CS Tutoring	Author: Haden Lee
Problem Set #5	Date: 09/04/2011

1 Factorial

Description

Compute $n!$ when a positive integer n is given. Since the number could be very huge, you have to compute $(n!)$ modulo $1,000,007$.

Note

You MUST use a recursive function. Use the following signature:

```
int factorial(int x) {
  // your implementation goes here
  // (1) define your 'base case'
  // (2) solve a recurrence relation (in this case, it is n! = ((n-1)!) * n
  return 0;
}
```

Input Format

A single integer n is given. Assume $n \leq 1,000$.

Output Format

Output $(n!)$ modulo $1,000,007$.

Sample Input

```
10
```

Sample Output

```
362880
```

Sample Input 2

```
20
```

Sample Output 2

```
794133
```

2 Powers

Description

Compute n^p when you are given two positive integers n and p. Since this number could be huge, compute n^p modulo $1,000,007$.

Note

You MUST use a recursive function. Use the following signature:

```
int power(int n, int p) {
  // your implementation goes here
  // (1) define your 'base case'
  // (2) solve a recurrence relation (in this case, it is n^p = (n^{p-1}) * n
  return 0;
}
```

Input Format

Two integers n and p are given. Assume $1 \leq p \leq 100$ and $1 \leq n \leq 100$.

Output Format

Output n^p modulo $1,000,007$.

Sample Input

```
5 2
```

Sample Output

```
25
```

Sample Input 2

```
2 20
```

Sample Output 2

```
48569
```

3 Fibonacci, Take Two

Description

Compute the n-th Fibonacci number, given n. $F_0 = F_1 = 1$. Since F_n could be huge, compute F_n modulo $1,000,007$.

Note

You MUST use a recursive function. Use the following signature:

```
int fibo(int n) {
  // your implementation goes here
  // (1) define your 'base case'
  // (2) solve a recurrence relation (in this case, it is F_{n} = F_{n-1} + F_{n-2})
  return 0;
}
```

Note2

Since your function fibo() will be called exponentially many times, declare a global variable int answer[], and check if the answer was ALREADY computed previously by comb(). If so, you can just return the value in ans[n]; otherwise, you compute the value AND STORE it in ans[n].

Input Format

An integer n is given. n is always between 0 and $1,000$.

Output Format

Output F_n modulo $1,000,007$.

Sample Input

```
0
```

Sample Output

```
1
```

Sample Input 2

```
5
```

Sample Output 2

```
5
```

4 Pascal, Take Two

Description

Compute n choose r (formally, $\binom{n}{r} = \frac{n!}{r! \cdot (n-r)!}$). Since the number could be very huge, you have to compute $\binom{n}{r}$ modulo $1,000,007$.

Note

You MUST use a recursive function. Use the following signature:

```
int comb(int n, int r) {
  // your implementation goes here
  // (1) define your 'base case'
  // (2) solve a recurrence relation (in this case, it is n choose r = (n-1 choose r-1)
  return 0;
}
```

Note2

Since your function comb() will be called exponentially many times, declare a global variable int answer[][], and check if the answer was ALREADY computed previously by comb(). If so, you can just return the value in ans[n][r]; otherwise, you compute the value AND STORE it in ans[n][r].

Input Format

Two integers n and r are given. Assume $0 \leq r \leq n \leq 1,000$.

Output Format

Output $\binom{n}{r}$ modulo $1,000,007$.

Sample Input

```
5 2
```

Sample Output

```
10
```

Sample Input 2

```
30 15
```

Sample Output 2

```
116435
```

5 Car Plate

Description

Given a string, determine whether it is a valid Californian car plate number, according to the following rule:

For simplicity, a valid Californian car plate number:

- Must be 7 characters long such that
- the first character is a number between 1 and 9,
- the following three characters are upper-case letters,
- and the following three characters are digits including zeros

Valid car plate numbers are: 1ABC123, 9XZY009, and 6SLJ356. Some invalid car plate numbers are: VF, 0XTZ123, 3aBX334, ASDF999, 8HELLO5, etc.

Input Format

A string is given, whose length is between 1 and 10.

Output Format

Output "Valid" or "Invalid" based on the rule above.

Sample Input

```
1ABC123
```

Sample Output

```
Valid
```

Sample Input 2

```
3AbX312
```

Sample Output 2

```
Invalid
```

Sample Input 3

```
0XYZ992
```

Sample Output 3

```
Invalid
```