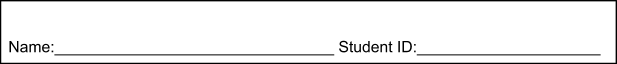
# **CMPSCI 187 Midterm #1 Fall 2018**



# **Short Answer (10 x 4 = 40 points)**

1. **Class** ​Triangle​ **implements the** ​Geometry​ **interface and class** ​RightTriangle​ **extends** ​Triangle​**. Consider compiling each of these:** 
   1. 1pt)​ Geometry g = new Geometry();
   2. 1pt)​ Geometry g = new Triangle();
   3. 2pt)​ Geometry g = new RightTriangle();

**If it does compile, write True, otherwise False in the correct box.**

|  |  |  |
| --- | --- | --- |
| a) false | b) true | c) true |

1. ​**Assume** ​LLNode<T>​ **is a generic, linked list node class defined as follows:**

​public class LLNode<T> { public T info; public LLNode<T> link; public LLNode(T in) { info = in; link = null; }

}

**What’s the output of each** ​**System.out.println**​ **below?**

LLNode<String> x = new LLNode<String>(new String("CS187"));

LLNode<String> y = new LLNode<String>(new String("CS187"));

​**System.out.print(x.info == y.info); // Line a)**

**System.out.print(x.link == y.link); // Line b)**

|  |  |
| --- | --- |
| a) true ! false | b) true |

1. **Suppose, you are given three linked lists, A, B, C where:**

head\_A->10->45->23->9->null head\_B->9->3->2->91->null head\_C->7->5->29->null

And you call: randomFunction(head\_A,head\_B,head\_C) ​ defined as:​

public static void randomFunction(LLNode<int> A, LLNode<int>B, LLNode<int> C) { LLNode<int> temp = A; while(temp.link!=null){ temp = temp.link;

} temp.link = B; temp = B; while(temp.link!=null){ temp = temp.link;

}

temp.link = C;

}

How many nodes are in the linked list starting with head\_A?

|  |
| --- |
| 11? |

**4. What does this code output?**

public class DeepNetworks extends ArtificialIntelligence { public void start() {

System.out.println("Finding intelligence");

} } public class ArtificialIntelligence { public void start() {

System.out.println("Learning started");

}

public static void main(String args[]){

ArtificialIntelligence alchemy = new DeepNetworks(); alchemy.start();

}

}

|  |
| --- |
| Finding Intelligence |

**5. What is the equivalent infix expression for this postfix expression?**

**A B + C D + \***

|  |
| --- |
| A+(B+C\*D) ! |

**6. Complexity**

**a 1pt)** ​**What’s the Big-O cost of the insertTail method of a linked list of size n if we don’t have a tail**​  **pointer:**

# O(N )

**b 1pt)** ​**What’s the Big-O cost of the insertTail method of a linked list of size n if we** ​ **do**​  **have a tail**​  **pointer:**

# O( 1 ​ )​

**c 1pt)** ​**What’s the Big-O cost of the method below?**​

for(int i = N; i > 0; i = i/3){

System.out.println(“ ‘i’ is currently: “ + i);

}

# O( log(n) )

**d 1pt)** ​**What’s the Big-O cost of the method below?**​

int i = N; while(i >= N/3){ for(int j=1; j<N/2; j\*=5) { array[i] = j;

} i--;

}

# O( nlog(n) )

1. **What’s the output of the following code? (assume a pop() that returns the object)**

LinkedStack<Integer> s = new LinkedStack<Integer>();

s.push(40);

s.push(0);

s.push(90);

System.out.print(s.peek() + " " + s.pop() + " " + s.pop() + " ");

|  |
| --- |
|  |

1. **What does the following function output for the given Linked List?**

head->10->45->23->9->null

public void foo (LLNode<T> node) { if (node == NULL) return;

foo(node.next);

System.out.printf("%d ", node.data);

}

|  |
| --- |
| 9,23,45,10 |

1. **In the sets project, say you did the following, what would the output be?**

Set<String> set0;

Set<String> set1;

Set<String> set2;

set0 =new linkedSet<String>(); set1 = set0.adjoin(“foo”); set2 = set1.adjoin(“bar”); set0 = null; set1 = null;

System.out.println(set2.contains(“foo”));

|  |  |
| --- | --- |
|  |  |

1. **What does this output?**

class Point { public int x; public int y;

}

public class test {

public static void bar(int b) { b= 10; } public static void foo(Point a) { bar(a.x); }

public static void main(String[] args) { Point one = new Point(); one.x = 1; one.y = 2; foo(one);

System.*out*​ ​.println(one.x);

}

}

|  |
| --- |
|  |

## 11) Matrix (30 points)

class myMatrix<T> { private T[][] matrix; int rows, int columns;

public myMatrix(int rows, int columns) {

matrix = (T[][]) new Object[rows][columns]; this.rows = rows; this.columns = columns;

}

}

myMatrix is for saving data in a matrix. Matrix is a 2 dimensional array. Data is stored in the private member *matrix*​ (in class myMatrix) which has some number of rows and columns. You can also see it as array of arrays. Everytime you want to access the element in row i and column j, you can just use matrix[i][j].

**a) (15 points)**​ Now we want to flip over its main diagonal. It’s also called transpose of matrix. Now given a​ matrix, set the matrix to its transpose. For example:​

| a00 a10 a20 |

| a00 a01 a02 a03 | transpose | a01 a11 a21 |

| a10 a11 a12 a13 | ---------------------> | a02 a12 a22 |

| a20 a21 a22 a23 | | a03 a13 a23 |

One more rule: you cannot use any data structures other than arrays in this question. Your solution should be no worse than O(M\*N), where M is the number of rows and N is the number of columns.

public void transpose(){

T[][] newmatrix = (T[][]) new Object[columns][rows];

int temp = columns; columns = rows; rows = temp; matrix = newmatrix; return;

}

**b) (15 points)** ​Now we want to check whether the matrix is Toeplitz. A matrix is Toeplitz if every diagonal from top-left to bottom-right has the same element. Now given an M x N matrix, return True if and only if the matrix is Toeplitz. The value of all elements in the matrix are positive numbers.

For example:

| 1 0 6 | | 1 6 0 3 | | 2 1 0 |

| 2 1 6 0 | is ​Toeplitz. | 4 2 1 | is not ​ ​Toeplitz since the element (3,2) is different to the element (1,0) and (2,1).

| 5 2 1 6 | | 5 4 3 |

One more rule: you cannot use any data structures other than the given matrix in this question. Your solution should be no worse than O(M\*N), where M is the number of rows and N is the number of columns.

public boolean isToeplitz(){

}

## 12) Linked Lists (30 points)

**a) (15 points)**​ Write an instance method for the LinkedList<T> class that returns the number of times, if any, that the element x occurs in the calling list. Your method should leave the list unchanged.

public class LLNode<T> { public T data; public LLNode<T> next;

public LLNode(T data) { this.data = data;}

}

public class LinkedList<T>{

LLNode<T> head;

/\* DO NOT ADD ANY INSTANCE VARIABLES, You can make local variables. \*/

public int countEquals(T x){

}

}

**b) (15 points)**​ Write the duplicateSortedLinkedList<T> method that clone each node in a sorted LinkedList and keep the LinkedList sorted all the time.

For example: the linked list [1] -> [2] -> [3] -> null will become [1] -> [1] -> [2] -> [2] -> [3] -> [3] after using duplicateSortedLinkedList.

One more rule: you cannot use any data structures other than the LinkedList n this question. Your solution should be no worse than O(N) where N is the number of node in the LinkedList.

public void duplicateSortedLinkedList(LLNode<T> node) {

/\* DO NOT ADD ANY INSTANCE VARIABLES, You can make local variables. \*/

}

​***Elegance counts***

Scratch paper, nothing will be graded on this page!!!!