DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

NAAC GRADE A+ Accredited University

UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE/IT-5th Sem)

Design and Analysis of Algorithms

Subject Code: 23CSH-301/ITH-301

Submitted to:
Faculty name: MD. Shaqlain

Submitted by:
Name: Reetesh Sahu
UID: 23BCS10018
Section: KRG-1
Group: A

INDEX

Ex. No	List of Experiments	Date	Conduct (MM: 12)	Viva (MM: 10)	Worksheet (MM: 8)	Total (MM:30)	Remarks/Signature
1.1	Analyze if stack Isempty, Isfull and if elements are present then return top element in stacks using templates and also perform push and pop operation in stack.	21/07/25					
1.2	Develop a program for implementation of power function and determine that complexity should be O (log n).	01/08/25					
1.3	Evaluate the complexity of the developed program to find frequency of elements in a given array.						
1.4	i. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and end of Singly Linked List. ii. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.						

2.1	Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random number generator.						
2.2	Develop a program and analyze complexity to implement subset-sum problem using Dynamic Programming.						
2.3	Develop a program and analyze complexity to implement 0-1 Knapsack using Dynamic Programming.						
3.1	Develop a program and analyze complexity to find shortest paths in a graph with positive edge weights using Dijkstra's algorithm.						
3.2	Develop a program and analyze complexity to find all occurrences of a pattern P in a given string S.						
3.3	Lab Based Mini Project.						

Discover. Learn. Empower.

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Experiment No: 1.2

Student Name: Reetesh Sahu
UID: 23BCS10018
Branch: CSE
Section/Group: Krg-1A
Semester: 5th
Date of Performance: 01/08/25
Subject Name: Design analysis and algorithm
Subject Code: 23CSH-301

Aim:

Code implements power function in O(logn) time complexity.

Procedure:

1. Read two inputs: **number (base)** and **exponent (n)**.
2. If n = 0, directly return **1** (since anything to power 0 is 1).
3. If n is negative:
   - Convert base → 1/base.
   - Make exponent positive → n = -n.
4. Set an initial variable ans = 1.
5. Repeat while n is greater than 0:
   - If n is **odd**, multiply current base into ans.
   - Square the base (base = base * base).
   - Reduce exponent by half (n = n / 2).
6. When loop ends, output ans.

Code:

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

long long fastPower(int base, int exp) {
    long long result = 1;

    while (exp > 0) {
        if (exp % 2 != 0) {    // check odd exponent
            result = result * base;
        }
        base = base * base;    // square the base
        exp = exp / 2;         // divide exponent by 2
```

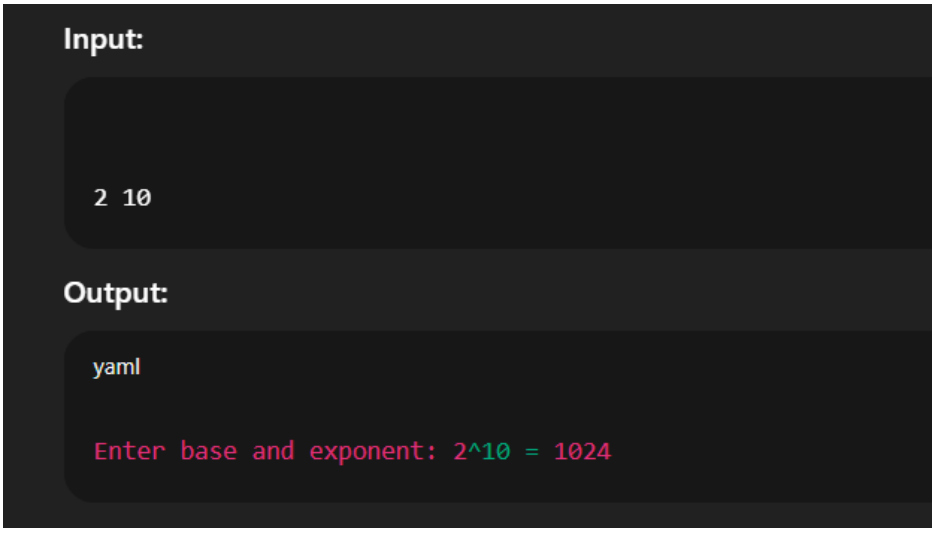
```
    }

    return result;
}

int main() {
    int base, exponent;
    cout << "Enter base and exponent: ";
    cin >> base >> exponent;

    cout << base << "^" << exponent << " = " << fastPower(base, exponent) << endl;
    return 0;
}
```

Output:



- **Time Complexity Analysis**

- The loop continues as long as n > 0.
- In each pass, the exponent is halved (n = n / 2).
- Hence, the number of iterations is about **$\log_2(n)$**.
- Inside the loop, only a few constant-time operations are done (multiplication, checking odd/even).
- **Final Complexity:** O(log n)

- **Learning Takeaways**

1. Efficient method of **Exponentiation using Repeated Squaring**.
2. Understanding how **binary representation of exponent** helps optimize power calculation.
3. Idea of minimizing chances of **overflow** by controlled multiplications.
4. Implementation is simple and doesn't rely on **extra libraries or STL functions**.