Name: \_\_\_\_\_\_\_\_\_\_\_\_\_key\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_ Quiz review on searching, sorting, and recursion Period: \_\_\_\_

Searches: Linear (sequential) and binary

Sorts: selection, insertion, and mergesort

1. Which sort(s) **do not** use recursion? selection and insertion
2. Which sort looks for the largest value in the array and swaps the largest value with the last value in the array during the first step?

selection

1. Which sort is it if it inserts the last value into position by moving the 2nd to last value to the right if it is larger than the last value?

insertion

1. Which search does not require the array to be sorted to work?

linear

1. Which search is quicker on average?

binary

1. Show the array at each step when doing a selection sort starting at index 0 and looking for the smallest each time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 45 | 22 | 60 | 16 | 2 |
| 2 | 22 | 60 | 16 | 45 |
| 2 | 16 | 60 | 22 | 45 |
| 2 | 16 | 22 | 60 | 45 |
| 2 | 16 | 22 | 45 | 60 |

1. Show the array at each step when doing an insertion sort starting at index 1 and sorting in ascending order.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 45 | 22 | 60 | 16 | 2 |
| 22 | 45 | 60 | 16 | 2 |
| 22 | 45 | 60 | 16 | 2 |
| 16 | 22 | 45 | 60 | 2 |
| 2 | 16 | 22 | 45 | 60 |

1. Show the arrays at each step when doing a mergesort.

{82,7,23,2,55} // array

 82 7 23 | 2 55

82 7 | 23

82 | 7

7 82 | 23

7 23 82 | 2 55

 2 | 55

 2 55

2 7 23 55 82

1. Can a recursive method be written in such a way that it never stops (loops infinitely?)

yes, if we aren’t working toward the base case in our recursion calls.

1. Is the following a recursive method?

no, there is no call to mystery.

 public static int mystery(int n)

 {

 int temp = 0;

 for (int i = 1; i <= n; i++)

 temp = temp + i;

 return temp;

 }

1. Is the following method a recursive method?

yes

public static int mystery(int n)

 {

 if (n == 1) return 1;

 else return n + mystery(n-1);

 }

1. Do questions 10 and 11 give the same result for a call of mystery(3)?

They both return 6

1. Given the following array which search will find the value 3 in the least number of steps?

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 10 | 18 | 22 |

linear

1. Given the following array which search will find the value 18 in the least number of steps?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 10 | 18 | 22 | 35 |

binary

1. Given the following steps to sort an array which sort is this?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 82 | 4 | 16 | 32 | 8 |
| 4  | 82 | 16 | 32 | 8 |
| 4 | 16 | 82 | 32 | 8 |
| 4 | 16 | 32 | 82 | 8 |
| 4 | 8 | 16 | 32 | 82 |

insertion

1. Given the following steps to sort an array which sort is this?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 82 | 4 | 16 | 32 | 8 |
| 4 | 82 | 16 | 32 | 8 |
| 4 | 8 | 16 | 32 | 82 |
| 4 | 8 | 16 | 32 | 82 |
| 4 | 8 | 16 | 32 | 82 |

selection

1. Which sort takes the same time no matter if the array is sorted or not?

selection

1. What is the output from the following method when called with mystery(123)?

 public static int mystery(int n)

 {

 if ((n / 10) == 0) return n;

 else return (mystery(n / 10));

 }

m(123) = m(12) = 1

m(12) = m(1) = 1

m(1) = 1

Returns 1

1. What is the output from the following method when called with mystery(“dog”)?

 public static String mystery(String s)

 {

 if (s.length() == 1) return s;

 else return s.substring(0,1) + mystery(s.substring(1));

 }

m(“dog”) = “d” + m(“og”) = “dog”

m(“og”) = “o” + m(“g”) = “og”

m(“g”) = “g”

returns “dog”

1. What is output from the following method when called with mystery(6)?

 public static int mystery(int n)

 {

 if (n == 1) return 1;

 else return n + mystery(n-1);

 }

m(6) = 6 + m(5) = 6 + 15 = 21

m(5) = 5 + m(4) = 5 + 10 = 15

m(4) = 4 + m(3) = 4 + 6 = 10

m(3) = 3 + m(2) = 3 + 3 = 6

m(2) = 2 + m(1) = 2 + 1 = 3

m(1) = 1

1. What is output from the following method when called with mystery(4)?

public static int mystery(int n)

 {

 if (n == 0) return 1;

 else if (n % 2 == 1) return mystery(n-1);

 else return n + mystery(n-1);

 }

m(4) = 4 + m(3) = 4 + 3 = 7

m(3) = m(2) = 3

m(2) = 2 + m(1) = 2 + 1 = 3

m(1) = m(0) = 1

m(0) = 1

1. What is the output from the following method when called with mystery(6)?

public static int mystery(int n)

 {

 if (n == 1) return 1;

 else return n - mystery(n-1);

 }

m(6) = 6 – m(5) = 6 – 3 = 3

m(5) = 5 – m(4) = 5 – 2 = 3

m(4) = 4 – m(3)= 4 – 2 = 2

m(3) = 3 – m(2) = 3 – 1 = 2

m(2) = 2 – m(1) = 2 – 1 = 1

m(1) = 1