



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment 2

**Student Name: ISHITA NISHANT**

**Branch: BE CSE**

**Semester: 5**

**Subject Name: DAA**

**UID: 23BCS11354**

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

**Date of Performance: 30/7/2025**

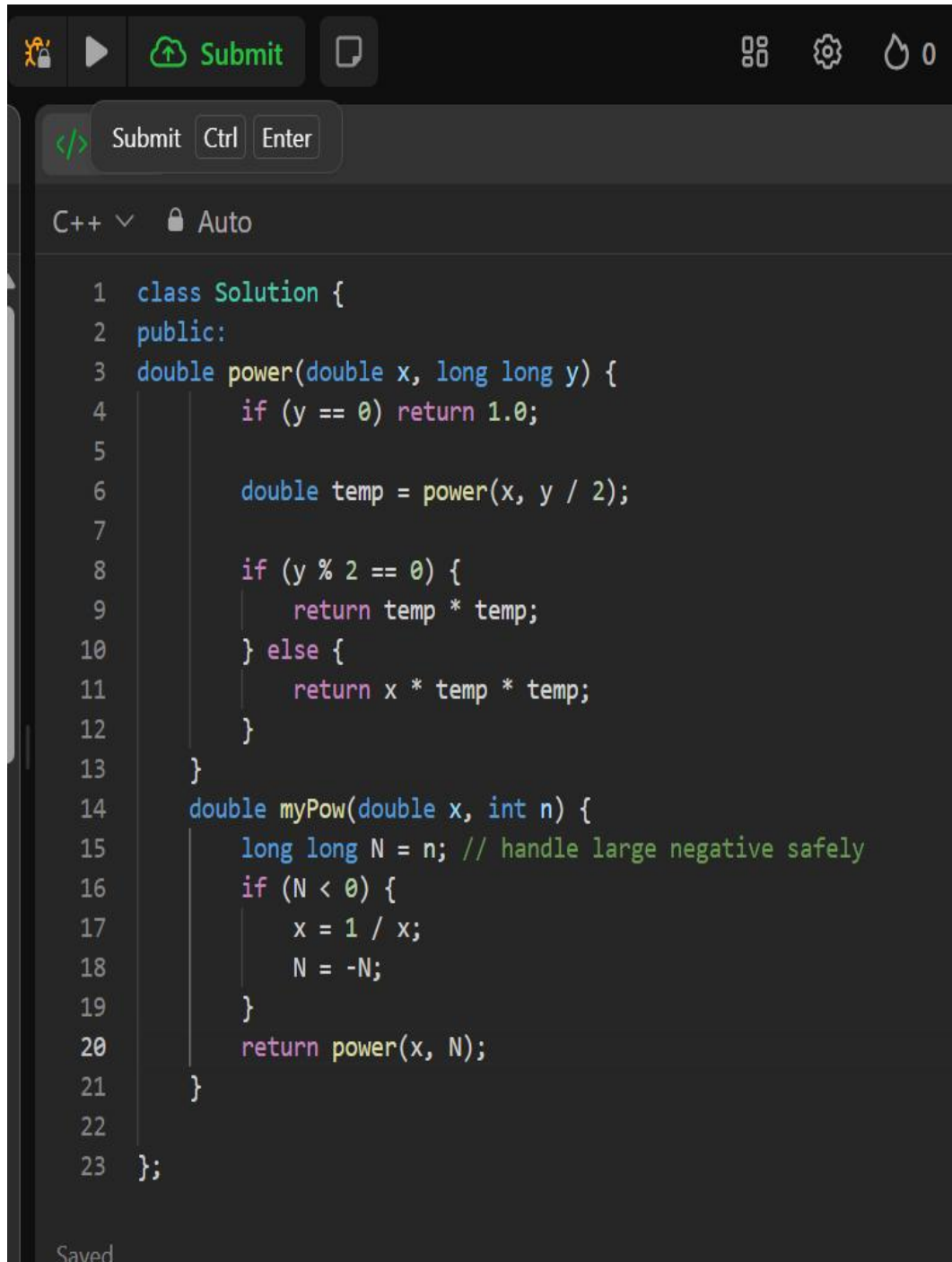
**Subject Code: 23CSH-301**

- 1. Aim:** Implement a power function that computes  $xyx^{yx}$  in  $O(\log n)$  time complexity.
- 2. Objective:** To implement and understand an optimized recursive approach for exponentiation using divide and conquer.
- 3. Implementation/Code:**

```
#include <iostream>
using namespace std;
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=2;
    int y=-3;
    cout<<"Result: "<<power(x,y)<<endl;
```

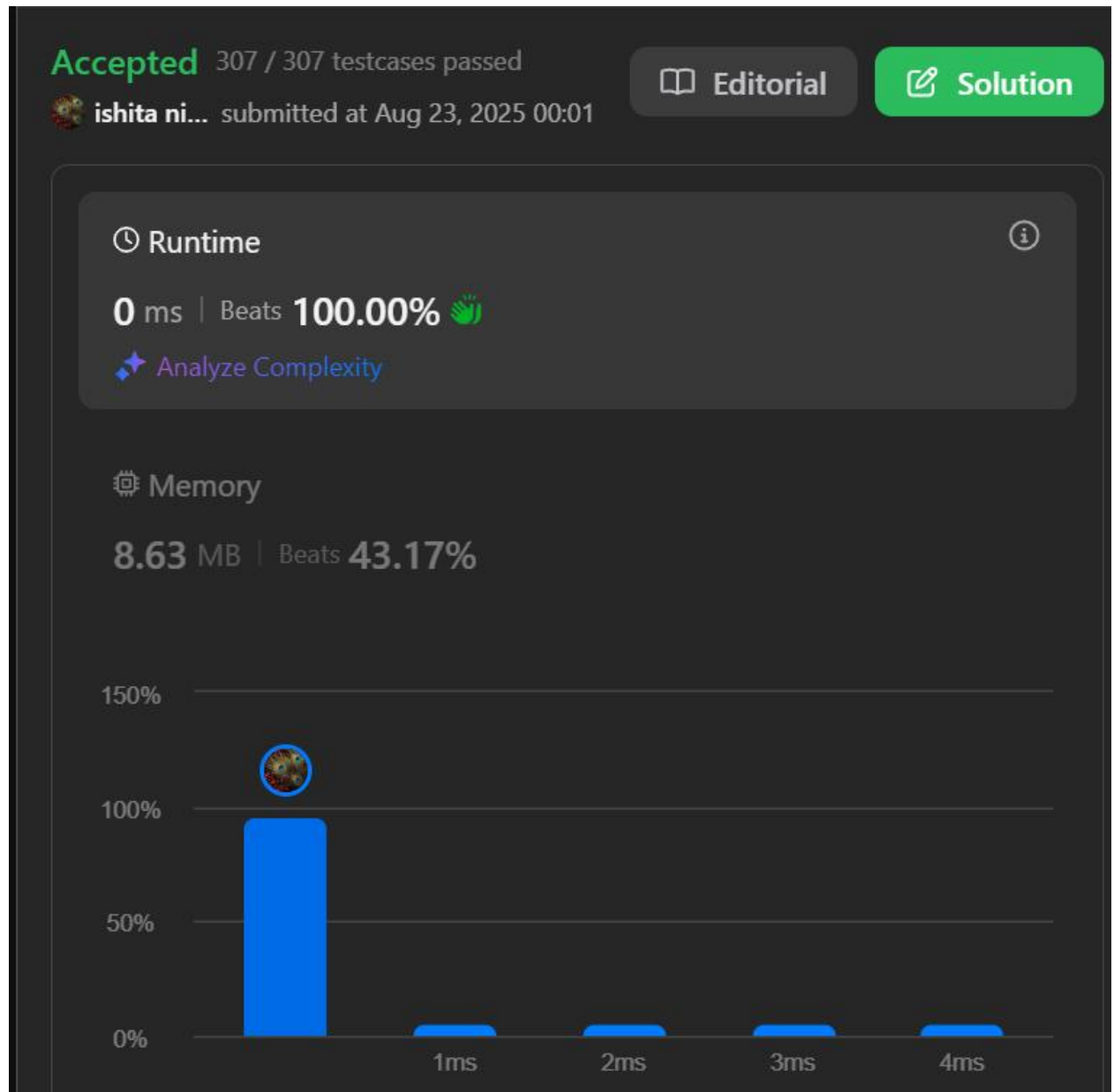
```
    return 0;  
}
```

## 4. Output



```
1  class Solution {  
2  public:  
3  double power(double x, long long y) {  
4      if (y == 0) return 1.0;  
5  
6      double temp = power(x, y / 2);  
7  
8      if (y % 2 == 0) {  
9          return temp * temp;  
10     } else {  
11         return x * temp * temp;  
12     }  
13 }  
14 double myPow(double x, int n) {  
15     long long N = n; // handle large negative safely  
16     if (N < 0) {  
17         x = 1 / x;  
18         N = -N;  
19     }  
20     return power(x, N);  
21 }  
22  
23 };
```

Saved



## 5. Learning Outcome

- Understand the concept of divide-and-conquer in recursive functions.
- Implement exponentiation in  $O(\log n)$  time.
- Handle negative and zero power cases efficiently.



# **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.