Write an iterative O(Log y) function for pow(x, y)

Given an integer x and a positive number y, write a function that computes xy under following conditions.  
a) Time complexity of the function should be O(Log y)  
b) Extra Space is O(1)

**Examples:**

Input: x = 3, y = 5

Output: 243

Input: x = 2, y = 5

Output: 32

We have discussed [recursive O(Log y) solution for power](https://www.geeksforgeeks.org/write-a-c-program-to-calculate-powxn/). The recursive solutions are generally not preferred as they require space on call stack and they involve function call overhead.

Following is implementation to compute xy.

|  |
| --- |
| // Iterative C program to implement pow(x, n)  #include <stdio.h>    /\* Iterative Function to calculate (x^y) in O(logy) \*/  int power(int x, unsigned int y)  {      int res = 1; // Initialize result        while (y > 0) {          // If y is odd, multiply x with result          if (y & 1)              res = res \* x;            // n must be even now          y = y >> 1; // y = y/2          x = x \* x; // Change x to x^2      }      return res;  }    // Driver program to test above functions  int main()  {      int x = 3;      unsigned int y = 5;        printf("Power is %d", power(x, y));        return 0;  } |

**Output:**

Power is 243

/\* 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));

return 0;

}