CS 112, A Section Only.
MidtermExam1
65 minutes
Instructor: A.Attarwala
Boston University

Enter your First Name: ______________________________

Enter your Last Name: ______________________________

Enter your Student#: ______________________________

Question:	1	2	3	Total
Points:	21	12	9	42
Score:				

- This is a closed book exam
- No questions, will be taken during the exam. If you have a doubt, state your assumptions and continue writing the exam.
- You will receive a 0.25 bonus mark in total, if you write your first name and your last name and your BU ID on every single page, top right, (front and behind). The maximum you can receive on this exam, including the bonus points is 100%. At no point, your mark will exceed more than 100%.
- **Allocate your time wisely. Do not spend too much time on any single question. You may want to spend 20 minutes roughly per question.**

ONLY TURN THIS PAGE WHEN YOU ARE ASKED TO DO SO. You can use this page as extra sheet if you so require. If you use this page, to write your solution, clearly tell us for which question is this solution for.

ONLY TURN THIS PAGE WHEN YOU ARE ASKED TO DO SO. You can use this page as extra sheet if you so require. If you use this page, to write your solution, clearly tell us for which question is this solution for.

1. We are providing you with a function called `mysterySort`. This function is identical to what I coded up in lecture for one of the sorting algorithms. The number at the start of each line represents the line number. The `mysterySort` function has a total of 17 lines of code. You can safely assume that the code compiles and runs fine. Study the code carefully, before answering the following sub questions:

```
1          public static void mysterySort(int[] items) {
2                  int t = 1;
3                  while (t < items.length) {
4                          int item = items[t];
5                          int k = t - 1;
6                          while (k >= 0) {
7                                  if (items[k] > item) {
8                                          items[k + 1] = items[k];
9                                  } else
10                                         break;
11                                 k--;
12                         }
13                         items[k + 1] = item;
14                         t++;
15                         System.out.println("DEBUG " + Arrays.toString(items));
16                 }
17         }
```

(a) What sorting algorithm is represented by `mysterySort`? Hint: It is either `bubble sort`, or `insertion sort` or `selection sort`. (1)

insertion sort

(b) Provide any two examples of ***local variables*** inside the function `mysterySort`. If there are none, simply state none. Provide a one sentence explanation to support your answer. (2)

t and item

(c) Provide any example of ***local reference*** inside the function `mysterySort`. If there are none, simply state none. Provide a one sentence explanation to support your answer. (1)

items

(d) Provide any example of ***instance variable*** inside the function `mysterySort`. If there are none, simply state none. Provide a one sentence explanation to support your answer. (1)

None

(e) Now assume that the `mysterySort` is called by passing in the array $[1, 2, 3, 4, 5, 6, 7, 8, 9]$ as input. This array is of length 9 and is already sorted in ascending order.

i. You are now asked to trace the `mysterySort`. What does the function `mysterySort` print for the above input array? This question is simply asking you to provide what is printed on the screen (i.e. line #15 of the `mysterySort` function) for the above input array. You can use any blank pages of the exam for tracing and rough work, **however, you must in the space provided below clearly tell us the exact output for each time line #15 is executed.**[1] (3)

```
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
8
```

ii. Again for any input array that is already sorted in ascending order, and n represents the length of the array, how many times does the inner loop execute? If there are multiple correct answers, make sure to select them all. (3)

- ○ $\sum_{j=0}^{n-1} \sum_{i=0}^{99} \sum_{k=0}^{i-1} 1$
- [✓ $\sum_{t=1}^{n-1} \sum_{k=t-1}^{t-1} 1$]
- [✓ $\sum_{t=1}^{n-1} 1$]
- ○ $\sum_{t=1}^{n-1} \sum_{k=0}^{t-1} 1$
- ○ $\sum_{t=1}^{n-1} t$
- ○ None of the above

iii. What is the BigOh i.e. $O(...)$ of the `mysterySort` when the input array is sorted in ascending order? n represents the length of the array. (1)

- ○ $O(1)$
- [✓ $O(n)$]
- ○ $O(n^2)$
- ○ $O(log_2 n)$
- ○ None of the above

[1] Here is how `Arrays.toString` works:

```
int[] items={1,2,3,4,5};
System.out.println("The items of the array are "+Arrays.toString(items));
```

The output of the above println on the screen is:
The items of the array are [1, 2, 3, 4, 5]

(f) Now assume that the `mysterySort` is called by passing in the array $[9, 8, 7, 6, 5, 4, 3, 2, 1]$ as input. This array is of length 9 and is already sorted in descending order.

i. You are now asked to trace the `mysterySort`. What does the function `mysterySort` print for the above input array? This question is simply asking you to provide what is printed on the screen (i.e. line #15 of the `mysterySort` function) for the above input array. You can use any blank pages of the exam for tracing and rough work; **however, you must in the space provided below clearly tell us the exact output for each time line #15 is executed.**.[2] (5)

```
DEBUG [8, 9, 7, 6, 5, 4, 3, 2, 1]
DEBUG [7, 8, 9, 6, 5, 4, 3, 2, 1]
DEBUG [6, 7, 8, 9, 5, 4, 3, 2, 1]
DEBUG [5, 6, 7, 8, 9, 4, 3, 2, 1]
DEBUG [4, 5, 6, 7, 8, 9, 3, 2, 1]
DEBUG [3, 4, 5, 6, 7, 8, 9, 2, 1]
DEBUG [2, 3, 4, 5, 6, 7, 8, 9, 1]
DEBUG [1, 2, 3, 4, 5, 6, 7, 8, 9]
36
```

ii. Again for any input array that is already in descending order, and n represents the length of the array, how many times does the inner loop execute? If there are multiple correct answers, make sure to select them all. (3)

- ○ $\sum_{j=0}^{n-1}\sum_{i=0}^{99}\sum_{k=0}^{i-1} 1$
- ○ $\sum_{t=1}^{n-1}\sum_{k=t-1}^{t-1} 1$
- ○ $\sum_{t=1}^{n-1} 1$
- [✓ $\sum_{t=1}^{n-1}\sum_{k=0}^{t-1} 1$]
- [✓ $\sum_{t=1}^{n-1} t$]
- ○ None of the above

iii. What is the BigOh i.e. $O(...)$ of the `mysterySort` when the input array is sorted in descending order? n represents the length of the array. (1)

- ○ $O(1)$
- ○ $O(n)$
- [✓ $O(n^2)$]
- ○ $O(log_2 n)$
- ○ None of the above

Total for Question 1: 21

[2]Here is how `Arrays.toString` works:
```
int[] items={1,2,3,4,5};
System.out.println("The items of the array are "+Arrays.toString(items));
```
The output of the above println on the screen is:
The items of the array are [1, 2, 3, 4, 5]

2. In the very first lecture of this term, I coded up `BinarySearch`. Here is the exact code from the lecture provided to you. Make sure to study the code very carefully:

```
public static boolean performBinarySearch(int[] array,int targetValue)
{
        int rightIndex=array.length-1;
        int leftIndex=0;
        while(leftIndex<=rightIndex)
        {
                int middleIndex=(rightIndex+leftIndex)/2;
                if (array[middleIndex]==targetValue)
                        return true;
                else if (array[middleIndex]<targetValue)
                        leftIndex=middleIndex+1;
                else
                        rightIndex=middleIndex-1;
        }
        return false;

}
```

(a) What is the BigOh i.e. $O(...)$ of the `performBinarySearch` in the **worst case**? n represents the length of the array. The **best case** for binary search is when the element that you are searching for is in the middle. (2)

- ◯ $O(1)$
- ◯ $O(n)$
- ◯ $O(n^2)$
- [✓ $O(log_2 n)$]
- ◯ None of the above

(b) You are now asked to modify the binary search algorithm in `performBinarySearch` slightly. You are now given the task to find the **first** occurrence index of `targetValue` in `array` using **binary search** that may also contain duplicates. You will return an index in range $[0, n-1]$ if `targetValue` belongs to array or -1 if `targetValue` doesn't belong to array. Here are some examples: (5)

```
int[] items={1,2,3,4,5,6,9,9,9,9};
System.out.println(SortingAlgorithms.findFirstOccurance(items,9)); //must print 6

int[] items1={3,3,3,3,3,3,3,4,5,6};
System.out.println(SortingAlgorithms.findFirstOccurance(items1,3)); //must print 0
```

We have provided some starter code in the function `findFirstOccurance` and ask you to complete the rest that are denoted by the `TODO` comments. Please refer to the starter code on the next page.

```
public static int findFirstOccurance(int[] array,int targetValue)
        {
                int ans = -1;
                int leftIndex=0;
                int rightIndex=array.length-1;

                //TODO: Complete the condition of the while statement
            while (         ) {
                int mid = leftIndex + (rightIndex - leftIndex + 1) / 2;
                int midVal = array[mid];
                if (midVal < targetValue) {
                    //TODO: Complete this block (Approx: 1-2 lines of code)









                }
                else if (midVal > targetValue) {
                    //TODO: Complete this block (Approx: 1-2 lines of code)









                }
                else
                {
                    //TODO: Complete this block (Approx: 1-2 lines of code)







                }
            }

         //TODO: make sure to provide any return statements
         //(Approx: 1-2 lines of code)



        }
```

(c) You are now asked to modify the binary search algorithm in `performBinarySearch` slightly. You are now given the task to find the **last** occurrence index of `targetValue` in `array` using **binary search** that may also contain duplicates. You will return an index in range $[0, n-1]$ if `targetValue` belongs to array or -1 if `targetValue` doesn't belong to array. (5)

Here are some examples:

```
int[] items={1,2,3,4,5,6,9,9,9,9};
System.out.println(SortingAlgorithms.findLastOccurance(items,9)); //must print 9

int[] items1={3,3,3,3,3,3,3,4,5,6};
System.out.println(SortingAlgorithms.findLastOccurance(items1,3)); //must print 6
```

We have provided some starter code in the function `findLastOccurance` and ask you to complete the rest that are denoted by the `TODO` comments. Please refer to the starter code on the next page.

```
public static int findLastOccurance(int[] array,int targetValue)
        {
                int ans = -1;
                int leftIndex=0;
                int rightIndex=array.length-1;

                //TODO: Complete the condition of the while statement
            while (         ) {
                int mid = leftIndex + (rightIndex - leftIndex + 1) / 2;
                int midVal = array[mid];
                if (midVal < targetValue) {
                    //TODO: Complete this block (Approx: 1-2 lines of code)









                }
                else if (midVal > targetValue) {
                    //TODO: Complete this block (Approx: 1-2 lines of code)









                }
                else
                {
                    //TODO: Complete this block (Approx: 1-2 lines of code)







                }
            }

         //TODO: make sure to provide any return statements
         //(Approx: 1-2 lines of code)



        }
```

Total for Question 2: 12

3. Java Memory Model

   Draw two separate Java memory model for the following two subparts. I want you to draw what the stack and the heap looks like after line #8 has been executed. DO NOT SHOW ME HOW THE STACK AND THE HEAP CHANGES FROM LINE#4 TO LINE#8. JUST SHOW ME THE FINAL PICTURE AFTER LINE #8 HAS BEEN EXECUTED. You can assume that line #9 has not been executed. (**Hint:** `int` is a primitive type AND `Integer` is an object. All objects go on the heap and all local variables go on the appropriate stack frame).

   (a) [3] (5)

```
1   class JavaMemoryModel {
2       public static void main(String[] args)
3       {
4           String s = new String("Wombat");
5           Integer i = new Integer(27);
6           Integer j = new Integer(29);
7           Integer k = new Integer(31);
8           Integer[] v={i,j,k};
9           String x="Do not draw the memory model from this line onwards";
10      }
11  }
```

[3]Make sure that inside the stack region of the memory diagram, the only stack frame on the stack is the `main` stack frame.

(b) [4] (4)

```
class JavaMemoryModel {
    public static void main(String[] args)
    {
        String s = new String("Wombat");
        int i = 27;
        int j = 29;
        int k = 31;
        int[] v={i,j,k};
        String x="Do not draw the memory model from this line onwards";
    }
}
```

Total for Question 3: 9

[4] Make sure that inside the stack region of the memory diagram, the only stack frame on the stack is the `main` stack frame.

Extra Sheet: If you use this page, to write your solution, clearly tell us for which question is this solution for.