CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Design and Analysis of Algorithms

Experiment 1.2

23CSP-301

Submitted To:
Faculty Name: Er. Moh. Shaqlain

Submitted By:
Name: Garvi Dabas
UID: 23BCS11346
Section: KRG - 2B
Semester: 5th

Aim:
To implement the power function efficiently using **Divide and Conquer** (exponentiation by squaring).

Algorithm(Pseudocode):

```
function power(x, y):
    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   // handles negative powers
```

Code:

```
#include <iostream>
using namespace std;

// Function to calculate power in O(log n)
double power(double x, int y) {
    if (y == 0)
        return 1;

    double 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;
    }
}

int main() {
    double x;
    int y;

    cout << "Enter base (x): ";
    cin >> x;
```

```
    cout << "Enter power (y): ";

    cin >> y;
    cout << x << "^" << y << " = " << power(x, y) << endl;
    return 0;
}
```

Output 1:

```
Enter base (x): 2
Enter power (y): 10
2^10 = 1024

Enter base (x): 2
Enter power (y): -3
2^-3 = 0.125

Enter base (x): 5
Enter power (y): 0
5^0 = 1
```

Time Complexity Analysis

- Each recursive call divides y by 2, so the number of steps is $\log_2(y)$.
- Hence, Time Complexity = O(log n).
- Space Complexity (due to recursion) = O(log n).

Conclusion:

The power function was successfully implemented in O(log n) time using the divide and conquer approach (exponentiation by squaring). The program handles zero, negative, even, and odd powers effectively.