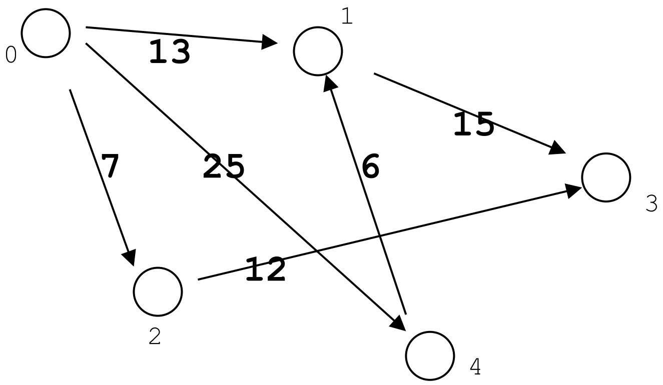
SAMPLE EXAM #1 GRAPHs, LISTs & BSTs

//==

1) Convert this graph diagram into:



a) A Matrix representation

b) A "sparse format" Adjacency List

2) Convert this sparse format input file into:

```
3
0  1  8
0  2  10
1  0  5
2  1  9
```

a) a Matrix

b) a graph diagram

3) Here is a Matrix representing a graph:

	0	1	2	3	4	5	6	7
0	0	-1	-1	4	-1	-1	-1	8
1	-1	0	-1	5	-1	-1	-1	-1
2	-1	-1	0	-1	-1	5	-1	3
3	-1	-1	4	0	-1	10	-1	3
4	-1	-1	-1	-1	0	-1	-1	-1
5	-1	4	-1	-1	-1	0	-1	-1
6	-1	-1	-1	-1	-1	2	0	-1
7	-1	-1	-1	-1	3	-1	-1	0

a) inDeg of Node 4 = __________

b) outDeg of Node 4 = _________

c) maxInDeg of entire graph = ________

4) For 4a – 4e I'll give you a diagram of a linked list where each element (ListElement) is an int and a next pointer. For each exercise I'll give the intial list and some code that executes on that list. You will redraw the list after the code executes to show the new list state -OR- you will write the error that occurs if that code does something illegal like referencing a null pointer or a bogus assignment. If I declare other pointers you MUST draw them too and indicate what they points to.

** You are to assume the garbage collector erased any garbage created by code DON'T DRAW ANY ORPHANED ELEMENTS

4a)   head -> [1,-]-> [3,-]-> [5,-]-> [7,null] // INITAL LIST

```
// CODE EXECUTED ON LIST

ListElement curr = head;
curr.setNext( curr.getNext().getNext() );
```

REDRAW LIST, CURR & HEAD

4b)   head -> [5,-]-> [2,-]-> [1,-]-> [1,null]

```
// CODE EXECUTED ON LIST

ListElement curr=head;
head=null;
while (curr.getNext()!=null)
curr=curr.getNext();
head = curr;
```

REDRAW LIST, CURR & HEAD

5) SAME DRILL AS ABOVE EXCEPT NOW YOU WRITE THE OUTPUT

```
5a)   ListElement head;
      head= new ListElement(3,new ListElement(1,
           new ListElement(2, new ListElement(0,null))));

// CODE EXECUTED ON LIST

for (ListElement curr=head ; curr!=null ; curr=curr.getNext() )
System.out.print( curr.getData() + " " );

// PRINT THE OUTPUT
```

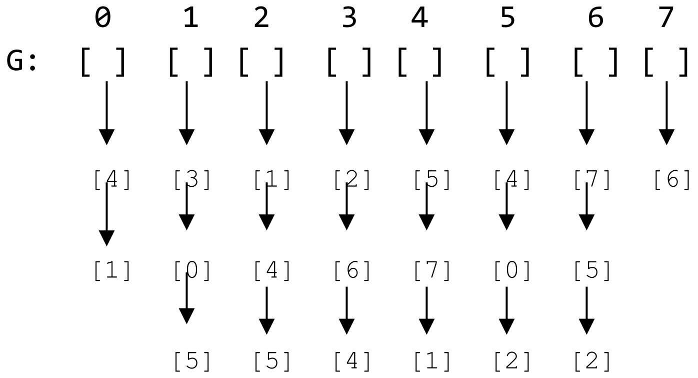
```
5b)   int arr[] = { 9,8,7,6,5,4,3,2,1 };
      ListElement head;

// CODE EXECUTED ON LIST

for (int i=0 i< a.length ; ++i)
head = new ListElement( arr[i], head );
for (ListElement curr=head ; curr!=null ; curr=curr.getNext())
System.out.println( curr.getData() + " " );

// PRINT THE OUTPUT
```

6) Here is an adjacency list (linked list) graph:



a) inDeg of Node 4 = ____________

b) outDeg of Node 4 = ____________

c) maxInDeg of entire graph = ____________

#7 Tree Insertions and traversals

Here is the input file from left to right:

100 90 200 80 95 150 225 70 85 97 140 175

7-a) Draw the resulting Binary Search Tree assuming the above values were read in from left to right and inserted in that order:

7-b) List the nodes in preOrder: put them all on same line with a space between:

7-c) List the nodes in InOrder: put them all on same line with a space between:

7-c) List the nodes in postOrder: put them all on same line with a space between:

#8 More Trees

Here is code snippet that inserts some numbers into a Binary Search Tree:

Assume each node is just an *int*, and a *next* pointer

```
// ASSUME ADD DOES NOT KEEP TREE BALANCED
public static void main( String args[] )
{
     Tree t = new Tree();
     for (int i=0 ; i < 8 ; ++i)
          t.add( i );  // ASSUME ADD DOES NOT REBALANCE
}
```

8-b) Draw the above Tree

8-b) How many levels does the above tree have?