



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment - 2

**Student Name:** Ayush Ranjan

**Branch:** BE-CSE

**Semester:** 5<sup>th</sup>

**Subject Name:** DAA

**UID:** 23BCS10187

**Section/Group:** KRG-2-B

**Date of Performance:** 24/7/25

**Subject Code:** 23CSH-301

1. **Aim:** To implement a program in C++ that computes the power of a number using fast exponentiation with  **$O(\log n)$**  time complexity.

## 2. Procedure:

- Start the program and include necessary headers.
- Define a function `power(base, exponent)` that uses divide-and-conquer.
- If `exponent = 0`, return 1 (base case).
- Recursively compute `power(base, exponent / 2)`.
- If `exponent` is even, `result = half × half`.
- If `exponent` is odd, `result = base × half × half`.
- In the `main()` function, read or define values of base and exponent, call the function, and display the result.
- In the `main()` function, test all these operations with user-defined values.

## 3. Code:

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

```
long long power(long long base, long long exp) {
    if (exp == 0) return 1;
    long long half = power(base, exp / 2);
    if (exp % 2 == 0)
        return half * half;
    else
        return base * half * half;
}
```

```
int main() {
    long long base = 2, exp = 10;
    cout << base << "^" << exp << " = " << power(base, exp) << endl;
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
return 0;  
}
```

## 4. Output:

A dark-themed terminal window showing the output of a program. The text '2^10 = 1024' is displayed in a monospaced font, with '2' in green, '^' in red, '10' in green, '=' in red, and '1024' in red.

```
2^10 = 1024
```

## 5. Learning Outcomes:

- Learnt how to reduce power calculation from  $O(n)$  multiplications to  $O(\log n)$ .
- Gained understanding of **divide-and-conquer** approach in recursive functions.
- Understood the difference between handling even and odd exponents.
- Practised writing recursive functions and analyzing their time complexity.
- Developed confidence in optimizing algorithms using mathematical properties.