Arrays

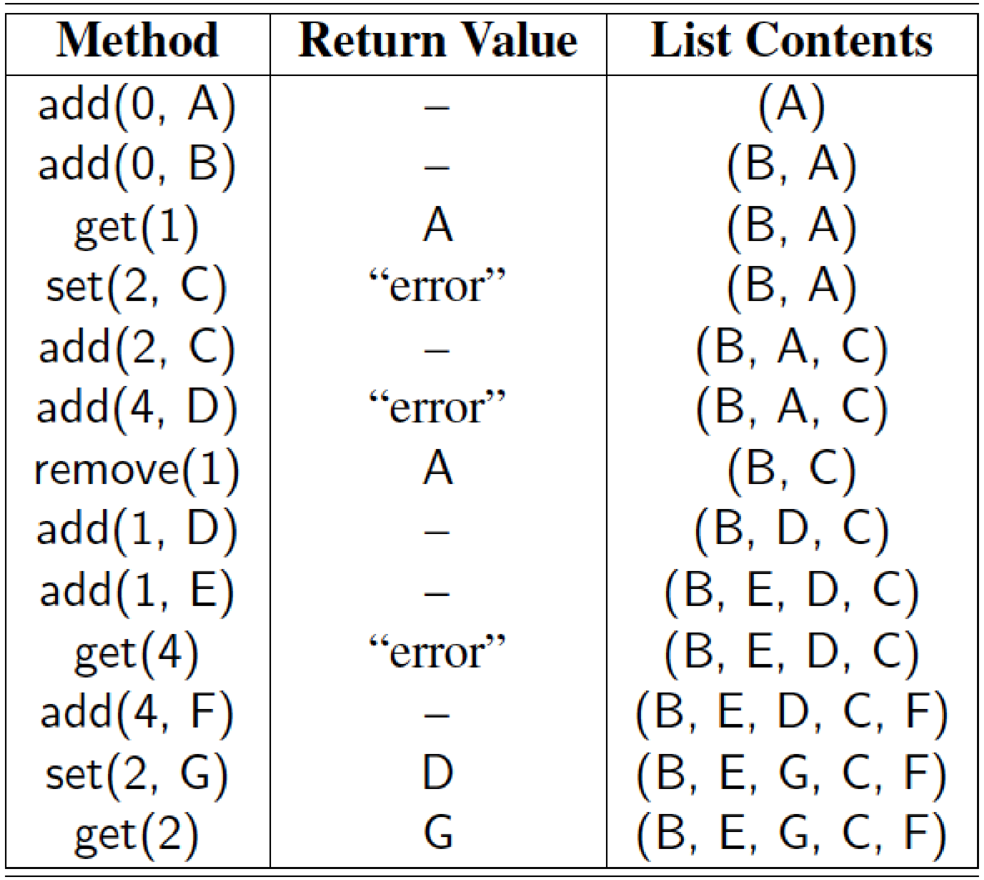
ENSF 594 – Principles of Software Development II

UNIVERSITY OF CALGARY

Abstract Data Type (ADT) LISTS

- A linearly ordered sequence of elements of the same type

Methods	Description
size()	Return the number of elements in the list.
isEmpty()	Return true if the list is empty, otherwise return false.
isFull()	Return true if the list is full, otherwise return false.
get(i)	Return an element from the list at any given position i.
set(i,e)	Replace an element at any position i by another element e.
add(i,e)	Insert an element e at any position i of the list.
remove(i)	Remove the element at a specified location from a non-empty list.

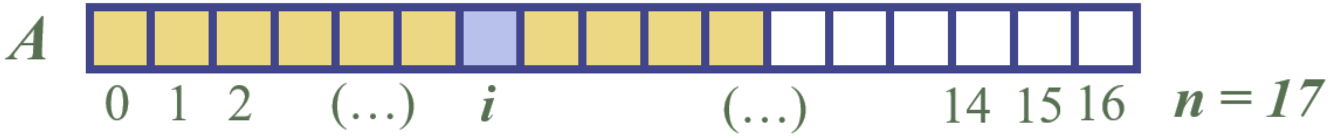


Method	Return Value	List Contents
add(0, A)	–	(A)
add(0, B)	–	(B, A)
get(1)	A	(B, A)
set(2, C)	"error"	(B, A)
add(2, C)	–	(B, A, C)
add(4, D)	"error"	(B, A, C)
remove(1)	A	(B, C)
add(1, D)	–	(B, D, C)
add(1, E)	–	(B, E, D, C)
get(4)	"error"	(B, E, D, C)
add(4, F)	–	(B, E, D, C, F)
set(2, G)	D	(B, E, G, C, F)
get(2)	G	(B, E, G, C, F)

Data Structures → Linear → Static: ARRAYS

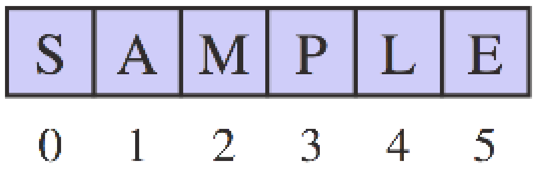
Definition

- ✓ An obvious choice for implementing the list ADT is to use an ***array***
- ✓ An ***array*** is a sequenced collection of variables all of the same type.
- ✓ Each variable, or ***cell***, in an array has an ***index***, which uniquely refers to the value stored in that cell.
- ✓ The cells of an array, A, are numbered 0, 1, 2, and so on.
- ✓ Each value stored in an array is often called an ***element*** of that array.

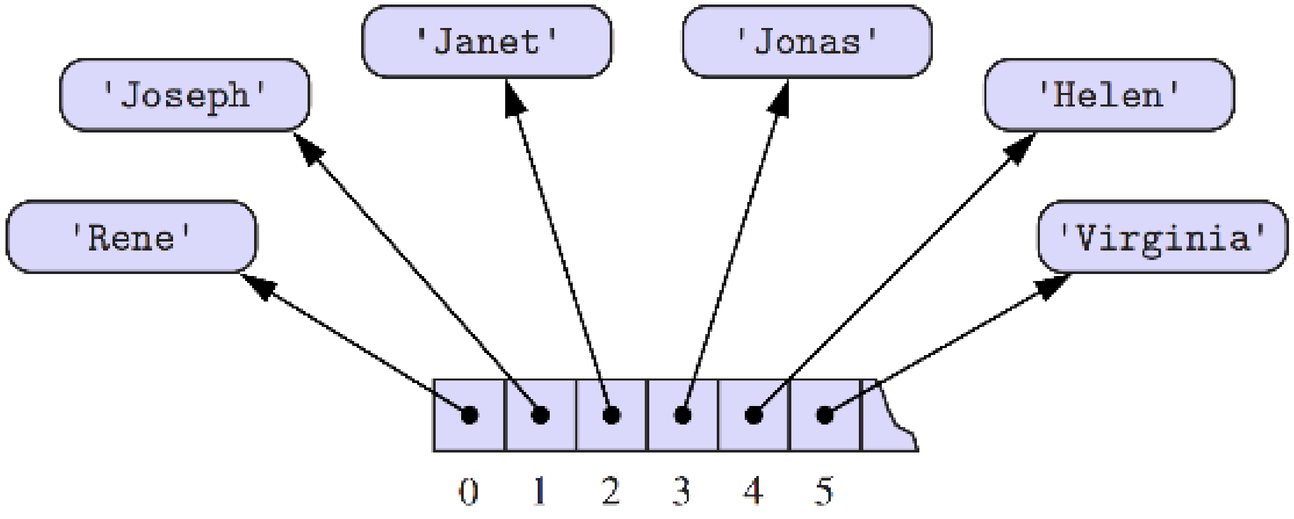


Data Structures → Linear → Static: ARRAYS
Definition

- ✓ An array can store primitive elements, such as characters and numbers.



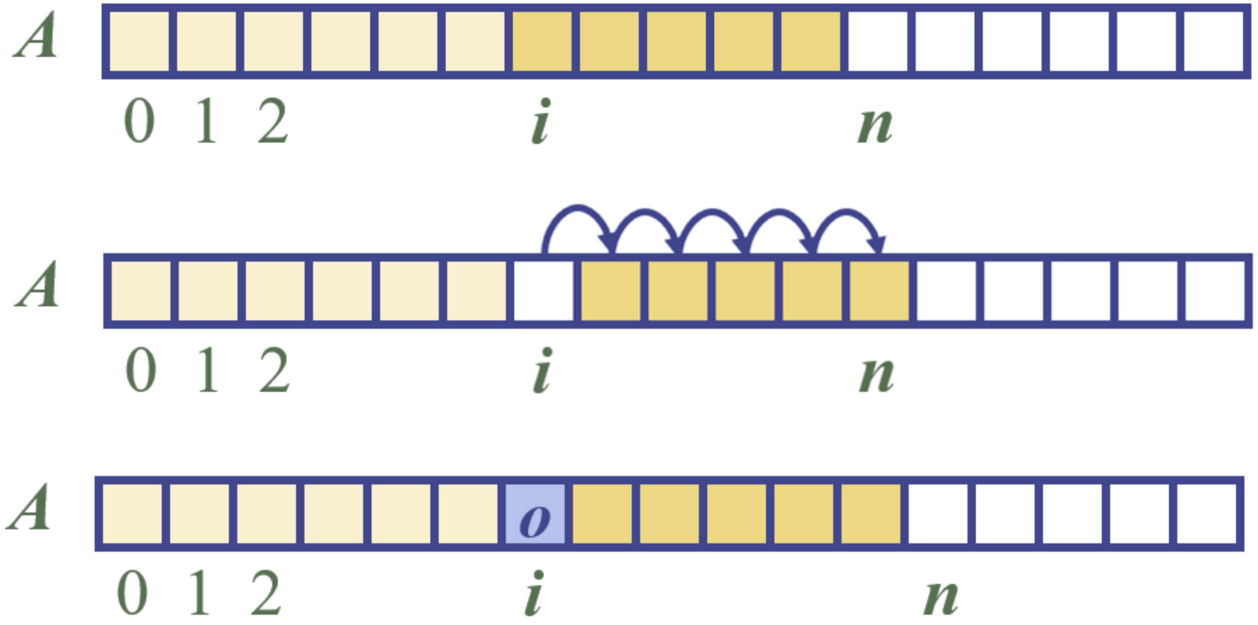
- ✓ An array can also store references (i.e., pointers) to objects.



Data Structures → Linear → Static: ARRAYS
Definition

- In an operation ***add***(***i*, *data item***), we need to make room for the new element by shifting forward the ***n*** - ***i*** elements ***A***[***i***], …, ***A***[***n*** - 1]
- In the worst case (***i*** = 0), this takes ***O***(***n***) time

***add*(*i*, *data item*)**

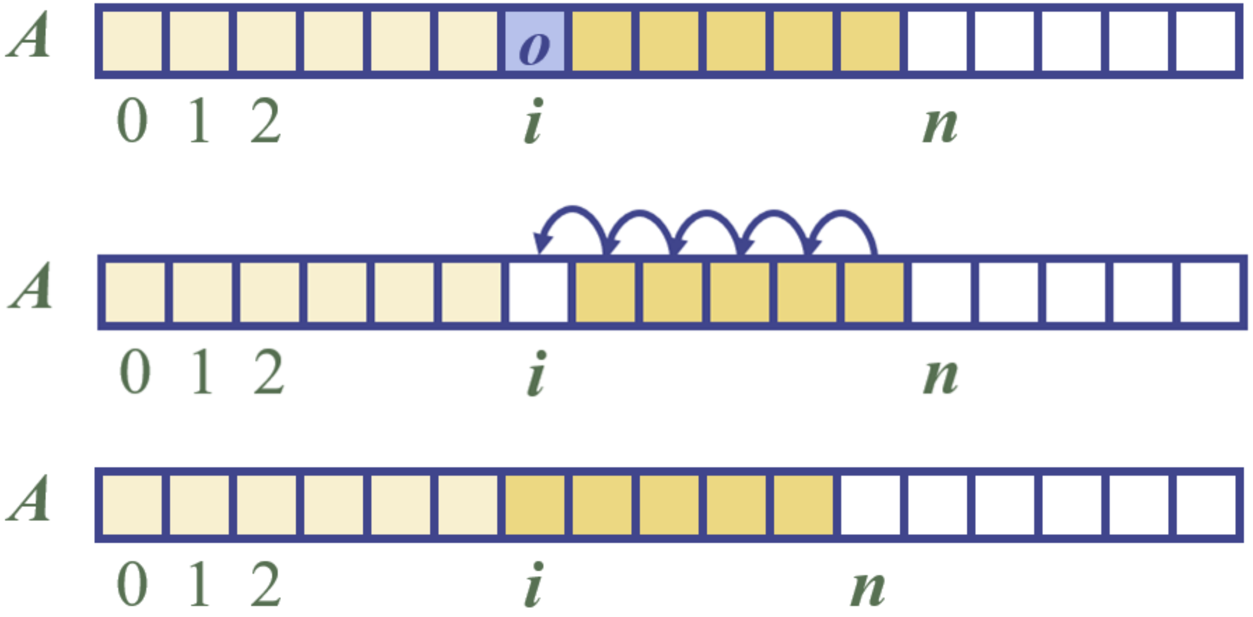


***add*(*i*, *o*)**

Data Structures → Linear → Static: ARRAYS

Definition

- In an operation ***remove***(i), we need to fill the hole left by the removed element by shifting backward the ***n*** - ***i*** - 1 elements ***A***[***i*** + 1], …, ***A***[***n*** - 1]
- In the worst case (***i*** = 0), this takes ***O***(***n***) time



remove(***i***)

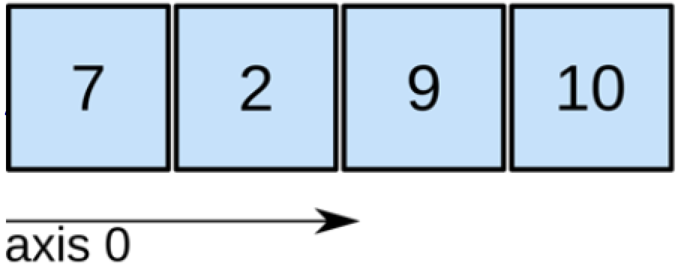
Data Structures → Linear → Static: ARRAYS

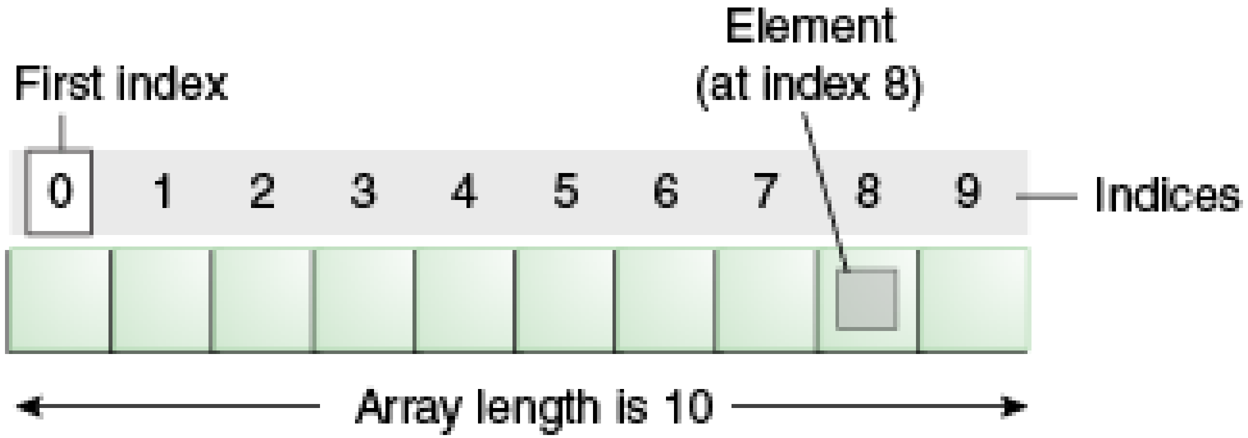
Definition

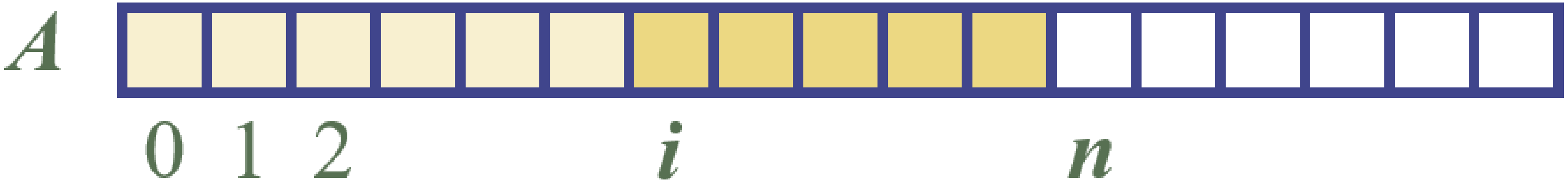
List ADT Methods	Asymptotic Worst Case Performance
`add(i,e)`	$O(n)$
`remove(i)`	$O(n)$

Data Structures → Linear → Static: ARR

1-D (one-dimensional)







Matrix = “pointer to a pointer” of MY_DATA

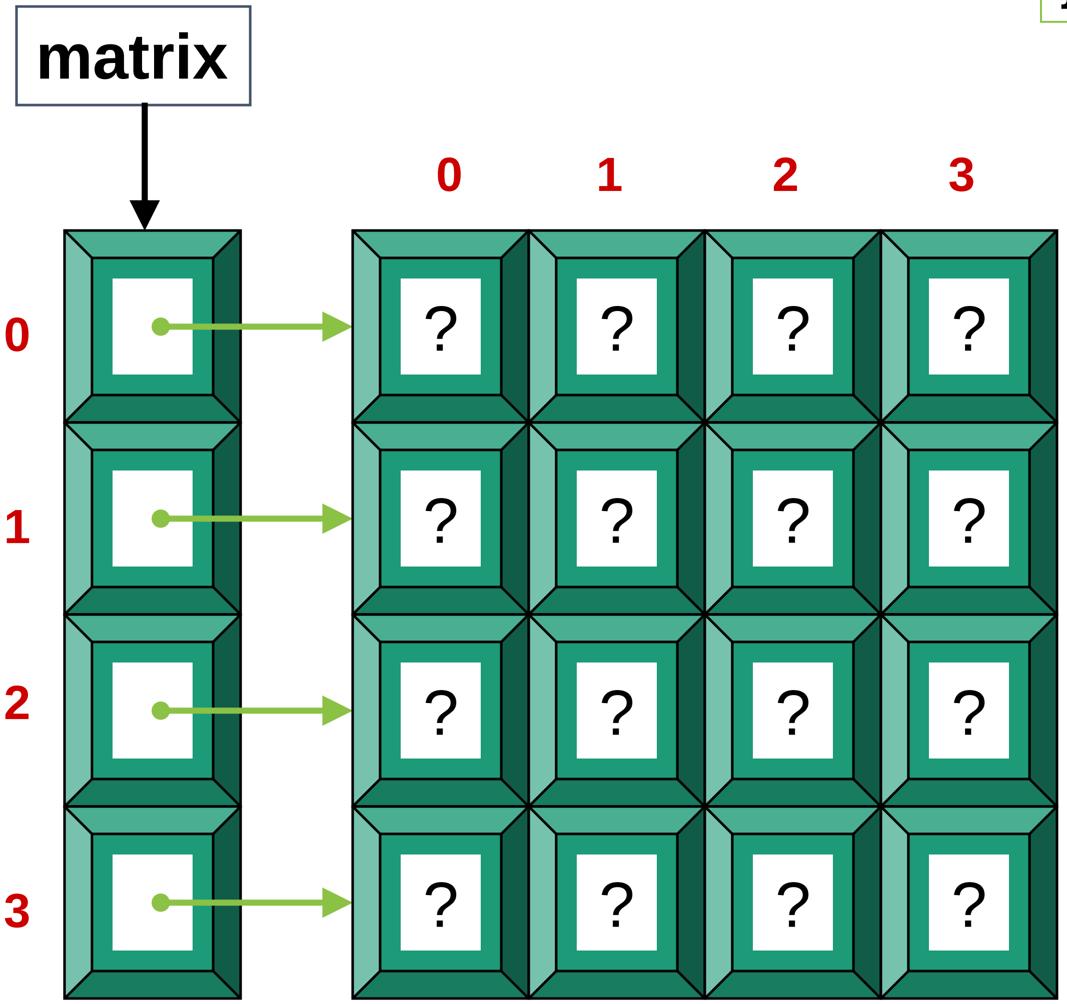
Example: nrows = 4, ncols = 4

```
matrix = allocate nrows nodes;

for (i = 0; i < nrows; i++)
{
matrix [i] = allocate ncols nodes;
}
```

create

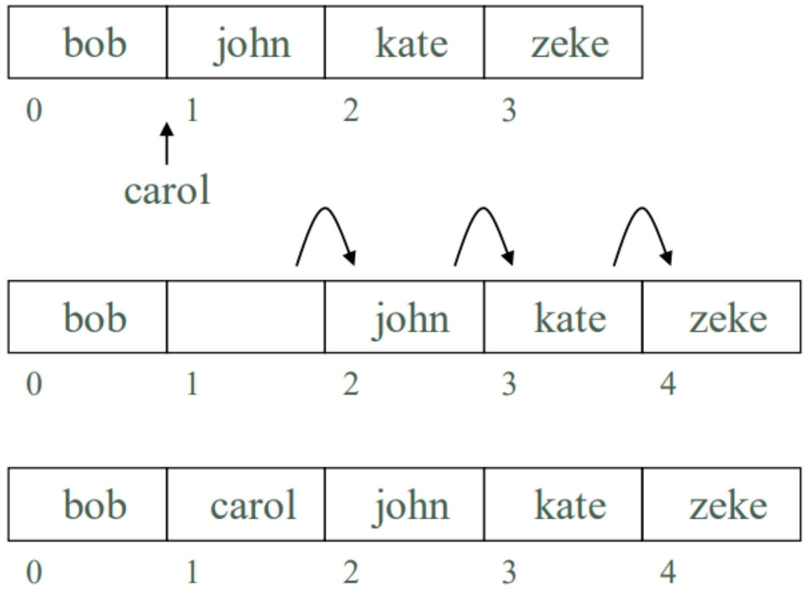
traverse



```
for (i = 0; i < nrows; i++)
{
   for (j = 0; j < ncols; j++)
   {
       matrix [i] [j] = value;
   }
}
```

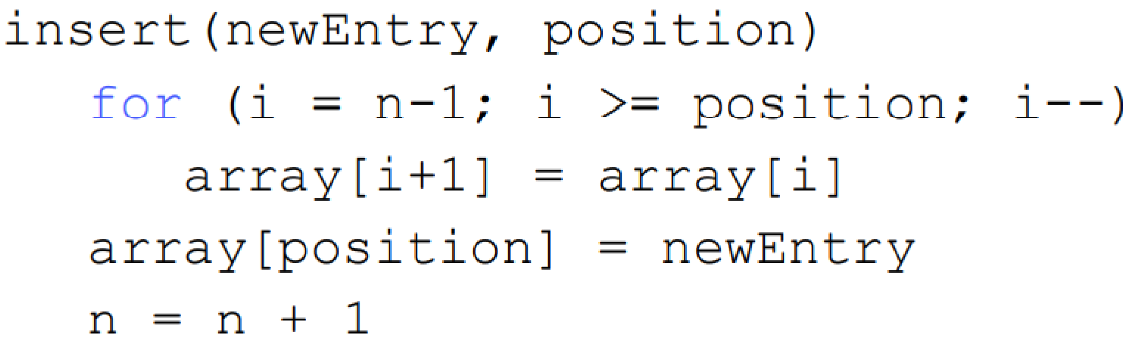
Insert/Add element in an array

- Inserting an item into an array may require shifting elements to make room for it



Insert/Add element in an array

- Pseudocode:

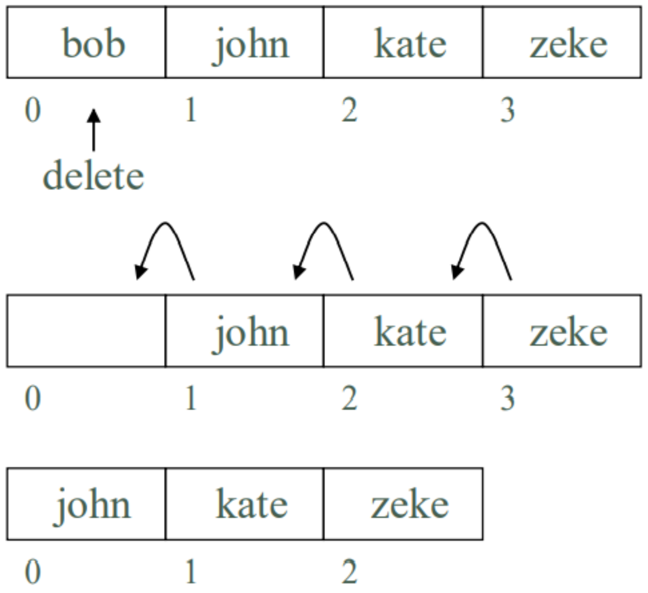


```
insert(newEntry, position)
   for (i = n-1; i >= position; i--)
      array[i+1] = array[i]
   array[position] = newEntry
   n = n + 1
```

- Is *O*(*n*) in the worst case (inserting at position 0)

Delete/Remove element from an array

- Deleting an item may require shifting items to fill the gap



bob	john	kate	zeke
0	1	2	3

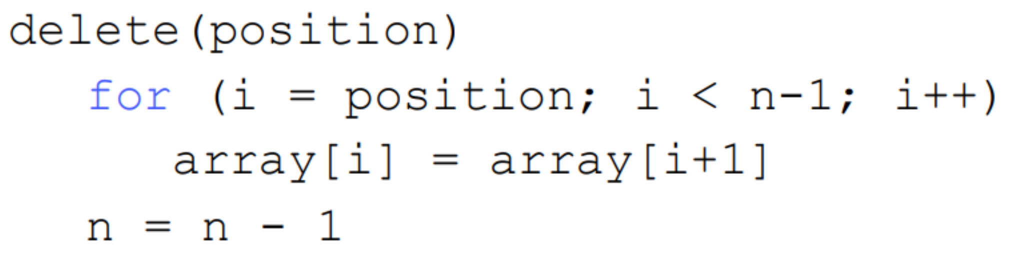
delete

	john	kate	zeke
0	1	2	3

john	kate	zeke
0	1	2

Delete/Remove element from an array

- Pseudocode:



```
delete(position)
   for (i = position; i < n-1; i++)
      array[i] = array[i+1]
   n = n - 1
```

- Is *O*(*n*) in the worst case (deleting at position 0)
- Accessing an item by position is *O*(1)
  - i.e. Getting or replacing entries is very quick

Limitations of List implemented as Java Arrays

- Length is not changeable at runtime
  - You may need to create a new, larger array and copy elements
    - – This is not a good way of doing it

- In addition to ordinary arrays, Java provides the classes: ▫
  - java.util.Vector
  - java.util.ArrayList

- We will see examples of ArrayList

Example of ArrayList

```
import java.util.ArrayList;
public class Week03 {
  public static void main(String[] args) {
       ArrayList<Integer> myList = new ArrayList<Integer>();
       for (int index = 0; index < 5; index++){
              myList.add(index);
       }
       int sum = 0;
       for (int num : myList){
              sum = sum + num;
       }
       System.out.println(myList);
       System.out.println(sum);
  }
}
```

```
Output:
[0, 1, 2, 3, 4]
10
```

Additional Resources for ArrayList

- Basic understanding about ArrayList

- https://www.w3schools.com/java/java_arraylist.asp