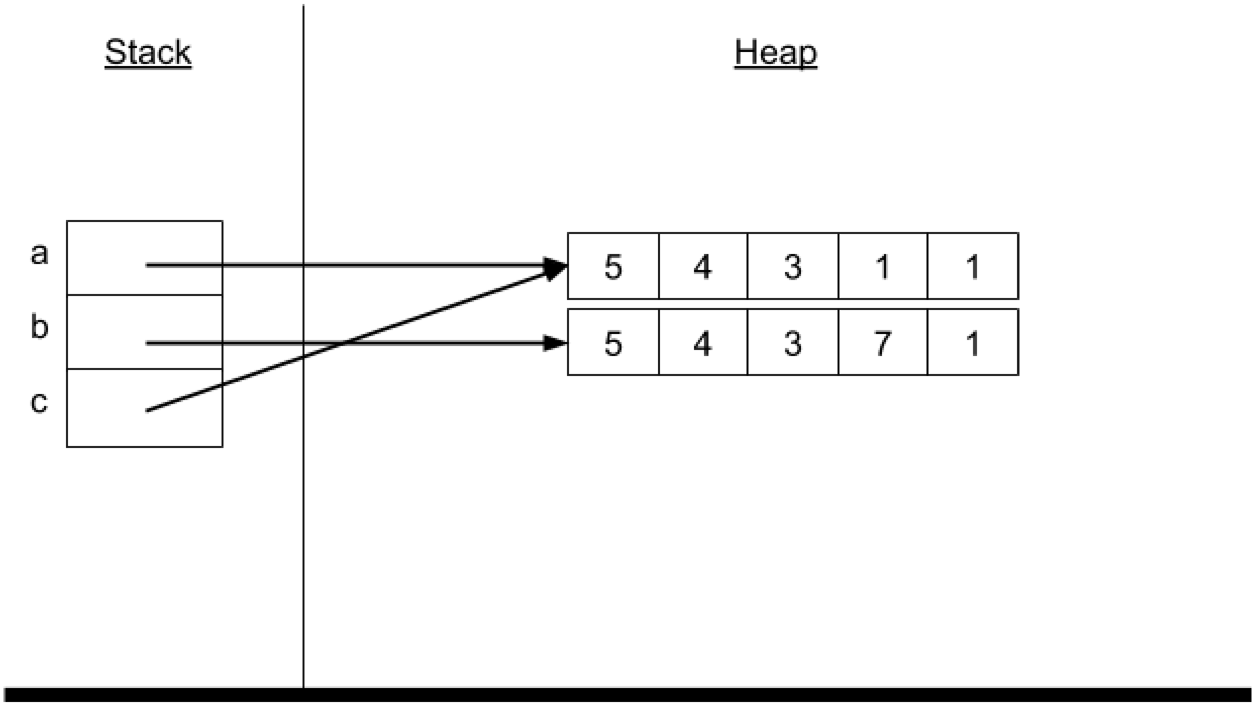
Benjamin (Ben) Kuhn

Problem Set 1, Part I

Problem 1: Memory management and arrays

1-1)



1-2)

The final line of code will print ‘1 7 1’.

Problem 2: Array practice

2-1)

```
public static void shiftRight(int[] arr){
    if(arr == null){
        throw new IllegalArgumentException("Argument to shiftRight cannot be null");
    } else if (arr.length <= 1){
        return;
    } else {
        int previousVal = arr[0];  // save the zeroth element of the array
        int currentVal = 0;
        for(int i = 1; i < arr.length; i++){
            currentVal = arr[i];  // retain the value of the current element of the array
            arr[i] = previousVal;  // set the current element equal to the element one value left
            previousVal = currentVal;  // set up the next iteration of the loop
        }
        arr[0] = currentVal;   // set the zeroth element equal to the last value of the array
    }
}
```

2-2)

```
    public static int indexOf(int[] arr1, int[] arr2){

        for (int i = 0; i < arr2.length - arr1.length + 1; i++){
            for (int j = 0; j < arr1.length; j++){
                if (arr1[j] != arr2[i+j]){  // elements are unequal; move on
                    break;
                }
                if (j == arr1.length - 1){  // all elements in arr1 were equal
                    return i;
                }
            }
        }
        return -1;
    }
```

Problem 3: Recursion and the runtime stack

3-1)

```
mystery(5, 6)
-------------
      a = 5
      b = 6
      myst_rest = mystery(4, 4) = 8
      return 14

      mystery(4, 4)
      -----------------
      a = 4
      b = 4
      myst_rest = mystery(3, 2) = 4
      return 8

         mystery (3, 2)
         -----------------
         a = 3
         b = 2
         myst_rest = mystery(2, 0) = 2
         return 4

             mystery (2, 0)
             ----------------------
            a = 2
            b = 0
            return 2
```

3-2)

mystery(5, 6) returns 14.

3-3)

When the base case is reached, there are 5 frames on the stack, including the main frame.

3-4)

Infinite recursion would occur with any combination of a<0 and odd or negative b. For example, for the call mystery(-1, 3), the base case will never be reached, and a and b will grow more and more negative until stack overflow occurs.

Problem 4: Rewriting a method

4-1)

```
public static boolean search(Object item, Object[] arr) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i].equals(item)) {
            return true;
        }
    }

    return false;
}
```

4-2)

```
public static boolean search(Object item, Object[] arr, int start){
    if (arr[start].equals(item)){
        return true;
    }

    if (start == arr.length - 1){  // stopping condition
        return false;
    }

    return search(item, arr, start+1);
}
```