**DD2459: Software Reliability**

**Lab 2: Black-box and Requirements-Based Testing: Sorting and Searching**

Chenzhou Huang

Robin Ganguly

**Question 1** Draw a condensation graph for Algorithm 2 (Binary Search)

Diagram

Description automatically generated

**Question 2** Write appropriate pre and postconditions using the JML language (i.e., write appropriate requires-ensures conditions) for:

1. Sorting

/\*@

requires arr != null

ensures \forall int i; 0 <= i && i <= arr.length-1; arr[i] <= arr[i+1]

ensures \old(arr.length) == arr.length

@\*/

1. Searching [hint: assume *key* is a native data type e.g., *int* *key* (otherwise must check key is also non-null)]

/\*@

requires arr != null

requires \exists int i; 0 <= i && i <= arr.length-1; key == arr[i] ==>

arr[\result] == key

ensures \forall int i; 0 <= i && i <= \result; key > arr[i] ==>

\result == -1

@\*/

1. Membership

/\*@

requires arr != null

ensures \exists int i; 0 <= i && i <= arr.length-1; key == arr[i] ==>

\result == 1

ensures \forall int i; 0 <= i && i <= arr.length-1; key != arr[i] ==>

\result == 0

@\*/

1. Binary Searching

/\*@

requires arr != null

requires \exists int i; 0 <= i && i <= arr.length-1; key == arr[i]

ensures \result == -1 || arr[\result] == key

ensures \forall int i; 0 <= i && i <= \result; key > arr[i]

ensures \forall int i; \result < i && i <arr.length; key < arr[i]

@\*/

**Question 3** Implement three programs (in your favorite programming language) to perform.

1. Sorting of integer arrays of arbitrary length

Text

Description automatically generated

1. Membership queries on sorted arrays of arbitrary length using binary search.

Graphical user interface, text, application

Description automatically generated

1. Membership queries on unsorted arrays of arbitrary length, by combining program (i) with program (ii).

Text

Description automatically generated