Department of Computing

ITEC625 Fundamentals of Computer Science
Workshop -

Learning outcomes

This weeks workshop aims at getting some practice with methods that operate on arrays. A template project is provided in `itec625workshop06template.zip`. No test file is provided but there is a client, whose output, when all methods are correctly implemented, should be:

```
90
null
null
100
0
2
2
5
[148, 184, 19, 65, 0]
[70, 80, 60]
```

1. **Method definition**

   Define each of the following methods based on the specifications:

(a)

```java
/**
* @param arr
* @return the last item in the array.
* return null if array is null or empty
*/
public static Integer getLastItem(int[] arr) {









}
```

(b)

```java
/**
* @param arr
* @param except: item to be excluded
* @return sum of all the items in the
* array except item to be excluded
*/
public static int addAllBut(int[] arr, int except) {











}
```

(c)

```java
/**
* @param arr
* @param start: starting index
* @param end: ending index
* assume 0 <= start < arr.length
* assume 0 <= end < arr.length
* assume start <= end
* @return index of the smallest
* item in the index range [start, end]
*/
public static int getMinItemIndex(int[] arr, int start, int end) {









}
```

(d)

```java
/**
* @param a: assume every item occurs once
* @param b: assume every item occurs once
* @param c: assume every item occurs once
* @return number of items that exist
* in all three arrays
*/
public static int countCommonItems(int[] a, int[] b, int[] c) {










}

//HINT: Write a helper method that
//returns true if an array contains
//a given item, false otherwise
```

(e)

```
/**
* @param source
* @param idx: index of item in array
* source (assume 0 <= idx < source.length)
* @param dest
* @return index (in array dest) of
* item at index idx (in array source).
* return -1 if item doesn't exist in dest
*/
public static int vlookup(int[] a, int idx, int[] b) {











}
```

(f)

```
/**
* @param data
* @param nBits, assume nBits.length == data.length
* modify the array data such that each
* item is left shifted by
* corresponding number of bits from
* array nBits
* NOTE: assume each item of nBits is non-negative
*/
public static void leftShift(int[] data, int[] nBits) {







}
```

(g) **(Advanced)**

```java
/**
* @param a: assume every item occurs once
* @param b: assume every item occurs once
* @param c: assume every item occurs once
* @return: array containing items that
* occur in exactly two of the three arrays
*/
public static int[] twoOutOfThree(int[] a, int[] b, int[] c) {








}
```