#include <stdio.h>

long long modular\_exponentiation(long long base, long long exponent, long long mod)

{

long long result = 1;

base = base % mod;

while (exponent > 0)

{

if (exponent % 2 == 1)

{

result = (result \* base) % mod;

}

base = (base \* base) % mod;

exponent = exponent / 2;

}

return result;

}

int last\_two\_digits\_optimized(long long K, long long N)

{

long long cycle\_length = 100;

long long full\_cycles = K / cycle\_length;

long long remainder = K % cycle\_length;

long long sum\_full\_cycle = 0;

for (long long i = 1; i <= cycle\_length; i++)

{

sum\_full\_cycle = (sum\_full\_cycle + modular\_exponentiation(i, N, 100)) % 100;

}

long long total = (full\_cycles \* sum\_full\_cycle) % 100;

for (long long i = 1; i <= remainder; i++)

{

total = (total + modular\_exponentiation(i, N, 100)) % 100;

}

return (int)total;

}

int main()

{

int T;

scanf("%d", &T);

while (T--)

{

long long K, N;

scanf("%lld %lld", &K, &N);

int result = last\_two\_digits\_optimized(K, N);

printf("%02d\n", result);

}

return 0;

}