

LCS-ES Group Meeting #12

Exercise 1: Array Matching

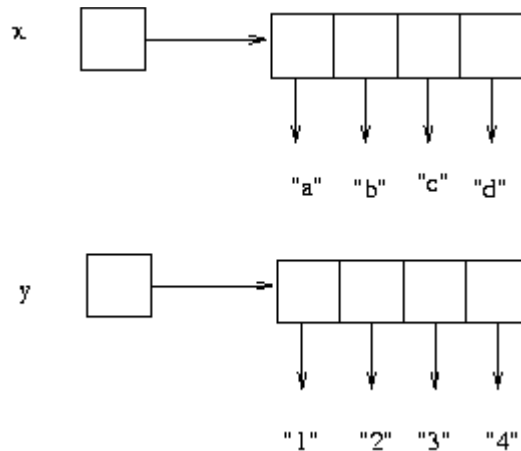
Match code snippets (on blue paper) with their descriptions (pink paper). In the code snippets assume that `arr` is a non-empty, initialized array of `int` values.

Exercise 2: Practice with Arrays

Assume that you have two arrays, `x` and `y`, declared as follows

```
String [] x = new String[4];  
String [] y = new String[4];
```

and initialized as shown below.

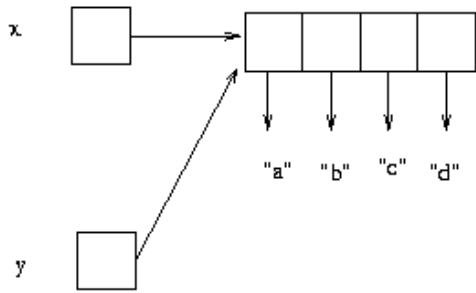


Here is an incomplete method that has two array parameters:

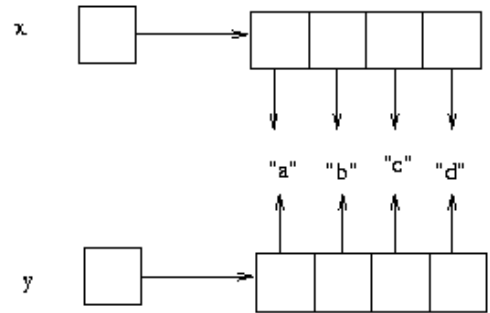
```
public static void change( String [] A1, String [] A2)  
{  
    // your code here  
}
```

For each of the pictures shown on the next page, write code for the body of the `change` method so that calling `change(x, y)` would change arrays `x` and `y` from their initial values as shown above, to the new value shown in the picture. If there is no such code, explain why.

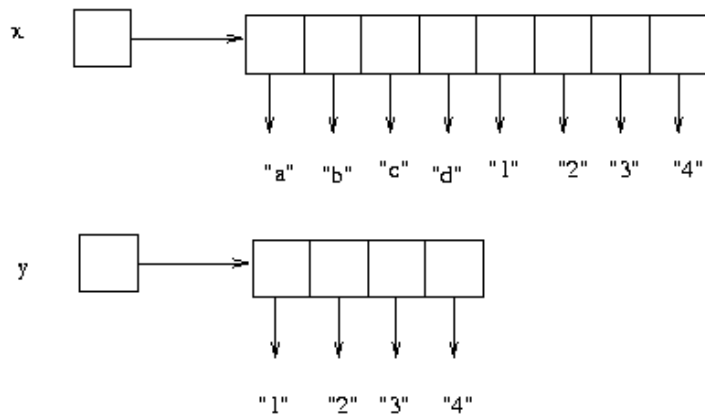
Picture 1



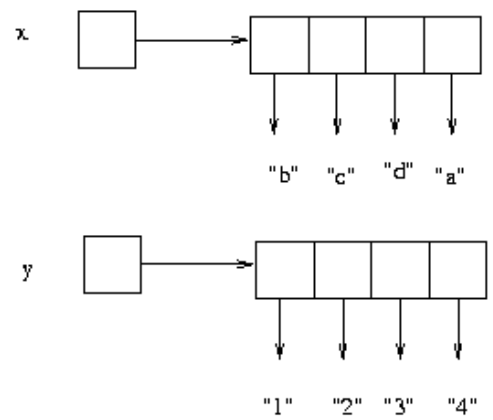
Picture 2



Picture 3



Picture 4



Exercise 3: Array Searching

Using the code from the web page, write a linear search assuming an unordered array, a linear search assuming an ordered array, and finally a binary search (which by definition assumes a sorted array).

Leader -- Print code snippets below on blue paper and the explanations on pink paper. Cut each page into strips. In each case, assume that arr is a non-empty, initialized array of int values. For each blue strip, find a matching pink strip.

```
for (int k=0; k < arr.length/2; k++)
{
    int tmp = arr[k];
    int index = arr.length - (k+1);
    arr[k] = arr[index];
    arr[index] = tmp;
}
```

```
for (int k=0; k < arr.length-1; k++)
{
    if (arr[k] > arr[k+1])
        return false;
}
return true;
```

```
for (int k=0; k < arr.length-1; k++)
{
    for (int j=k+1; j < arr.length; j++)
    {
        if (arr[k] == arr[j])
            return false;
    }
}
return true;
```

```
for (int k=0; k < arr.length-1; k++)
{
    for (int j=k; j < arr.length; j++)
    {
        if (arr[k] == arr[j])
            return false;
    }
}
return true;
```

```
for (int k=1; k < arr.length-1; k++)
{
    if (arr[0] == arr[k])
        return false;
}
return true;
```

```
int tmp = 0;
for (int k=0; k < arr.length; k++)
{
    tmp += arr[k];
}
return tmp > 0;
```

```
for (int k=0; k < arr.length; k++)
{
    if (arr[k] >= N)
        return false;
}
return true;
```

```
int tmp = arr[0];
for (int k=1; k < arr.length; k++)
{
    if (arr[k] > tmp)
        tmp = arr[k];
}
return tmp;
```

```
int tmp = 0;
for (int k=0; k < arr.length; k++)
{
    tmp += arr[k];
}
return (double) tmp/arr.length;
```

```
int tmp = 0;
for (int k=1; k < arr.length; k++)
{
    if (arr[k] == arr[0])
        tmp++;
}
return tmp;
```

The code reverses the order of the values in the array.

The code returns true if the values in the array are in sorted order, from low to high.

The code returns true if the array contains no duplicate values (i.e., no value is stored in more than one place in the array).

The code always returns false.

The code returns true if the value in `arr[0]` doesn't occur in any other place in the array.

The code returns true if the sum of the values in the array is positive

The code returns true if all values in the array are less than N.

The code returns the largest value in the array.

The code returns the average of the values in the array.

The code returns the number of times the value in `arr[0]` occurs in other places in the array, too.

The code returns true if the values in the array are in sorted order, from high to low.

The code always returns true.

The code returns true if all values in the array are positive.

The code returns true if the number of values in the array is less than N.

The code returns the smallest value in the array.

The code checks whether the values in the array are the same forwards and backwards.