Bit_maniplulation

1

```
//bitwise operator is fast than no bitwise oprator
// AND OR XOR One's complement Left shift Right shift
#include <bits/stdc++.h>
using namespace std;
int main(){
    int AK = 0;
    cout << (~AK) << endl;

    //binary left shift <<
    // 5 << 2 then 0000101 ne 0010100 kare binary ma
    // so a << b is a*2^b

    //binary right shift >>
    //that means a>>b is a/2^b
}
```

2

```
//expression to check the number even or odd
// last bit decide that the number is even or odd
#include <bits/stdc++.h>
using namespace std;
int main(){
    int x;
    cin >> x;

    if(x&1){
        cout << "Odd" ;
    }else{
        cout << "Even";
    }
}
```

3

```
//get the position of the bit from the number
#include <bits/stdc++.h>
using namespace std;
int getIthBit(int n, int i){
    int mask = (1<<i);
    return (n & mask) > 0 ?1 :0;
}
//set the ith bit { ==> set bits are those which are one}
void setIthBit(int &n , int i ){
    int mask = (1<<i);
    n = (n|mask);
}
//clear th ith bit
void clearIthBit(int n, int i){
    int mask = ~(1<<i);
    n = n & mask;
}
//update the ith bit
void updateIthBit(int &n, int i, int v){
    clearIthBit(n,i);
    int mask = (v<<i);
    n = n|mask;  //sets the right value
}
//clear give the bits from the last
void clearLastBits(int &n , int i){
    int mask = (-1<<i);
    n = n & mask;
}
//clear bit in ranges given
void clearBitsInRange(int &n , int i,int j){
    int a = (~0)<<(j+1);
    int b = (1<<i) - 1;
    int mask = a|b;
    n = n & mask ;
}
int main(){
    int n = 5;
    int i;
    cin >> i;
    // cout << getIthBit(n,i) <<endl;
     setIthBit(n,i);
    cout << n << endl;
    int a=15;
    int j=2;
    clearLastBits(a,j);
    cout << a << endl;

}
```

4

```
//Replace bits in N by M
/*
you are given two 32-bit numbers, N and M  and two bit positions i and j
write a mothod to set all bits between i and j in N equal to M.
M (becomes a substring of N locationed at and starting at j)

Example:
```

```cpp
N = 10000000000;
M = 10101;
i = 2 and j = 6;
Output : 1001010100
*/
#include <bits/stdc++.h>
using namespace std;
//clear bits by ranges given
void clearBitsInRange(int &n , int i,int j){
    int a = (~0)<<(j+1);
    int b = (1<<i) - 1;
    int mask = a|b;
    n = n & mask ;
}
void replaceBits(int &n , int i , int j , int m){
    clearBitsInRange(n,i,j);
    int mask = (m<<i);
    n = n|mask;
}
int main(){
    int n = 15;
    int i = 1;
    int j = 3;
    int m = 2;
    replaceBits(n,i,j,m);
    cout << n;
    return 0;
}
```

5

```cpp
//power of two or not
#include <bits/stdc++.h>
using namespace std;
int main(){
    // 10000 => 16
    // 01111 => 15 (n-1)
    // and if we do & then we will get 0
    int n;
    cin >> n;
    if((n&(n-1))==0){
        cout << "Yes";
    }
    else{
        cout << " no";
    }
}
```

6

```
//set bits of any number
//set bits are those which is 1
//so we have to count the available 1 in the given input's binary
represntation
#include <bits/stdc++.h>
using namespace std;
int count_bits(int n){
    int count = 0;
    while(n>0){
        int last_bit = (n&1);
        count += last_bit;
        n = n>>1;
    }
    return count;
}
//counting sets bits : Hack (Faster Method)
int count_bits_hack(int n){
    int ans = 0;
    while(n>0){
        //removes the last set bit from the current number
        n = n & (n-1);
        ans++;
    }
    return ans;
}
int main(){
    int n;
    cin >> n;
    cout <<count_bits(n)<<endl;
    cout <<count_bits_hack(n);

}
```

7

```
//Fast Exponentiation  like 5^3=125 and its take o(n) time but this is better
// in log(n) time

#include <bits/stdc++.h>
using namespace std;
int fastExpo(int a ,int n){
    int ans = 1;
    while(n>0){
        int last_bit = (n&1);
        if(last_bit){
            ans = ans*a;
        }
        a = a*a;
        n = n>>1;
    }
    return ans;
}
int main(){
    int a,n;
    cin >> a >> n;
    cout << fastExpo(a,n);

}
```

8

```
//decimal to binary
#include <bits/stdc++.h>
using namespace std;
int converToBinary(int n){
    int ans = 0;
    int p = 1;
    while(n>0){
        int last_bit =(n&1);
        ans += p*last_bit;
        p = p*10;
        n = n>>1;
    }
    return ans;
}
int main(){
    int n;
    cin >> n;
    cout << converToBinary(n) << endl;

}
```