Student Name: Dongsun Kim Set: O

### 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*

Basic operation: c.add()

Worst efficiency count:

n ∈ 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];

Basic operation: Addition in the innermost loop

Worst efficiency count:

10n ∈ 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)*

Parameter for input size: n, the size of array

Basic operation: Arrays.sort()

Worst efficiency count:

O(nlogn)

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

a. Explain what this algorithm does.

Sorting the array from smallest element to biggest element.

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?

Basic operation: Comparison on line 7

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

n-1 ∈ O(n)

Check -> (no swap) -> Check -> (no swap) -> ... : n times

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

n-1 ∈ O(n)

Check -> swap -> Check -> swap -> ... : times

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 has 8 adjacent squares.

*Input*

The input is matrix that represent a 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).

*Output*

The output for a field matrix is a matrix that all "." characters is replaced by the number of adjacent mines.

*Sample Input*

\*...

....

.\*..

....

*Sample Output*

\*100

2210

1\*10

1110

a) Design an algorithm that receives an n by n field matrix and generates the output matrix.

void MineSweeper(A[0...n-1][0...n-1])

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

for (int j = 0; j < n; j++) {

if (A[i][j] = ".") {

A[i][j] = 0

for (int x = -1; x <= 1; x++) {

if (i+x < 0 || i+x > n)

continue;

for (int y = -1; y <=1; y++) {

if (j+y < 0 || j+y > n || A[i+x][j+y] = A[i][j])

continue;

if (A[i+x][j+y] = "\*")

A[i][j]++;

}

}

System.out.println(A[i][j]);

} else {

System.out.println(A[i][j]);

}

System.out.println("\n");

}

b) Analyze your solution and determine it's big-oh efficiency class for an n by n input matrix.

Parameter for input size: n, the size of input matrix

Basic operation: comparison on the line of *"if (j+y < 0 || j+y > n || A[i+x][j+y] = A[i][j])"*

Worst efficiency count:

= = = =

∈ O(