Student Name: Set:

### 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 the lab. You must submit your answers to the D2L”lab-2” prior to the end of the lab.
* Please type your answers with blue color font.

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*

O(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];**

O(n)

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)*

O(n)

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

a. Explain what this algorithm does.

Bubble sort

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

2. bottom ← 1; top ← n

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

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

Comparison, on line 7

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

O(n)

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

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. There must be an empty line between mine field outputs.

*Sample Input*

4 4

\*...

....

.\*..

....

3 5

\*\*...

.....

.\*...

0 0

*Sample Output*

\*100

2210

1\*10

1110

\*\*100

33200

1\*100

a) Write pseudo-code for your solution.

ALGORITHM Minesweeper(n, m, A[0..n-1][0..m-1])

1. B[n][m]🡨0
2. for i🡨0 to n-1 do
3. ……for j🡨0 to m-1 do
4. ………..if A[i][j] = \*
5. …………..increase(A,B,I,j)
6. Increase(A,B,I,j)
7. For r🡨i-1 to r🡨i+1
8. …for c🡨j-1 to j+1
9. ……if isNotOverBound(I,j,n,m) and A[r][c] != \*
10. ………B[r][c] = B[r][c]+1
11. isNotOvetBound(I,j,m,n)
12. ….if j <0 or j > m or I < 0 or I > n
13. …….return true
14. ….return false

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).