Write a program to calculate pow(x,n) - GeeksforGeeks

Below solution divides the problem into subproblems of size y/2 and call the subproblems recursively.

```
#include<stdio.h>/* Function to calculate x raised to the power y */
int power(int x, unsigned int y)
{
    if( y == 0)
        return 1;
    else if (y%2 == 0)
        return power(x, y/2)*power(x, y/2);
    else
        return x*power(x, y/2)*power(x, y/2);
}/* Program to test function power */
int main()
{
    int x = 2;
    unsigned int y = 3;
    printf("%d", power(x, y));
    getchar();
    return 0;
}
```

Time Complexity: O(n)

Space Complexity: O(1)

Algorithmic Paradigm: Divide and conquer.

Above function can be optimized to O(logn) by calculating power(x, y/2) only once and storing it.

```
/* Function to calculate x raised to the power y in O(logn)*/
int power(int x, unsigned int y)
{
    int temp;
    if( y == 0)
        return 1;
    temp = power(x, y/2);
    if (y%2 == 0)
        return temp*temp;
    else
        return x*temp*temp;
}
```

Run on IDE

Time Complexity of optimized solution: O(logn)

Let us extend the pow function to work for negative y and float x.

```
/* Extended version of power function that can work
   for float x and negative y*/
#include<stdio.h>
float power(float x, int y)
{
    float temp;
    if( y == 0)
        return 1;
    temp = power(x, y/2);
    if (y%2 == 0)
        return temp*temp;
    else
    {
        if(y > 0)
            return x*temp*temp;
        else
            return (temp*temp)/x;
    }
}  /* Program to test function power */
int main()
{
    float x = 2;
    int y = -3;
    printf("%f", power(x, y));
    getchar();
    return 0;
}
```