SK_CODING_CHALLENGE_3 – 23/06/2020

Activity 1:

Find the number of ways that a given integer,X, can be expressed as the sum of the n th powers of unique, natural numbers.

Input Format
The first line contains an integer X.
The second line contains an integer N.

Constraints
X = {1 to 1000}
N = {2 to 10}

Output Format
Output a single integer, the number of possible combinations caclulated.

Solution:

```
public class ThePowerSum
{
	private static int solve(int x, int n, int num, double sum)
	{
		if (sum == x)
		{
			return 1;
		}
		else
		{
			int ans = 0;
			if (sum < x)
			{
				for (int i = num; i <= Math.pow(x, 1.0 / n); i++)
				{
```

```
                                        ans += solve(x, n, i + 1, sum + Math.pow(i, n));
                                    }
                                }
                                return ans;
                            }
                        }
                        public static void main(String[] args) throws FileNotFoundException
                        {
                            System.setIn(new FileInputStream(System.getProperty("user.home") + "/" + "in.txt"));
                            Scanner in = new Scanner(System.in);
                            int x = in.nextInt();
                            int n = in.nextInt();
                            System.out.println(solve(x, n, 1, 0));
                        }
                    }
```

****

Activity 2:

Given an integer, and find the super digit of the integer.
If x has only 1 digit, then its super digit is x.
Otherwise, the super digit of x is equal to the super digit of the sum of the digits of x.
You are given two numbers n and k. The number p is created by concatenating the string n*k times.
The superDigit has the following parameter(s):
n: a string representation of an integer
k: an integer, the times to concatenate n to make p

Input Format:
The first line contains two space separated integers,n and p.

Output Format:
Return the super digit of p.

Solution:

```
public class RecursiveDigitSum {
   private static BigInteger sum(BigInteger p) {
      BigInteger ans = BigInteger.ZERO;

      String num = p.toString();
      for (int i = 0; i < num.length(); i++) {
         ans = ans.add(new BigInteger("" + num.charAt(i)));
      }

      return ans;
   }

   private static BigInteger solve(BigInteger p, int k) {
      BigInteger sum = sum(p);
      BigInteger ksum = sum.multiply(new BigInteger("" + k));
      BigInteger spr = sum(ksum);
      if (spr.toString().length() > 1) {
         return solve(sum(p), k);
      }
      return spr;
   }

   public static void main(String[] args) throws FileNotFoundException {
      System.setIn(new FileInputStream(System.getProperty("user.home") + "/" + "in.txt"));
      Scanner in = new Scanner(System.in);

      String n = in.next();
      int k = in.nextInt();

      BigInteger p = new BigInteger(n);
      System.out.println(solve(p, k));
   }
}
```

****

Activity 3:

Given a time in 12 -hour AM/PM format, convert it to Railway's (24-hour) time.

Input Format
A single string s containing a time in -12 hour clock format (HH:mm:ss:am/pm)

Constraints
All input times are valid

Output Format
Convert and print the given time in -24 hour format.

Solution:

```
class GFG
{
	static void print24(String str)
	{
		// Get hours
		int h1 = (int)str.charAt(1) - '0';
		int h2 = (int)str.charAt(0) - '0';
		int hh = (h2 * 10 + h1 % 10);
		// If time is in "AM"
		if (str.charAt(8) == 'A')
		{
			 if (hh == 12)
			{
				System.out.print("00");
				for (int i = 2; i <= 7; i++)
				System.out.print(str.charAt(i));
			}
			else
			{
				for (int i = 0; i <= 7; i++)
				System.out.print(str.charAt(i));
			}
		}
```

```
            // If time is in "PM"
            else
            {
                if (hh == 12)
                {
                    System.out.print("12");
                    for (int i = 2; i <= 7; i++)
                    System.out.print(str.charAt(i));
                }
                else
                {
                    hh = hh + 12;
                    System.out.print(hh);
                    for (int i = 2; i <= 7; i++)
                    System.out.print(str.charAt(i));
                }
            }
        }
        // Driver code
        public static void main (String[] args)
        {
            String str = "07:05:45PM";
            print24(str);
        }
}
```

****

Activity 4:

Brie’s Drawing teacher asks her class to open their books to a page number. Brie can either start turning pages from the front of the book or from the back of the

book. She always turns pages one at a time. When she opens the book, page 1 is always on the right side: When she flips page 1, she sees pages 2 and 3. Each page except the last page will always be printed on both sides. The last page may only be printed on the front, given the length of the book. If the book is n pages long, and she wants to turn to page p, what is the minimum number of pages she will turn? She can start at the beginning or the end of the book.

Given n and p, find and print the minimum number of pages Brie must turn in order to arrive at page p.

Input Format
The first line contains an integer n, the number of pages in the book.
The second line contains an integer, p, the page that Brie's teacher wants her to turn to.

Solution:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
public class Solution {
    static int solve(int n, int p){
        // Brie start turning from page 1
        int min = p/2;
        // Brie start turning from page n
        int n_start = (n/2) - (p/2);
        if(n_start < min){
            min = n_start;
        }
        return min;
    }
```

```
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int p = in.nextInt();
        int result = solve(n, p);
        System.out.println(result);
    }
}
```

****