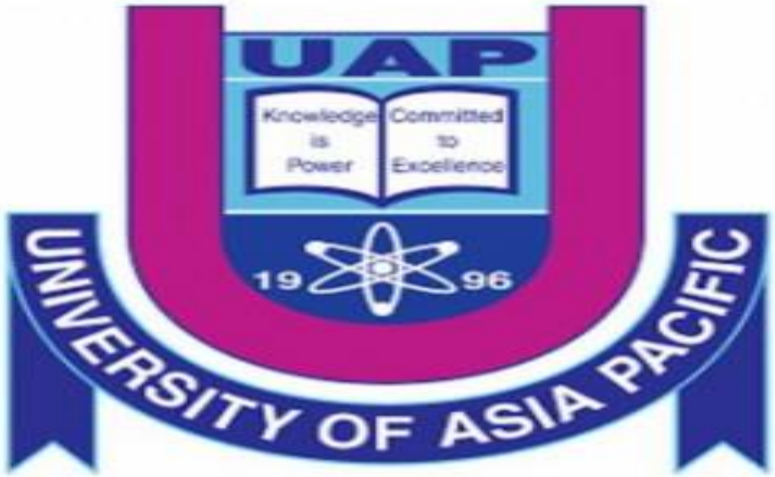

Submitted By	Submitted To
Name: Muhammd Mehedi Hasan Reg: 23201143 Section: C Subject: CSE 108	Name: Zaima Sartaj Taheri Lecturer Department of Computer Science and Engineering, University of Asia Pacific

Title: Binary Exponentiation

Explanation:

Binary exponenstiation (also known as exponentiation by squaring) is a trick which allows to calculate a^n using only (log n) multiplication (instead of 0(n) multiplications required by the native approach).

It also has important applications in many tasks unrelated to arithmetic, since it can be used with any operations that have the property of associativity.

(X.Y).Z = X . (y . Z)

Raising a to the power of n is expressed naively as multiplication by a done n-1 times: a^n = a.a….-a. However, this approach is not practical fo large a or n.

a^b+c = a^b. a^c and a^26 = a^b.a^b = (a^b)^2

The idea of binary exponentiation is, that we split the work using the binary representation of the exponent.

Let's write 72 in base 2, for example:

3^13 = 3^(1101)2 = 3^8. 3^4. 3^1

Since the number n has exactly [log2^n]+1 digits in base 2, we only need perform 0 (log n) multiplications, if we know the powers a^1, a^2, a^4, a^8,…..,a^2[log n]

3^1 = 3

3^2 = (3^1)^2 = 3^2 = 9

3^4 = (3^2)^2 = 9^2 = 81

3^8 = (3^4)^2 = 81^2 = 6561

Code in iterative method:

```
int power(int a, int b) {

int result = 1;

while(b > a) {
if(b % 2 == 1) result *= a;

a *= a;

b /= 2;

}

return result;

}
```

Code explanation with an example:

First, the recursive approach, which is a direct translation of the recursive formula: long long binpow(long long a, long long b) {

```
if (b == 0)

return 1;

long long res = binpow(a, b / 2);

if (b % 2)

return res * res * a;

else

return res * res;

}
```

```
#include <stdio.h>

long long binaryExponentiation(long long a, long long n) {

if (n == 0) return 1;
```

```
if (n % 2 == 0) {
long long temp = binaryExponentiation(a, n / 2);
return temp * temp;
} else {
return a * binaryExponentiation(a, (n - 1)/2);
}
}
int main() {
long long a, n;
printf("Enter the base (a): ");
scanf("%lld", &a);
printf("Enter the exponent (n): ");
scanf("%lld", &n);
printf("%lld raised to the power %lld is %lld\n", a, n, binaryExponentiation(a, n));
return 0;
```

Discussion:

Binary Exponentiation is an efficient algorithm for calculating large powers in logarithmic time, commonly used in competitive programming. It repeatedly squares the base and reduces the exponent by half, leveraging binary representation. This method reduces O(n)O(n)O(n) time complexity to O(log⁡n)O(\log n)O(logn), making it ideal for modular exponentiation.