Student Name: Kunlaya Kobunnoi Set: 3D

### The goals of this lab are to:

* give you practice evaluating code to determine its efficiency class
* give you practice solving problems, and writing pseudocode for your solutions

Due date:

* The questions in this lab are due by the end of today (**11:30 pm**). You must submit your answers to the D2L”lab-2” prior to the due date.
* Please type your answers with blue color font.
* Late submissions will not be marked.
* Don’t email the labs, it will not be marked.

Questions:

Answer the following 6 questions, showing all your work. Write your answers on these lab sheets so that you can submit it to D2L. In each case, the “worst case efficiency” should be given as a big-Oh class.

1. What is the worst case efficiency of the following code? [1 mark]

**ArrayList c;**

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

**c.add ( new Integer(i) );**

*Hint: ArrayList.Add is O(1) when adding at the end of the list, O(n) otherwise*

*= n-1+1*

*= n*

2. What is the worst case efficiency of the following code? [1 mark]

**int sum = 0;**

**int[]a = new int[n];**

**int[]b = new int[10];**

**for (int i= 0; i<n; i++)**

**for (int j = 0; j< 10; j++)**

**sum += a[i] / b[j];**

3. What is the worst case efficiency of the following code? [1 mark]

**void f ( int[] a )**

**{**

**Arrays.sort ( a );**

**for (int i = 0; i<n; i++)**

**a[n-i-1] = 3 \* i -2;**

**for (int i = 0; i<n; i++)**

**System.out.println ( a[i] );**

**return a;**

**}**

*Note: you should assume that Arrays.sort() is O(nlogn)*

= +((n-1)0+1) + ((n-1)-0+1)

= +n-1 + n-1

=the worst case is n(log n)

4. Consider the algorithm to the right and answer the following questions. [2 mark]

a. Explain what this algorithm does.

1. algorithm abc( A[0..n-1] )

2. bottom ← 0; top ← n-1

3. swapped ← true

4. while swapped is true do

5. swapped ← false

6. for i ← bottom to top-1 do

7. if A[i] > A[i+1]

8. swap A[i] and A[i+1]

9. swapped ← true

10. // end for loop

11. // end while loop

12. // end algorithm

Sorted array from the largest number to the smallest number.

b. What is the basic operation in this program, and on which line does this operation occur?

A comparison key on line 7

c. What is the best case efficiency class (ie: the best big-oh class) for the algorithm?

When array a is already sorted, line 7 is executed n-1 times. Therefore, the best case efficiency is O(n).

d. What is the worst case efficiency class (ie: the best big-oh class) for the algorithm?

When the array is sorted reversely, the worst case efficiency is O()

5. Consider the following problem, and answer the questions that follow. [5 mark]

Minesweeper

Have you ever played Minesweeper? It's a cute little game which comes within a certain Operating System. The goal of the game is to find where are all the mines within a **M**x**N** field. To help you, the game shows a number in a square which tells you how many mines there are adjacent to that square. For instance, suppose the following 4x4 field with 2 mines (which are represented by an \* character):

\*...

....

.\*..

....

If we would represent the same field placing the hint numbers described above, we would end up with:

\*100

2210

1\*10

1110

As you may have already noticed, each square may have at most 8 adjacent squares.

*Input*

The input will consist of an arbitrary number of fields. The first line of each field contains two integers **n** and **m** (0 < **n,m** <= 100) which stands for the number of lines and columns of the field respectively. The next **n** lines contains exactly **m** characters and represent the field. Each safe square is represented by an "." character (without the quotes) and each mine square is represented by an "\*" character (also without the quotes). The first field line where **n=m=0** represents the end of input.

*Output*

The output for each mine field consists of the **n** input lines with the "." characters replaced by the number of adjacent mines to that square.

*Sample Input*

4 4the

\*...

....

.\*..

....

3 5

\*\*...

.....

.\*...

0 0

*Sample Output*

\*100

2210

1\*10

1110

\*\*100

33200

1\*100

a) Write pseudo-code for your solution.

1. Algo (matrix[0-n-1,0…m-1]

2. For i = 0 TO n-1{

3. For j = 0 TO m-2 {

4. IF matrix[i][j] = ‘\*’ THEN

5. print matrix[i][j];

6. ELSE {

7. Sum = 0;

8. For k = i-1 TO i+1{

9. For l = j-1 TO j+1{

10. If k >= 0 and k < n and l >= 0 and l < m and matrix[k][l] = ‘\*’ THEN

11. sum = sum + 1;

12. Print sum;

13. }

14. } // END FOR

15. Print newline;

16. } // END FOR

17.} // END Function

b) Analyze your solution and determine it's big-oh efficiency class for an n by n matrix (for the processing of a single minefield).

=