Strobogrammatic Number

Solution

```
class Strobogrammaticis {

    public static void main(String args[]) {
        System.out.println("Count is: " + strobogrammaticCounter(0,1961));
    }

    public static int strobogrammaticCounter(long low, long high) {
        int count = 0;
        String number = "";
        for (long i = low; i <= high; i++) {
            number = Long.toString(i);
            if (!(number.contains("2") || number.contains("3") ||
number.contains("4") || number.contains("5") || number.contains("7"))) {
                if (i == reverseNumber(i)) {
                    System.out.println(i);
                    count ++;
                }
            }
        }
        return count;
    }

    private static char rotateSixOrNine(char number) {
        switch (number) {
            case '6':
                return '9';
            case '9':
                return '6';
            default:
                return number;
        }    }
    private static long reverseNumber(long number) {
        String numberString = Long.toString(number);
        numberString = new StringBuilder(numberString).reverse().toString();
        String newNumber = "";
        for (int i = 0; i < numberString.length(); i++){
            char digit = numberString.charAt(i);
            newNumber = newNumber + rotateSixOrNine(digit);
        }
        return Long.parseLong(newNumber);   }}
```

Sample Output:

Is 9006 is Strobogrammatic? true

Remainder Theorem

Solution

```
import java.util.*;
class CodeSpeedy{
 static int CRT(int a[], int m[], int n, int p){
   int x = 0;
    for(int i = 0; i<n; i++){
     int M = p/m[i], y = 0; // M1 = p/m1, M2 = p/m2 ....., Mn = p/mn
       for(int j=0; j<m[i]; j++){
       if((M*j)%m[i]==1){
         y = j; break; // Finding the values for  M inverse
       }      }
        x = x + a[i]*M*y; // x = a1*M1*y1 + a2*M2*y2 + ... + an*Mn*yn
   }
       return x%p;    }
 public static void main(String args[]){
   Scanner sc = new Scanner(System.in);
   System.out.println("Enter the number of congruence relations: ");
   int size = sc.nextInt();
   System.out.println("Enter the values of a: ");
   int a[] = new int[size];
   for(int i=0; i<size; i++)
     a[i] = sc.nextInt();
    System.out.println("Enter the values of m: ");
   int m[] = new int[size], p = 1;
   for(int i=0; i<size; i++){
     m[i] = sc.nextInt();
     p = p*m[i]; // p = m1*m2*...*mn
   }
   System.out.println("The solution is "+CRT(a,m,size,p));
 }
}
```

```
Enter the number of congruence relations:

4

Enter the values of a:

1    1    1    0

Enter the values of m:

3    4      5     7

The solution is 301
```

Toggle the switch & Alice Apple tree

Solution

```
// Java program for the above approach
import java.io.*;
class Toggle {

// Function to minimum no. of apples
static int minApples(int M,int K,int N,int S,int W,int E)
{

   // If we get all required apple
   // from South
   if(M <= S * K)
      return M;
    // If we required trees at
   // East and West
   else if(M <= S * K + E + W)
      return S * K + (M-S * K) * K;

   // If we doesn't have enough
   // red apples
   else
      return -1;
}

// Driver code
public static void main(String[] args)
{
   // No. of red apple for gift
   int M = 10;

   // No. of red apple in each tree
   int K = 15;

   // No. of tree in North
   int N = 0;

   // No. of tree in South
   int S = 1;

   // No. of tree in West
   int W = 0;

   // No. of tree in East
   int E = 0;
```

```
    // Function Call
    int ans = minApples(M,K,N,S,W,E);
    System.out.println(ans);
}
}
```

Input: M = 10, K = 15, N = 3, S = 0, W = 1, E = 0

Output: -1