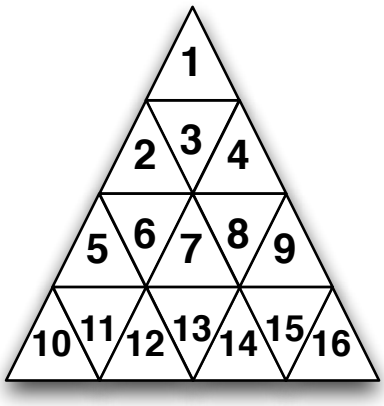
PROBLEM 1 : (*TriPyramid (20 points)*)

The picture below shows a four-rowed, two-dimensional pyramid constructed of triangles – this will be called a *4-pyramid* in this problem because it has four rows. The top triangle is number one, then the triangles are numbered left-to-right in a row and top-to-bottom as shown. In the fourth row there are seven triangles; in general in the N^{th} row there are $2N-1$ triangles. In answering questions below assume we're discussing an N-pyramid with N rows for a large value of N.



Part A (2 points)
What is the *exact* value or number of the right-most triangle in the seventh row?

Part B (2 points)
What is the *exact* value of the right-most triangle in the 40^{th} row?

Part C (2 points)
What is the *exact* value of the left-most triangle in the 61^{st} row?

Part D (2 points)

Using big-Oh what is the number of triangles in the bottom row of a pyramid with N rows. Justify your answer.

Part E (2 points)

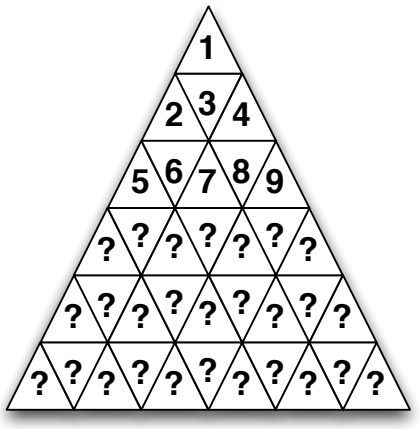
Using big-Oh what is the number of triangles in the bottom row of a pyramid with N^2 total triangles. Justify your answer.

Part F (2 points)

Using big-Oh what is the number of triangles in the bottom row of a pyramid with N^4 total triangles. Justify your answer.

Part G (8 points)

The diagram below shows four 3-pyramids combined to make an 11-pyramid. Assume there is a function or method *combine* that takes an N-pyramid as a parameter and returns a new pyramid created by combining four N-pyramids as shown. For example, the pyramid shown below would be returned by the call `combine(3)`.



G.1 (2 points)

What is the big-Oh number of triangles in the bottom row of the pyramid returned by the call `combine(N)`. Justify your answer.

G.2 (2 points)

What is the big-Oh value of the rightmost pyramid in the bottom row (the triangle with the largest number) in the pyramid returned by the call `combine(combine(N))`. Justify your answer.

G.3 (4 points)

Consider the pseudo-code below for a sequence of calls to create a pyramid. For example, when $N = 2$ the initial pyramid `p` is constructed before the loop with two rows and three triangles in the bottom row; the loop then executes twice. The first time through the loop results in `p` having four rows. The second time through the loop results in `p` having eight rows (with 15 triangles in the bottom row and the value printed is 64).

```
Pyramid p = new Pyramid(N); // create pyramid with N rows

int size = p.rows();  // set size to N
for(int k=0; k < size; k++){
    p = combine(p);
}
System.out.println("biggest number is "+p.lastPyramid());
```

If the value of $N = 10$ so that the initial pyramid has 10 rows in which the last pyramid is numbered 100 what is the value printed by the code above? Justify your answer. Your answer should be exact, but can be expressed using exponentiation and multiplication, e.g., $3^5 \times 100^2$ is acceptable as an answer.

PROBLEM 2 : (*Linkin (18 points)*)

The method `firstToLast` below moves the first element in a list so that it's the last element in the list.

```
public List<String> firstToLast(List<String> list){
    String first = list.remove(0);
    list.add(first);
    return list;
}
```

The timings below result from calling `firstToLast` with both a `LinkedList<String>` and an `ArrayList<String>` with the number of elements (size of the list) indicated in the left-most column. These timings are relatively consistent in their pattern across different versions of Java on different machines, e.g., running on a Mac, Linux, or Windows machine.

Times are in seconds for a `LinkedList<String>` and an `ArrayList<String>`.

size (10^3)	**link**	**array**
10	0.031	1.617
20	0.086	6.442
30	0.125	14.180
40	0.160	
50	0.199	
60	0.219	
70	0.232	
80	0.282	
90	0.358	
100	0.329	

The code for timing is below:

```
String first = list.get(0);
double start = System.currentTimeMillis();
for(int j=0; j < 20*list.size(); j++){
    list = firstToLast(list);
}
double end = System.currentTimeMillis();
System.out.printf("%d\t%1.3f\n",list.size(),(end-start)/1000.0);
if (! first.equals(list.get(0))){
    System.out.printf("first failure at %d\n",list.size());
}
```

Part A (3 points)

Expain in a sentence of two the timing results for the column labeled **link**. You should explain the pattern of the results, not the particular running times.

Part B (6 points)

Project a time for the column labeled **array** for a 40,000 element list and a 100,000 element list. Justify both answers.

(continued)