***Iterative Power***

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

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.

C++Java

// Iterative Java program

// to implement pow(x, n)

import java.io.\*;

class GFG

{

/\* Iterative Function to

calculate (x^y) in O(logy) \*/

static int power(int x, int y)

{

// Initialize result

int res = 1;

while (y > 0)

{

// If y is odd,

// multiply

// x with result

if ((y & 1) == 1)

res = res \* x;

// y must be even now

y = y >> 1; // y = y/2

x = x \* x; // Change x to x^2

}

return res;

}

// Driver Code

public static void main (String[] args)

{

int x = 3;

int y = 5;

System.out.println("Power is " +

power(x, y));

}

}

**Output:**

Power is 243

***Time Complexity:****O(log y), since in loop each time the value of y decreases by half it’s current value.*

***Auxiliary Space:****O(1), since no extra space has been taken.*