A. ICPC TUTORIAL

PROBLEM DESCRIPTION

Given *m*, *n*, and *t*, where *m* denotes the maximum amount of operations, *n* denotes the input size and *t* denotes the type of algorithm complexity.

Determine whether a user with a <Type t> complexity solution has successfully passed the test data or not (ie. O(n) <= m).

SOLUTION TECHINQUES

Brute force / Simulation

SOLUTION SKETCHES

Declare another variable *tmp* to calculate O(n).
Since both m and n are reasonably large, be sure to check whether *tmp* has surpassed *m* DURING calculation to avoid integer overflow.

TIME COMPLEXITY

Type 2 requires the most time: $O(\log_2 m)$

SOLUTION PROGRAM FOR REFERENCE

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>

using namespace std;

int main()
{
    char tle = 0;
    int i;
    long long m, n, t, tmp;
    cin >> m >> n >> t;
    if (t == 1)
    {
        tmp = 1;
        for (i = 1; i <= n; i++)
        {
            tmp = tmp * i;
            if (tmp > m) { tle = 1; break; }
        }
    }
    else if (t == 2)
    {
        tmp = 1;
        for (i = 0; i < n; i++)
        {
            tmp <<= 1;
            if (tmp > m) { tle = 1; break; }
        }
    }
    else if (t == 3)
    {
        tmp = 1;
        for (i = 0; i < 4; i++)
        {
            tmp = tmp * n;
            if (tmp > m) { tle = 1; break; }
        }
    }
    else if (t == 4)
    {
        tmp = 1;
        for (i = 0; i < 3; i++)
        {
            tmp = tmp * n;
            if (tmp > m) { tle = 1; break; }
        }
    }
    else if (t == 5)
    {
        tmp = 1;
        for (i = 0; i < 2; i++)
        {
            tmp = tmp * n;
            if (tmp > m) { tle = 1; break; }
```

```
        }
    }
    else if (t == 6)
    {
        if ((double)n * log2(n) > m) tle = 1;
    }
    else if (t == 7)
    {
        if (n > m) tle = 1;
    }
    puts(tle ? "TLE" : "AC");
    return 0;
}
```