CSC 231 HOMEWORK 4
GROUP PROJECT 1

PROFESSOR GODFREY MUGANDA

The goal of this project is to understand the recursive formulas for counting combinatorial objects to the extent that you can adapt those recursive strategies to actually generate those objects.

You will work in groups of 2 or 3s. Make sure that the names of all group members appear as authors in the code. When finished, appoint a group leader who will submit the assignment by attachment to email, with a cc to the other members of the group.

This is due Saturday at the end of Week 5.

1. Partitions of a set

Use your understanding of the recursive method for counting partitions of a set of size n that has k parts to generate a list of all such partitions.

Do this by writing a method

```
    /**
     * returns list of  all partitions of a set 1 .. n
     * that have k parts
     * @param n
     * @param k
     * @return
     */
    static LinkedList<Partition> partitionsList(int n, int k)
```

Here we are declaring a class

```
class Partition extends LinkedList<TreeSet>
{

}
```

to make the code a little bit more readable.

Test your work using this main method:

```
public static void main(String[] args)
    {
        Scanner sc = new  Scanner(System.in);
        int n, k;

        while (true)
        {
            System.out.print("Enter n and k with n >= k: ");
```

```
            n = sc.nextInt();
            k = sc.nextInt();

            if (n <= 0 || k > n || k == 0) return;

            List<Partition> partitions = partitionsList(n, k);
            partitions.forEach(System.out::println);
        }
    }
```

Here is a sample run.

```
Enter n and k with n >= k: 5 1
[[1, 2, 3, 4, 5]]
Enter n and k with n >= k: 5 5
[[1], [2], [3], [4], [5]]
Enter n and k with n >= k: 5 2
[[1, 2, 3, 4], [5]]
[[1, 2, 3, 5], [4]]
[[1, 2, 3], [4, 5]]
[[1, 2, 4, 5], [3]]
[[1, 2, 4], [3, 5]]
[[1, 2, 5], [3, 4]]
[[1, 2], [3, 4, 5]]
[[1, 3, 4, 5], [2]]
[[1, 3, 4], [2, 5]]
[[1, 3, 5], [2, 4]]
[[1, 3], [2, 4, 5]]
[[1, 4, 5], [2, 3]]
[[1, 4], [2, 3, 5]]
[[1, 5], [2, 3, 4]]
[[1], [2, 3, 4, 5]]
Enter n and k with n >= k: 0 0
```

2. Number Partitions

Problem 47 on page 359 of the class textbook defines the concept of a *partition* of a number and gives a recursive formula for determining the number of such partitions.

Build on your understanding of that formula to generate a list of such partitions. Write a program that computes the number of such partitions according to the maximum of a summand in such a partition.

You will need to view each number partition as a list of integers. Write a method

```
/**
 * Returns a list of a list of integers  e_1, ,,, e_m
 * such that each e_i <= k and all the e_i s add up to n.
 * @param n
 * @param k
 * @return
 */
static List<List<Integer>> numPartitions(int n, int k)
```

that returns the list of all such partitions.

Test your work with the following main method:

```
public static void main(String[] args)
   {
       Scanner sc = new  Scanner(System.in);
       int n, k;

       while (true)
       {
           System.out.print("Enter n and k with n >= k: ");
           n = sc.nextInt();
           k = sc.nextInt();

           if (n <= 0 || k > n || k == 0) return;

           List<List<Integer>> partitions = numPartitions(n, k);
           partitions.forEach(System.out::println);
       }
   }
```

Here is a sample run:

```
Enter n and k with n >= k: 5 1
[1, 1, 1, 1, 1]
Enter n and k with n >= k: 5 2
[1, 2, 2]
[1, 1, 1, 2]
[1, 1, 1, 1, 1]
Enter n and k with n >= k: 5 3
[1, 1, 3]
[2, 3]
[1, 2, 2]
[1, 1, 1, 2]
[1, 1, 1, 1, 1]
Enter n and k with n >= k: 5 4
[1, 4]
[1, 1, 3]
[2, 3]
[1, 2, 2]
[1, 1, 1, 2]
[1, 1, 1, 1, 1]
Enter n and k with n >= k: 5 5
[1, 4]
[1, 1, 3]
[2, 3]
[1, 2, 2]
[1, 1, 1, 2]
[1, 1, 1, 1, 1]
[5]
Enter n and k with n >= k: 0 0
```