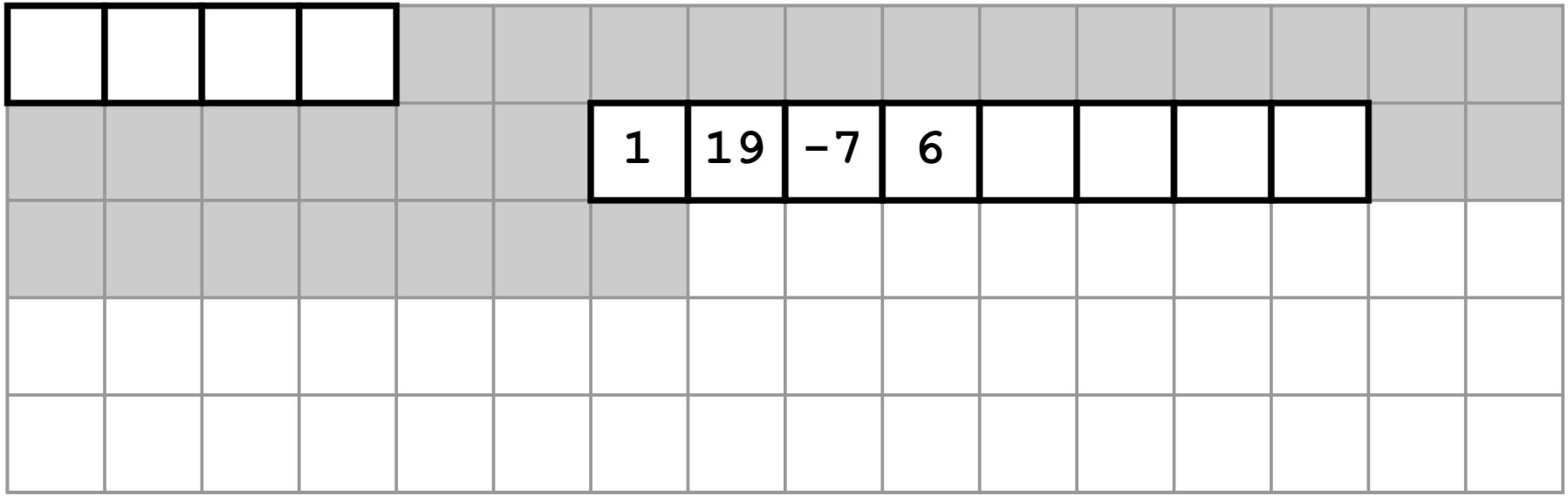


```
temp[2] = arr[2];
```



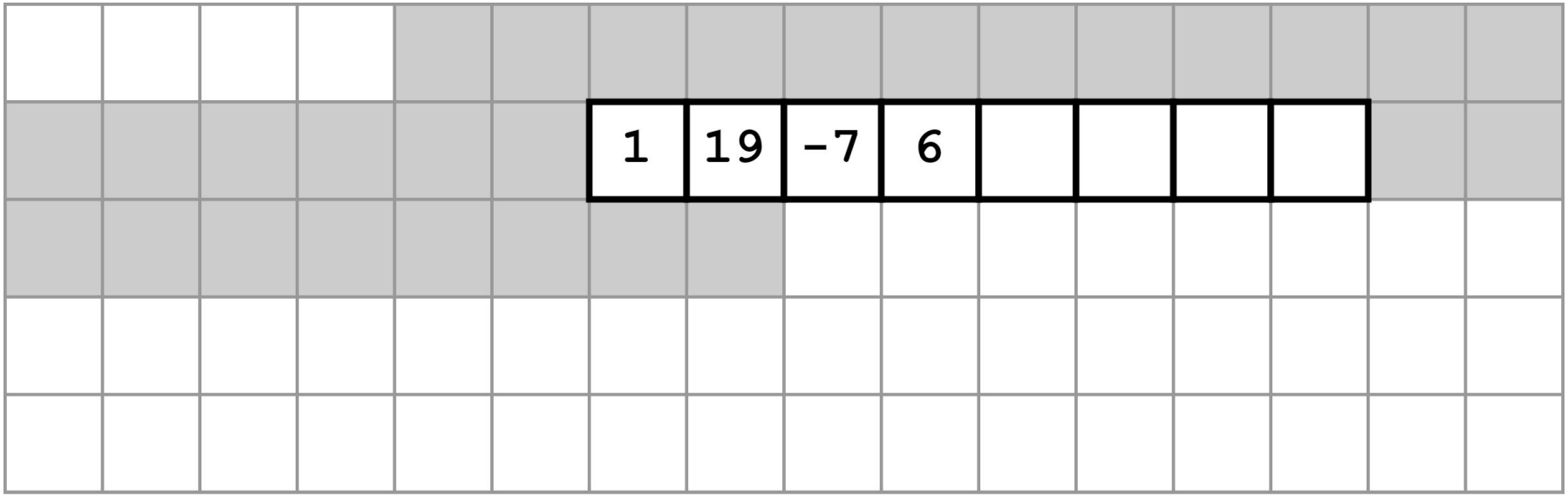
Program Memory

```
temp[3] = arr[3];
```



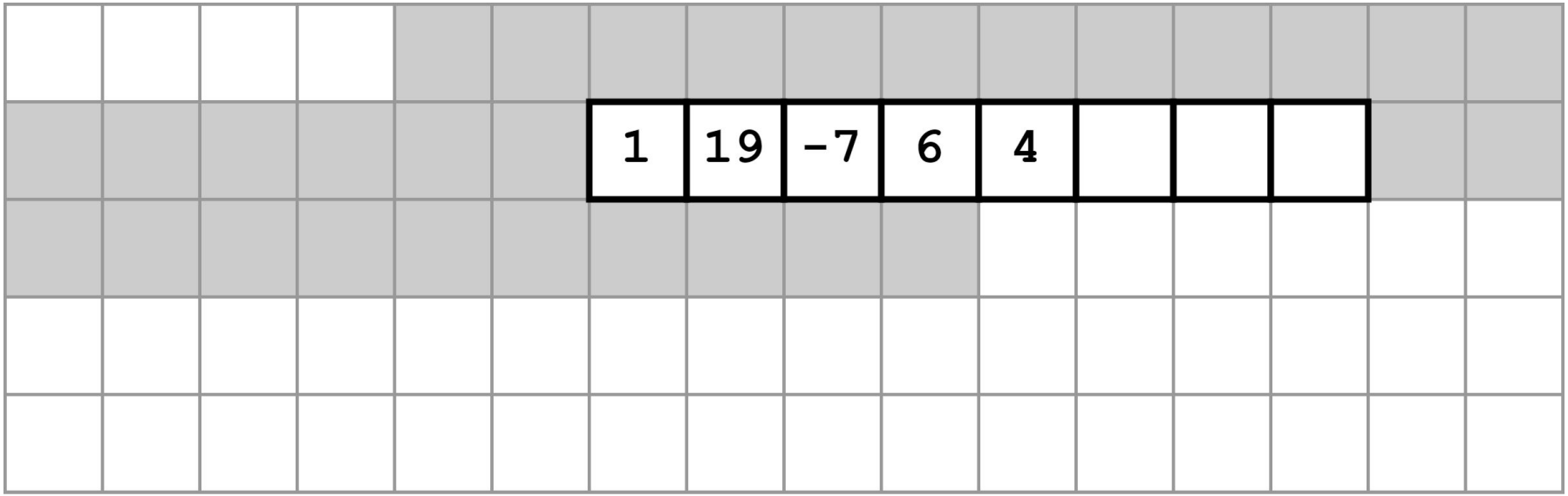
Program Memory

```
arr = temp;
```



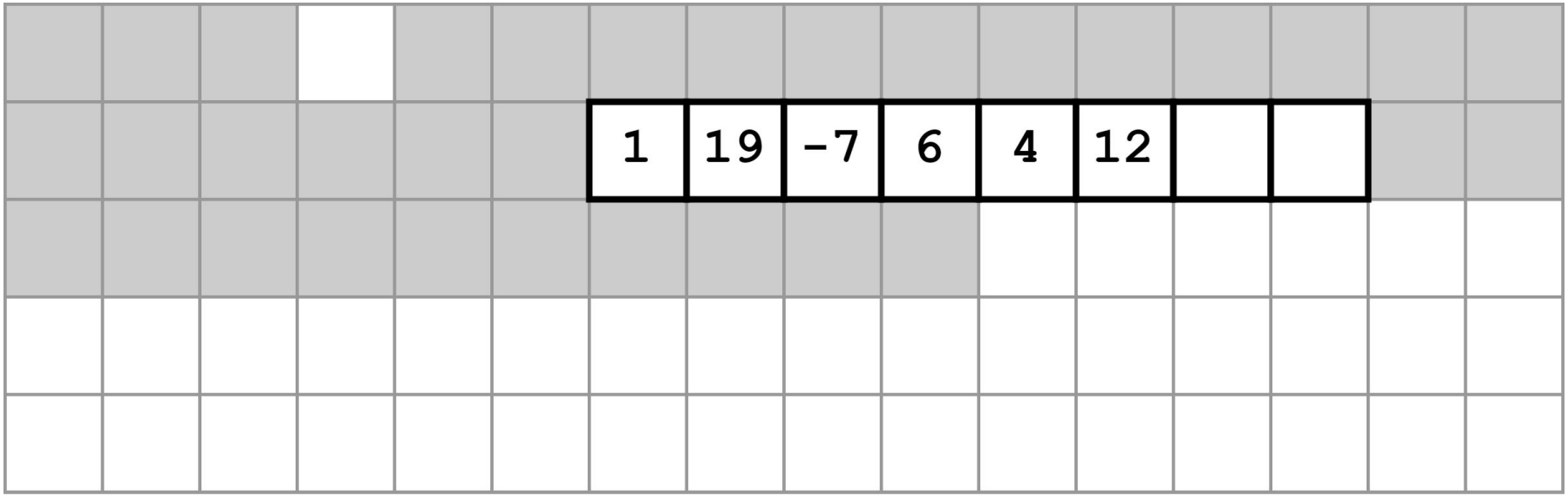
Program Memory

```
arr[4] = 4;
```



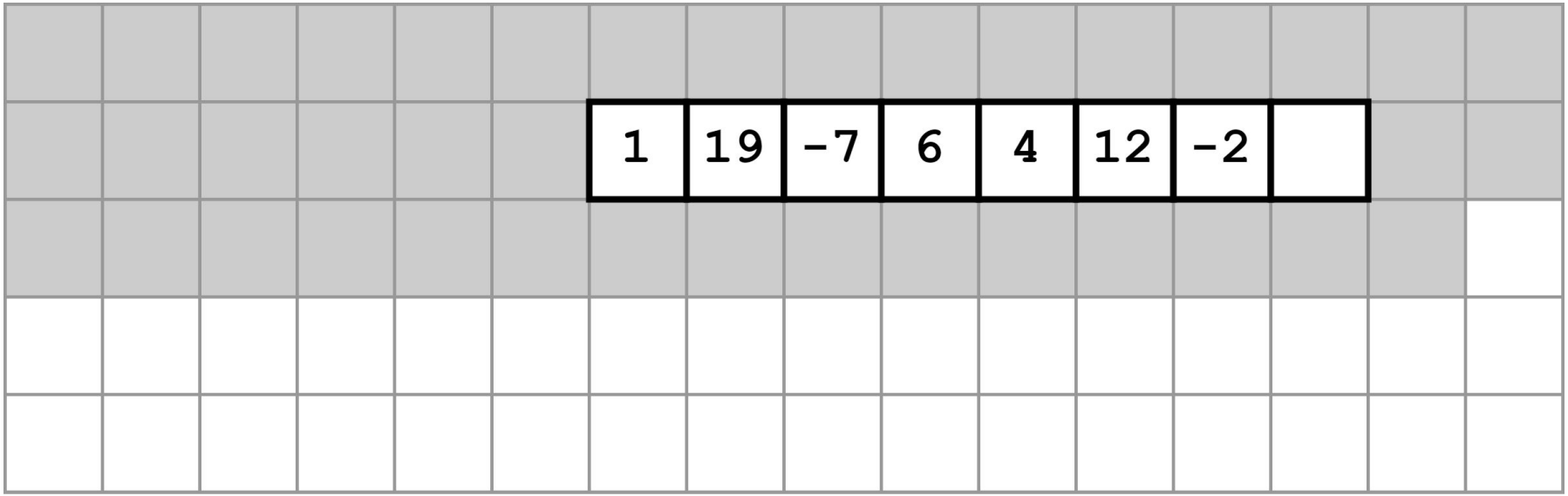
Program Memory

```
arr[5] = 12;
```



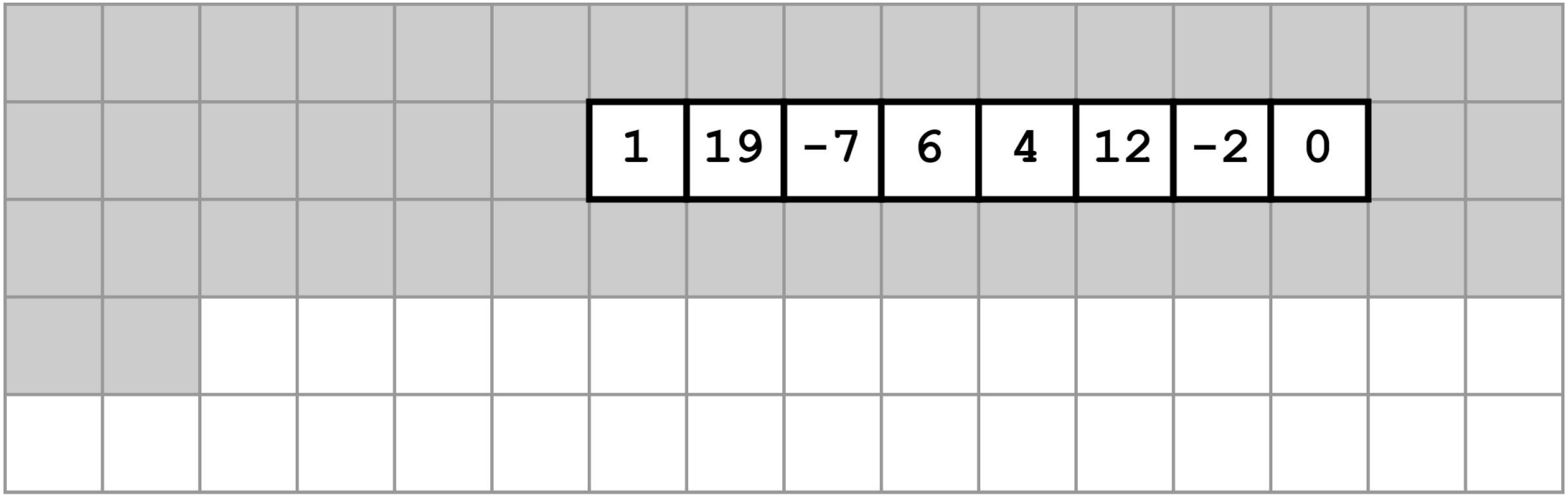
Program Memory

```
arr[6] = -2;
```



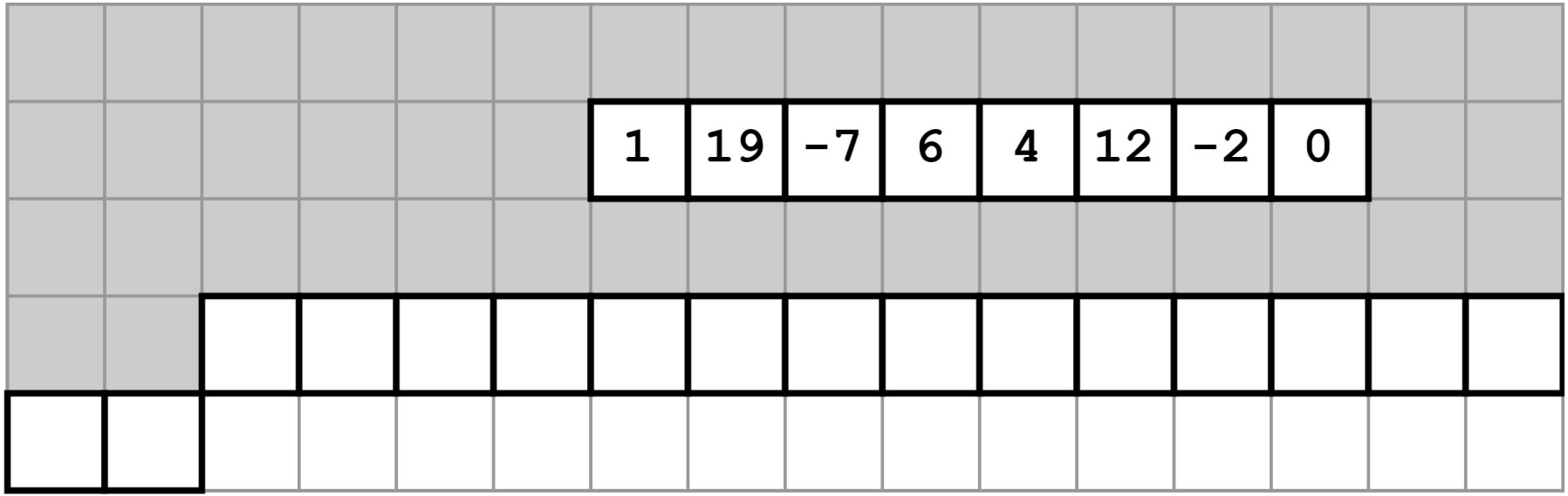
Program Memory

```
arr[7] = 0;
```

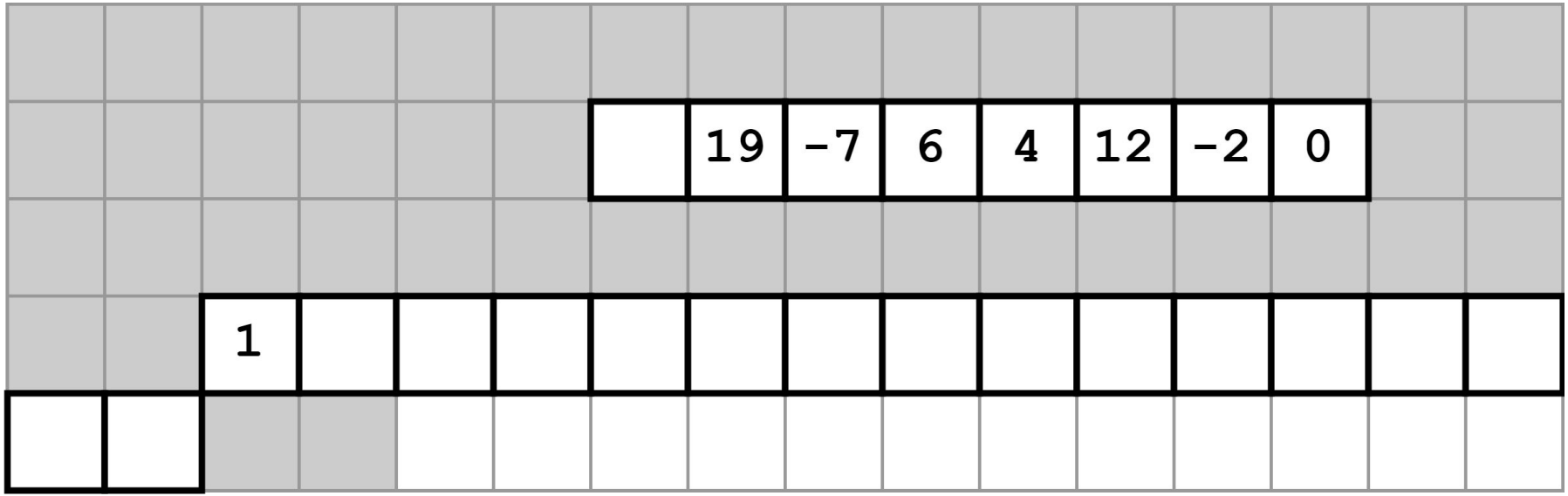


Program Memory

```
Integer[] temp = new Integer[16];
```

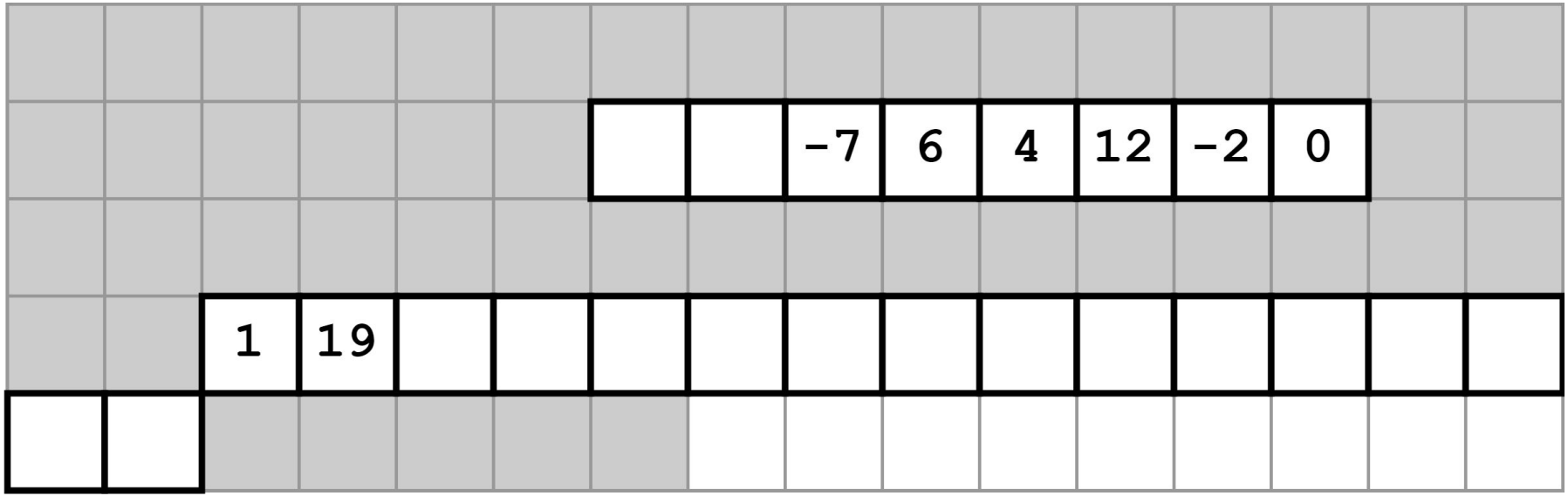


```
temp[0] = arr[0];
```



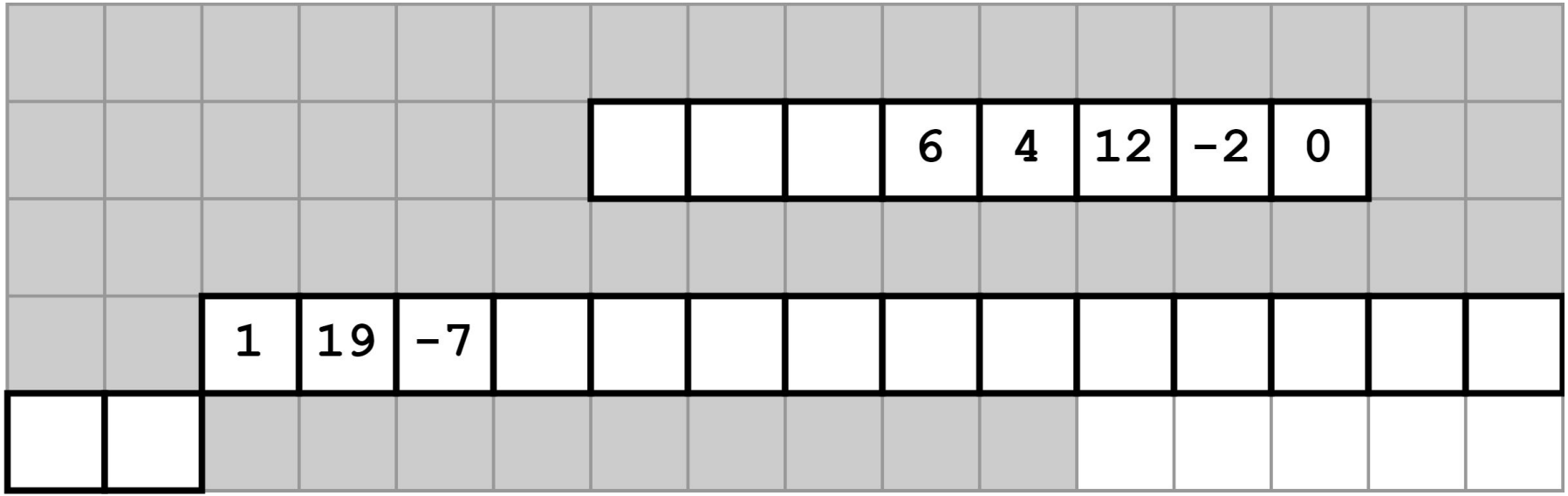
Program Memory

```
temp[1] = arr[1];
```



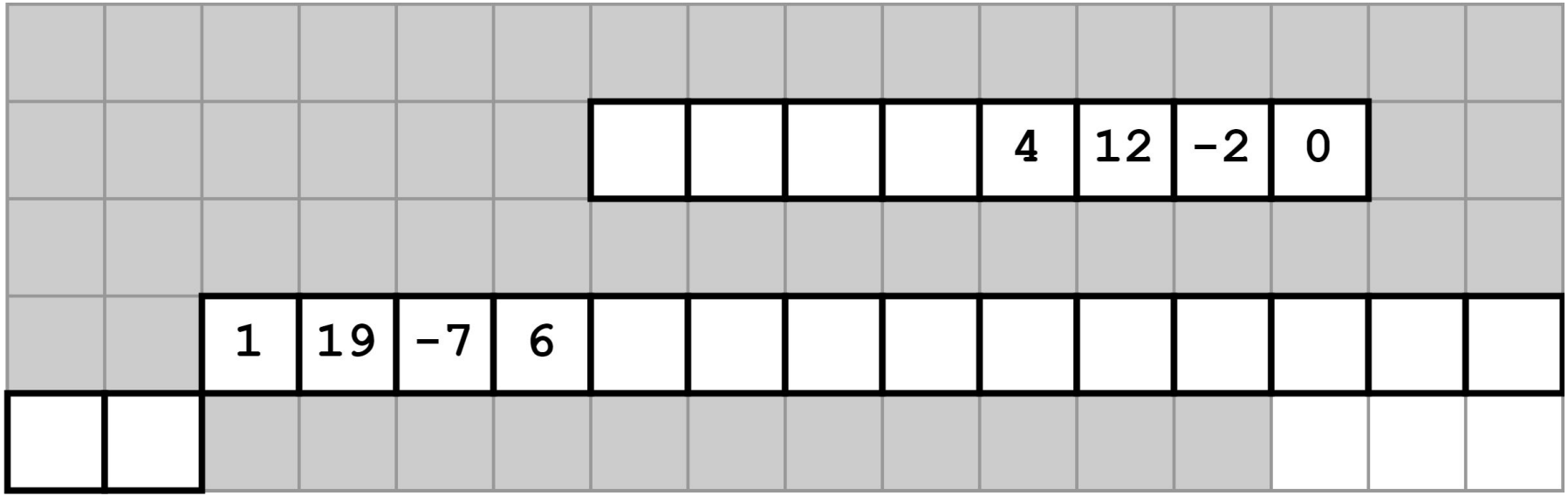
Program Memory

```
temp[2] = arr[2];
```



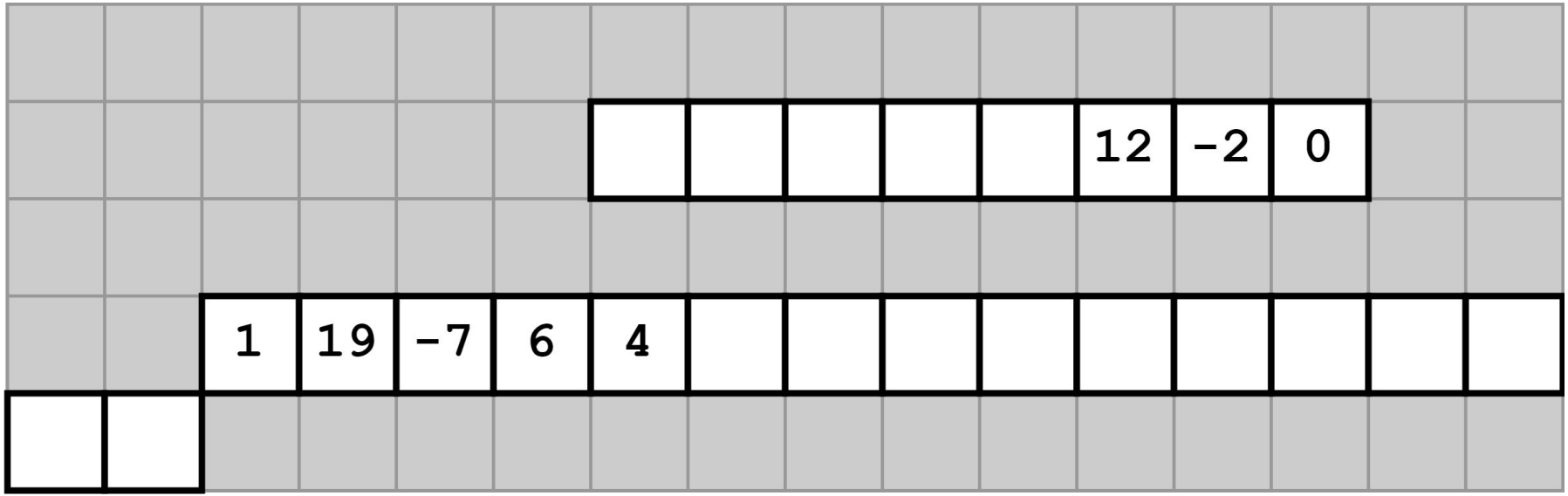
Program Memory

```
temp[3] = arr[3];
```



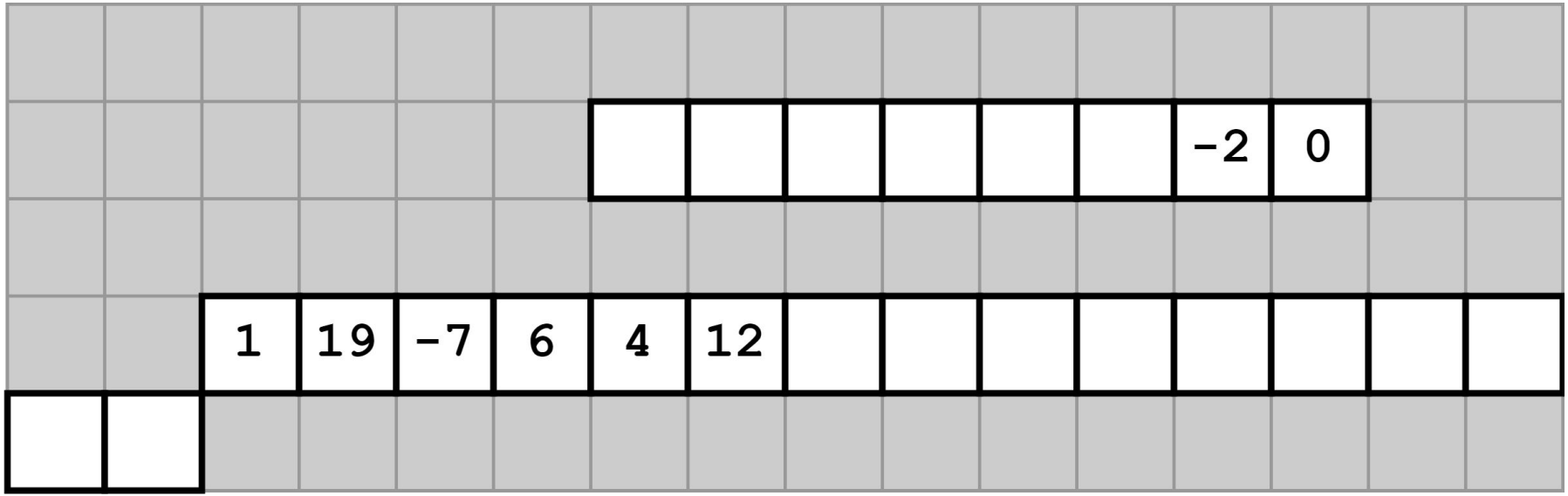
Program Memory

```
temp[4] = arr[4];
```



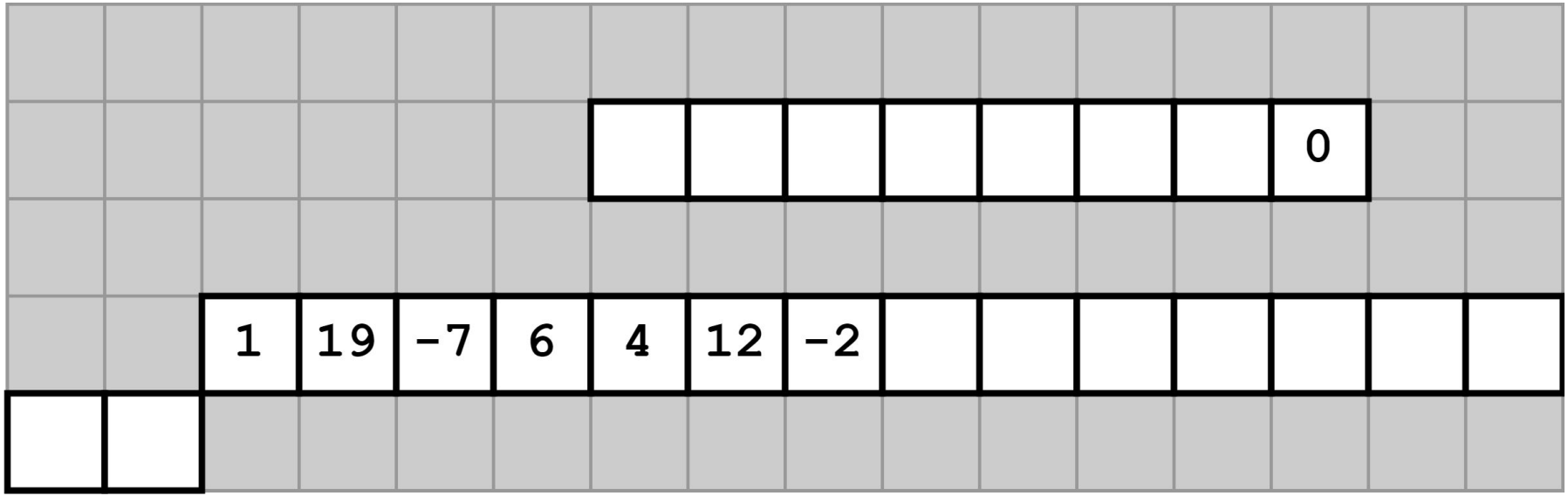
Program Memory

```
temp[5] = arr[5];
```



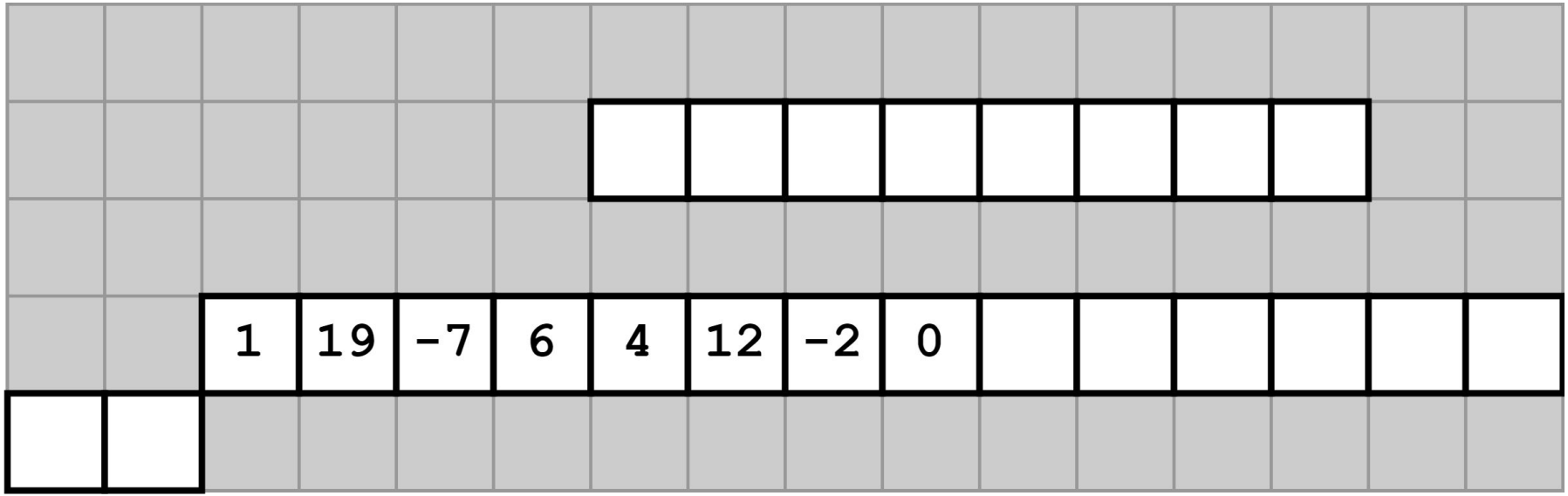
Program Memory

```
temp[6] = arr[6];
```



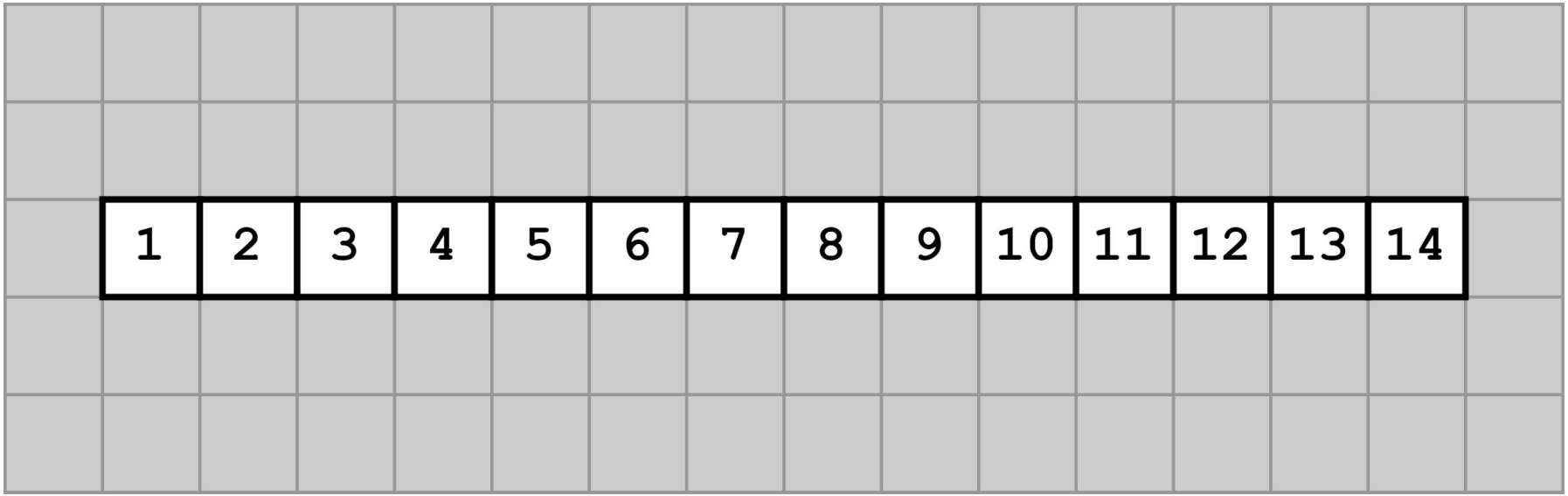
Program Memory

```
temp[7] = arr[7];
```

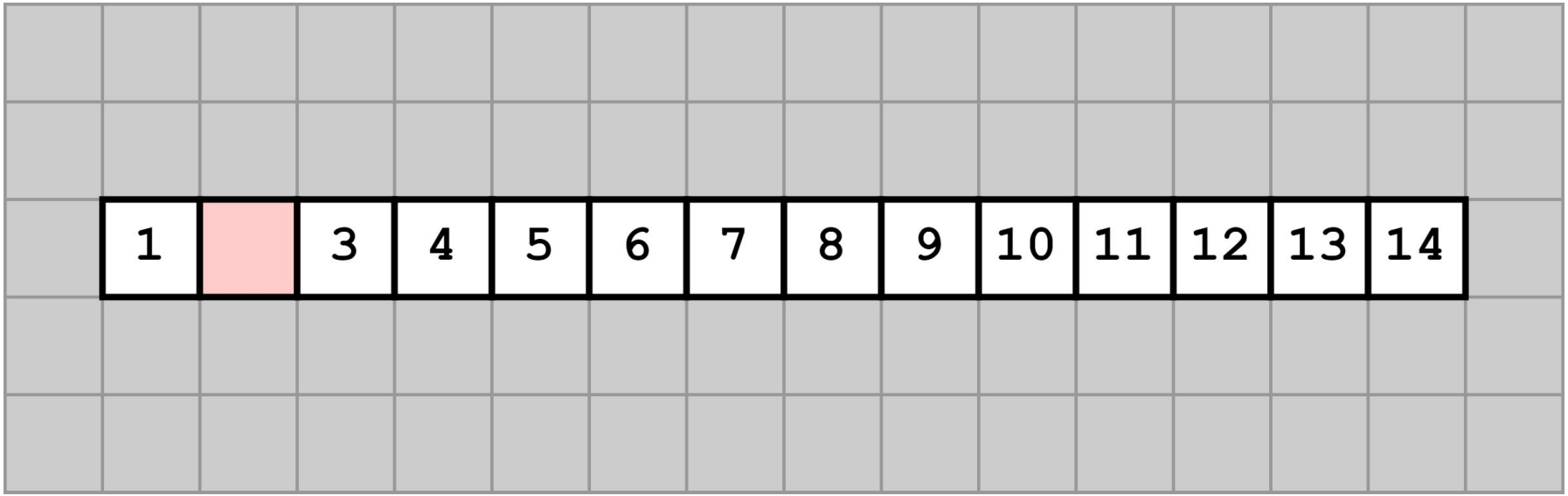


Program Memory

A new array issue

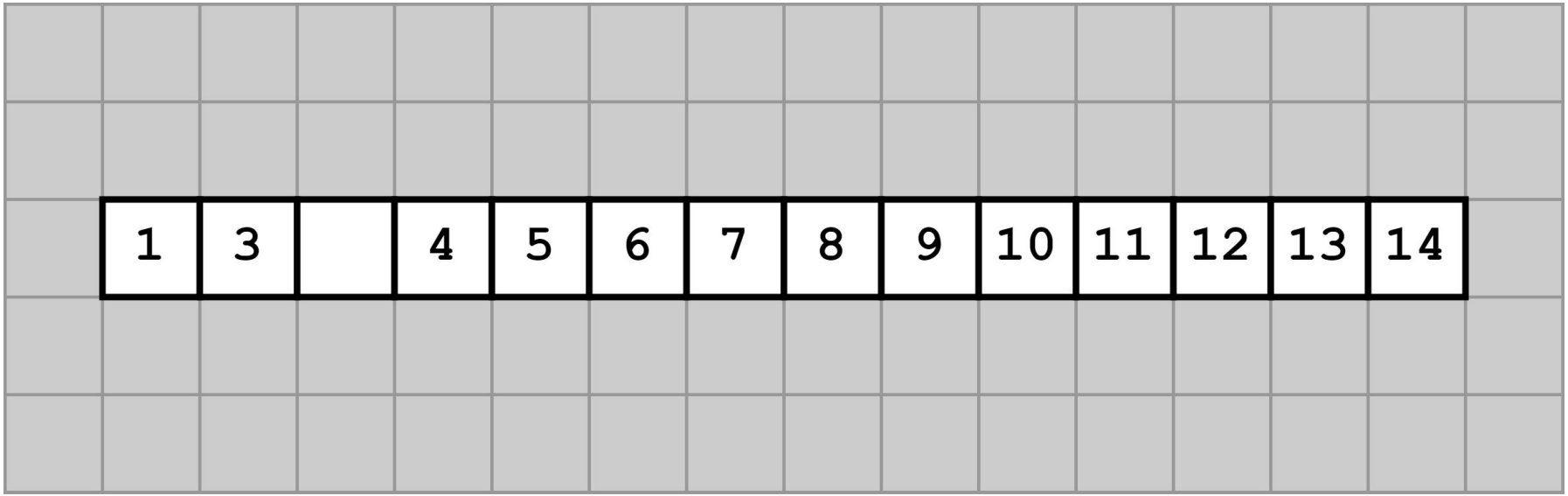


```
arr[1] = null;
```



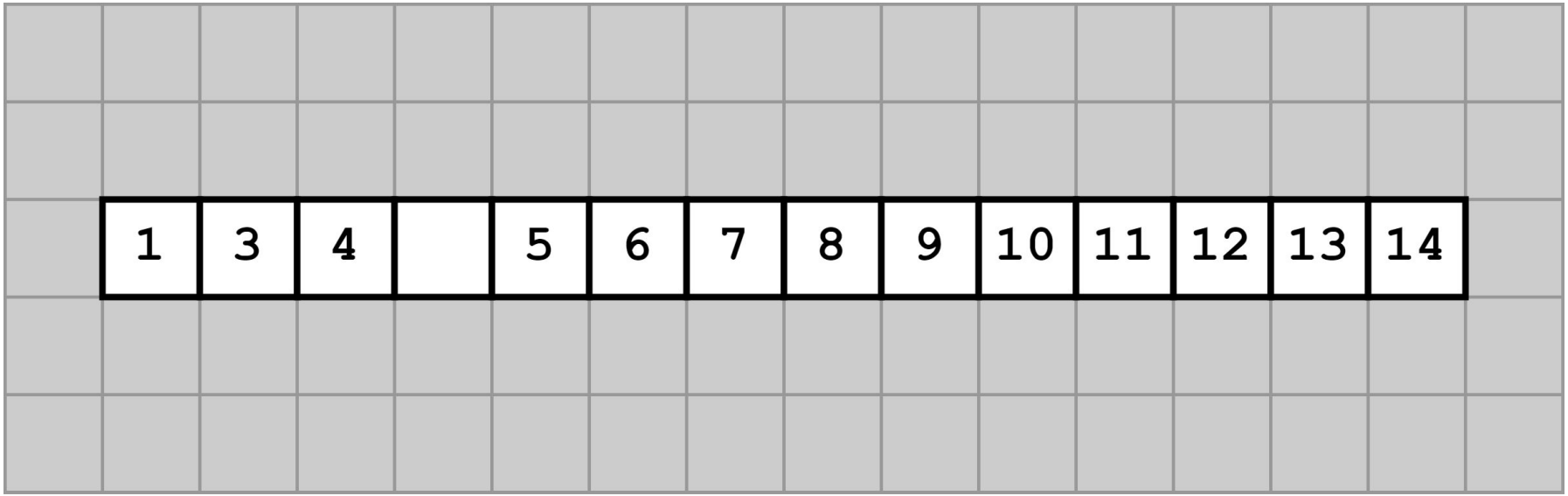
Program Memory

```
arr[1] = arr[2]; arr[2] = null;
```



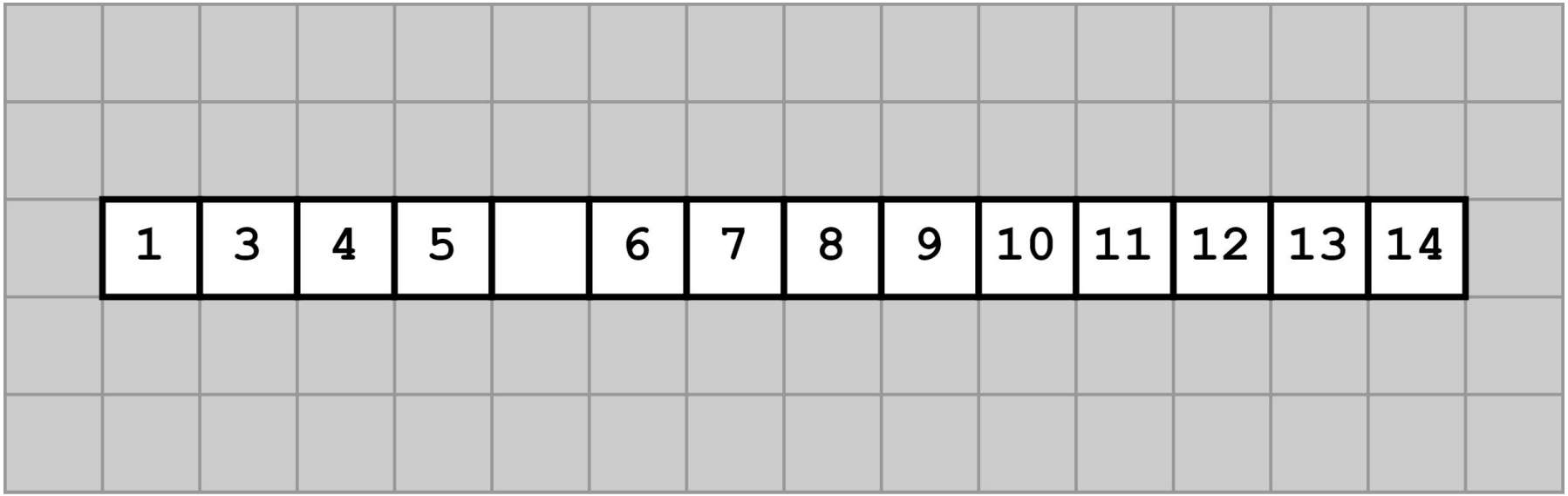
Program Memory

```
arr[2] = arr[3]; arr[3] = null;
```



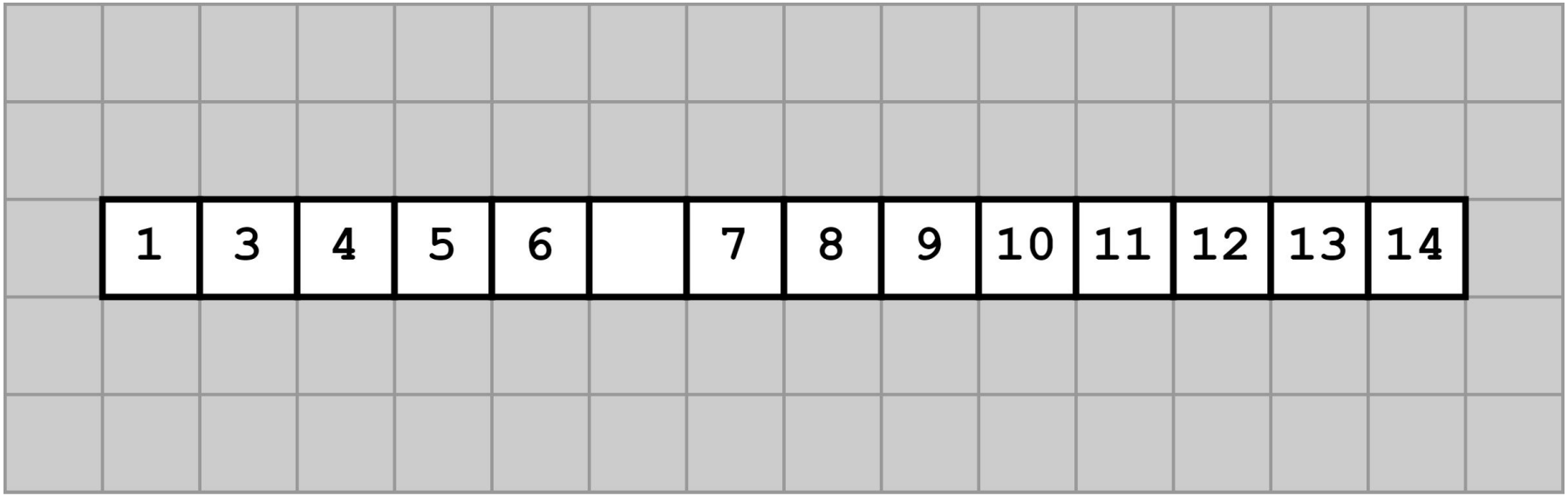
Program Memory

```
arr[3] = arr[4]; arr[4] = null;
```



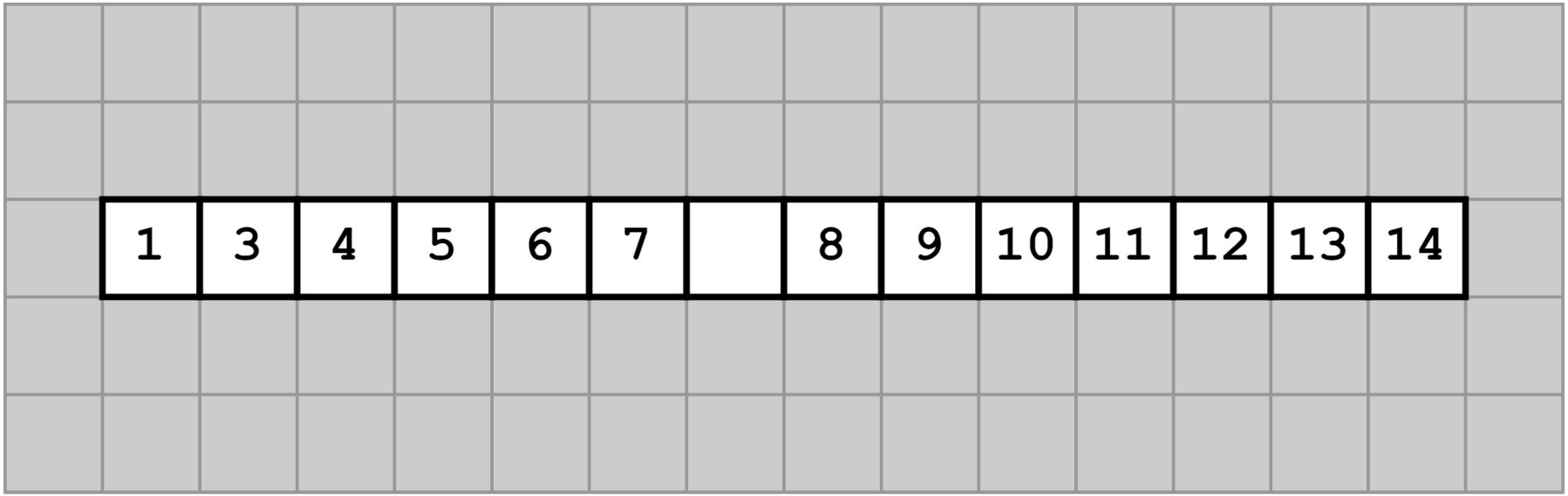
Program Memory

```
arr[4] = arr[5]; arr[5] = null;
```



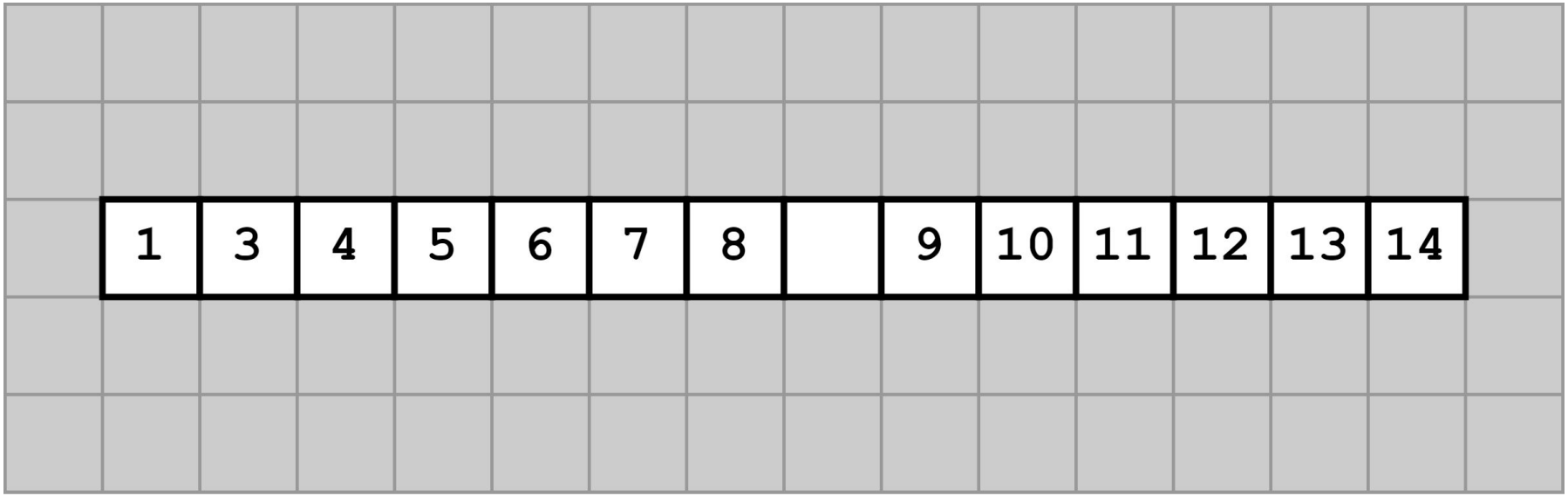
Program Memory

```
arr[5] = arr[6]; arr[6] = null;
```



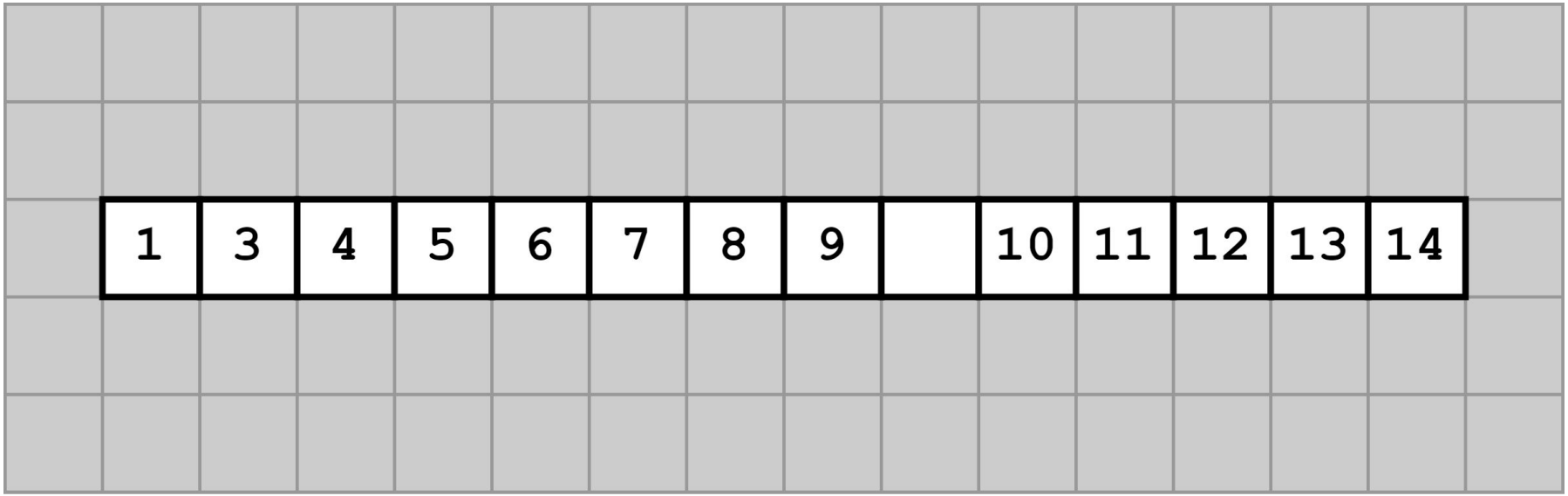
Program Memory

```
arr[6] = arr[7]; arr[7] = null;
```



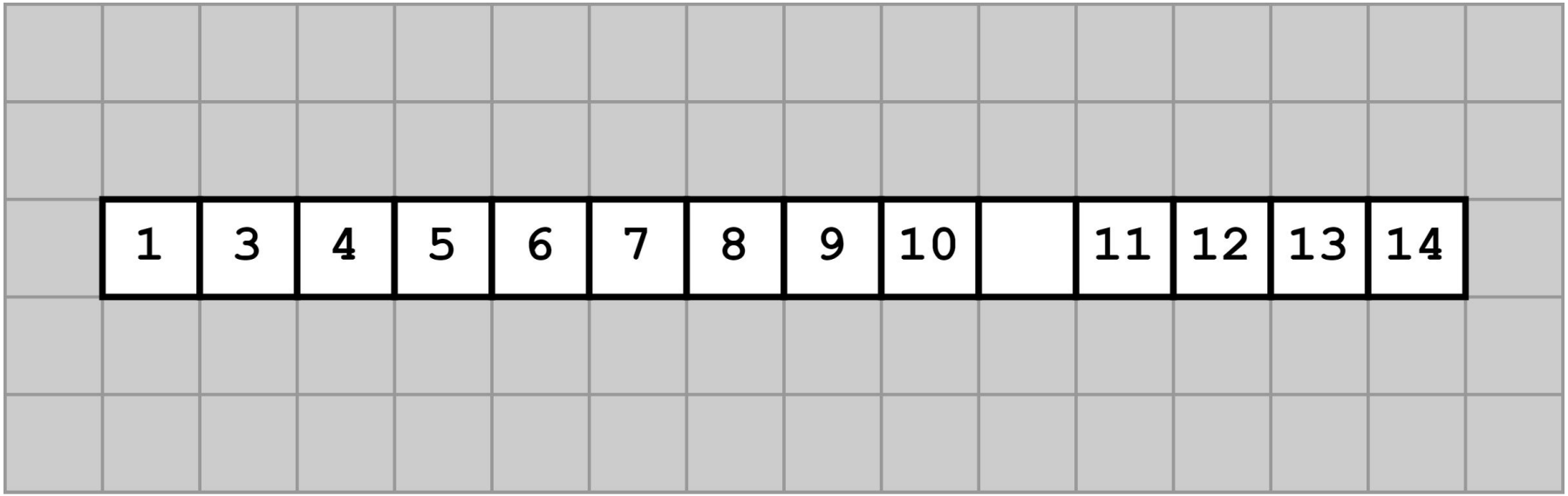
Program Memory

```
arr[7] = arr[8]; arr[8] = null;
```



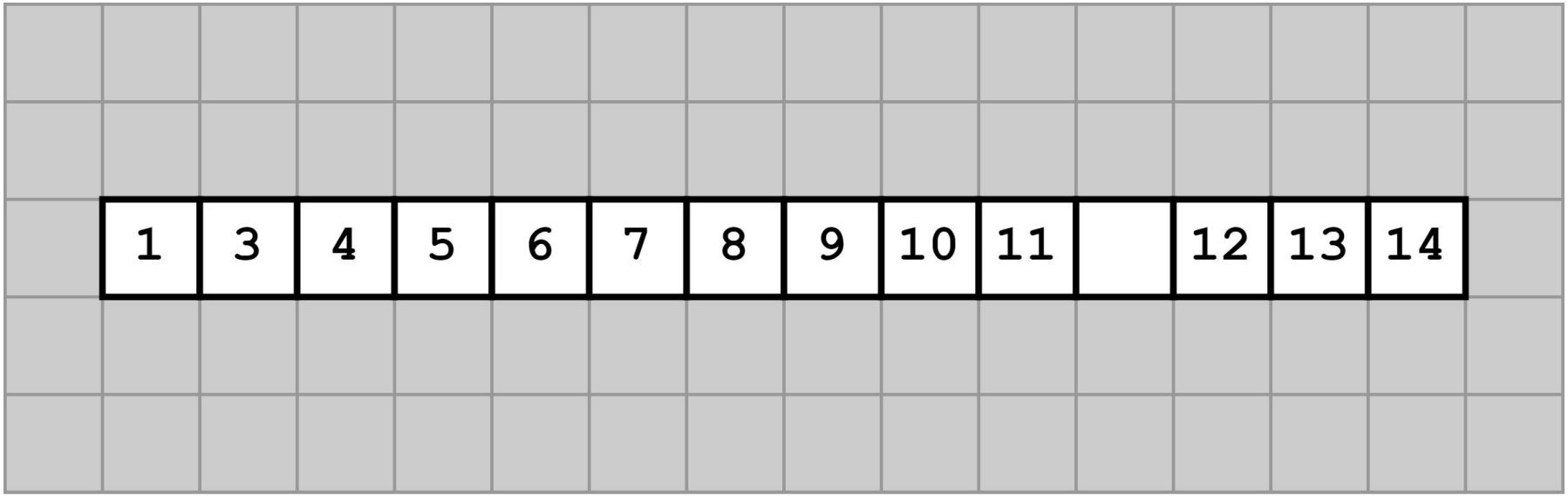
Program Memory

```
arr[8] = arr[9]; arr[9] = null;
```



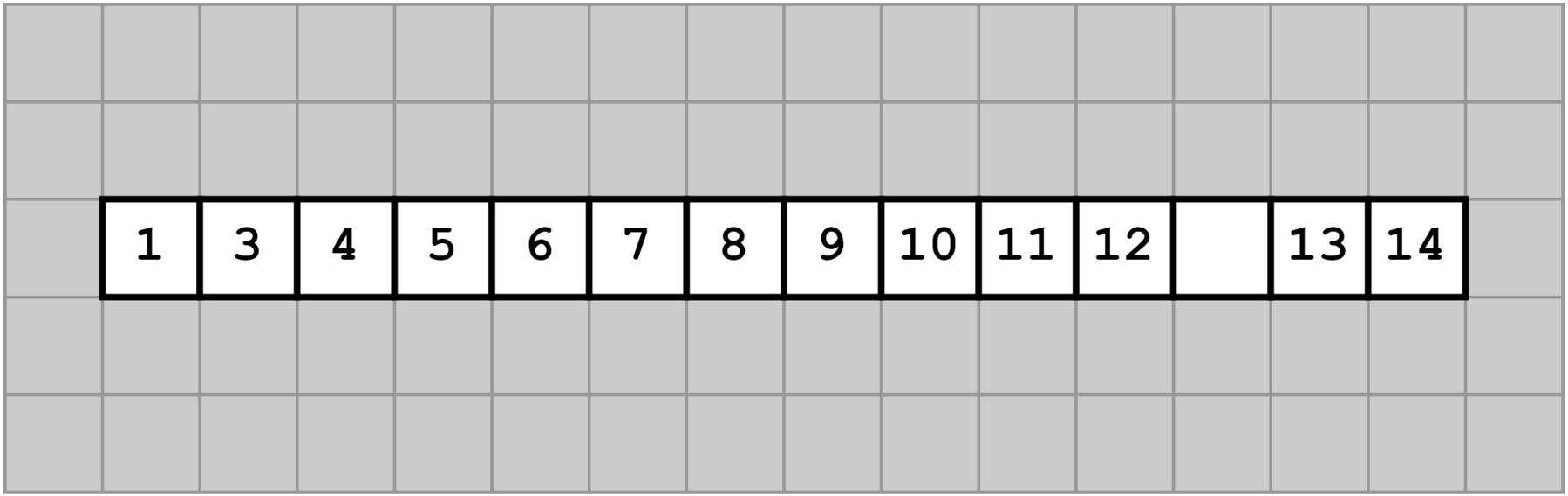
Program Memory

```
arr[9] = arr[10]; arr[10] = null;
```



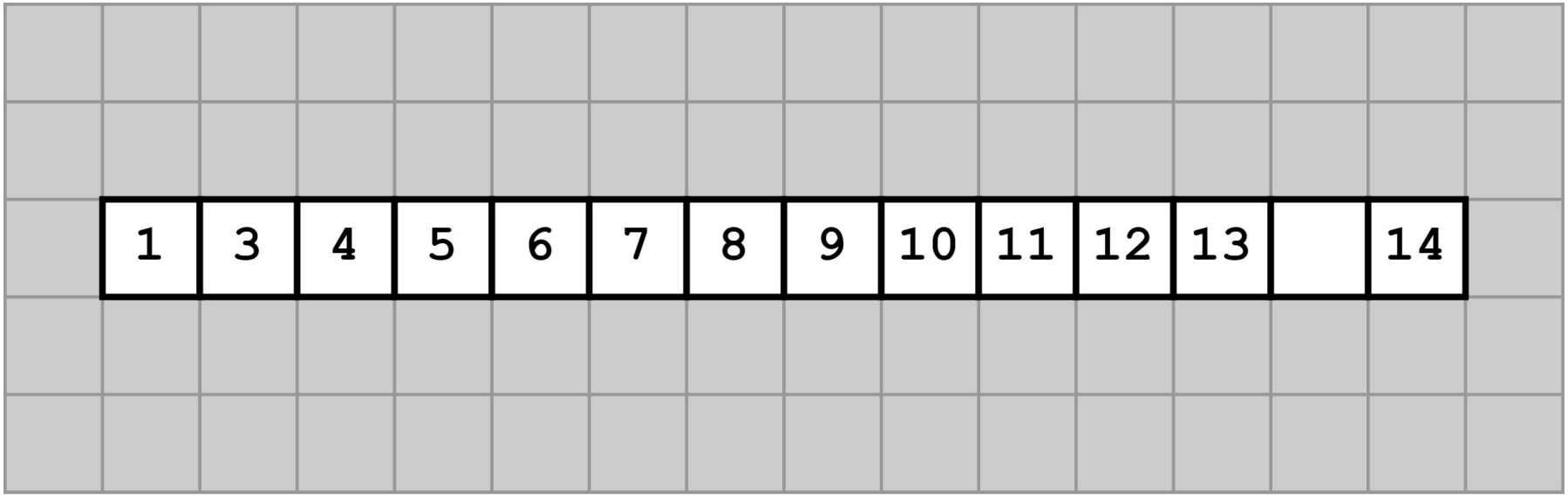
Program Memory

```
arr[10] = arr[11]; arr[11] = null;
```



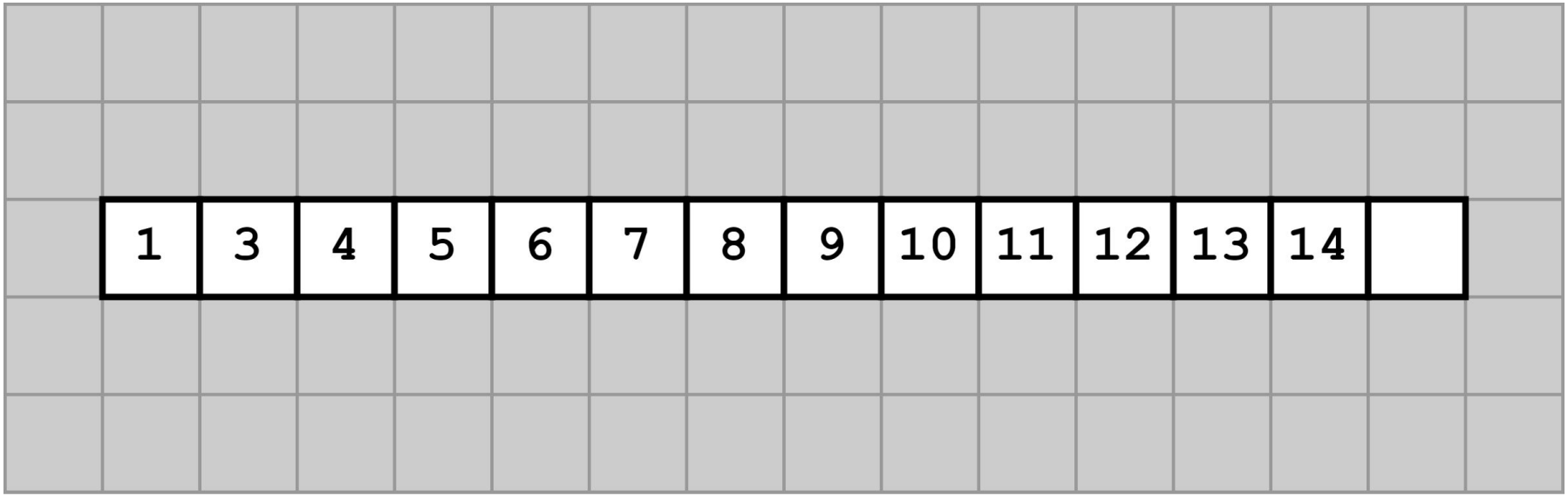
Program Memory

```
arr[11] = arr[12]; arr[12] = null;
```



Program Memory

```
arr[12] = arr[13]; arr[13] = null;
```



Program Memory

This is how ArrayList works

Because of this:

- add(int index, E element) takes O(n)
- remove(int index) takes O(n)

This is how ArrayList works

Because of this:

- add(int index, E element) takes O(n)
- remove(int index) takes O(n)
- add(E element) takes O(1)*

This is how ArrayList works

Because of this:

- add(int index, E element) takes O(n)
- remove(int index) takes O(n)
- add(E element) takes O(1)*

*Operation takes O(1) **amortized time**. The philosophy here is to pay attention to the worst-case runtime *per algorithm*, instead of *per operation*.

Source: https://en.wikipedia.org/wiki/Amortized_analysis

Iterators

What is an Iterator?

- Provide way to sequentially access elements in a Collection
- Some collections have no “get” method, must use an iterator

```
Iterator<E> iter = collection.iterator();
while(iter.hasNext())
{
    E element = iter.next();
    //Do stuff with "element"
}
```

What is a ListIterator?

- Subtype of Iterator
- Since lists are easily mutable, gives access to more mutation/traversal methods

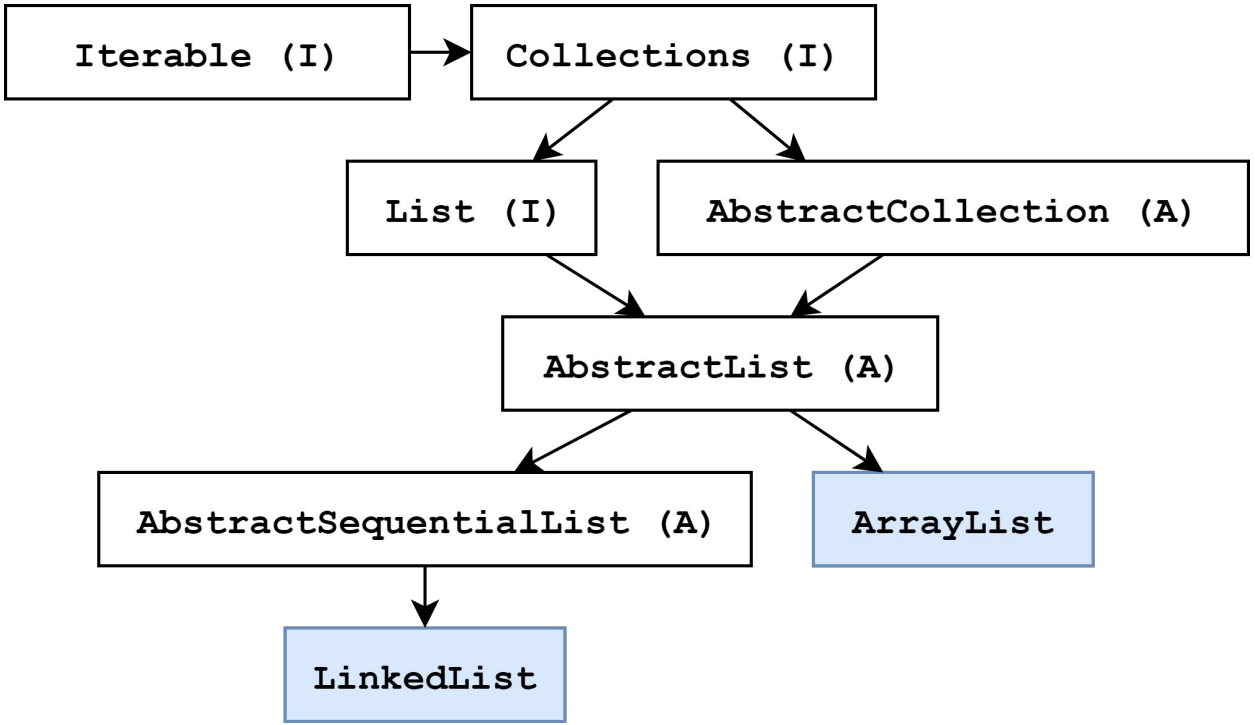
Iterator<E>

next()
hasNext()
remove()
forEachRemaining()

ListIterator<E>

next()
hasNext()
remove()
forEachRemaining()
previous()
hasPrevious()
nextIndex()
previousIndex()
set()
add()

Iterable Hierarchy



End