// 2^3=8 Compute power(a,b)

#include <iostream>

using namespace std;

int power(int a, int b)

{

if(b==0)

return 1;

return a\*power(a,b-1);

}

int main()

{

int a,b;

cin>>a>>b;

cout<<power(a,b);

}

// Its time complexity depends upon value of b. As 2^7 then 2\*2\*2\*2\*2\*2\*2, so the number of iterations is depend upon value of b. This can be reduced as follows:

F(a,b)=f(a,b/2)^2 if b is even

F(a,b)=a\*f(a,b/2)^2 if b is odd

Ex:

2^9= 2\*(2^4)^2

(2^2)^2

(2^1)^2

2\*(2^0)^2

// Now the time complexity is reduced to logB (base 2)

Program-

#include <iostream>

using namespace std;

int fastpower(int a, int b)

{

if(b==0)

return 1;

if(b%2==0)

return fastpower(a,b/2)\*fastpower(a,b/2);

else

return a\*fastpower(a,b/2)\*fastpower(a,b/2);

}

int main()

{

int a,b;

cin>>a>>b;

cout<<fastpower(a,b);

}