BIT MANIPULATION

Q1)normal operations in bit manipulation

```
#include <bits/stdc++.h>using
namespace  std; int main() {
 int x=3;int
 y=6;
 int z=x^y;//XOR int
 a=x|y;//OR int
 b=x&y;//AND
 cout<<z<<endl<<a<<endl<<b;return 0;
}
```

Q2)left/right shift operators

```
#include <bits/stdc++.h> using
namespace std;

int main() {int
 x=3; int y=1;
 int z=(x<<y);//left shift int
a=(x>>y);//right shift
cout<<z<<endl<<a; return 0;
}
```

3)signed/unsigned operators

```
#include <bits/stdc++.h>using
namespace  std; int main() {
unsigned int x=1;//only +ve values
signed int y=1;//can store -ve values as well
cout<<(~x)<<endl;
cout<<(~y);
return 0;
}
```

Q4)set the bits

Naïve(multiply method)

```
#include <bits/stdc++.h>
using namespace std;
void isset(int n,int k){
   int x=1;
   for(int i=0;i<(k-1);i++){
      x=x*2;
   }
   if((x&n)!=0){
      cout<<"yes"<<endl;
   }else{
      cout<<"no"<<endl;
   }
}
int main() {
int n,k;
cin>>n>>k;
isset(n,k);
return 0;
}
```

Naïve(divison method)

```
#include <bits/stdc++.h>
using namespace std;
void isset(int n,int k){

 for(int i=0;i<(k-1);i++){
n=n/2;
}
if((n&1)!=0){
 cout<<"yes"<<endl;
}else{
cout<<"no"<<endl;

 }
}

int main() { int n,k;
cin>>n>>k;
 isset(n,k);
return 0;
}
```

Pro(this is done using left/right shift operator)

```
#include <bits/stdc++.h>
using namespace std;
 void isset(int n,int k){
int x=(1<<(k-1));
if((x&n)!=0){
 cout<<"yes"<<endl;
}else{
cout<<"no"<<endl;
}
}


int main() {
int n,k;
 cin>>n>>k;
 isset(n,k);
return 0;
}
```

Q5)count the number of set bits

```
#include <bits/stdc++.h>
using namespace std;
void count(int n){
int res=0;
 while(n>0){
 if(n%2==1){
res++;
}
n=n/2;
}
cout<<res;
}
int main() {
int n;
 cin>>n;
count(n);
 return 0;
}
```

Built in function=_built_in_popcountll(a)
ll or not depend on the function

Q6) brian cunningham algorithm

```
#include <bits/stdc++.h>
 using namespace std;
 void count(int n){
int res=0;
while(n>0){
n=n&(n-1);
res++;
}
cout<<res;
}
int main() {
int n;
 cin>>n;
count(n);
 return 0;
}
```

Q7)power of 2
Naïve

```
#include <bits/stdc++.h>
using namespace std;
void power(int n){
if(n==0){
cout<<"no"<<endl;
exit(0);
}
while(n!=1){
if(n%2!=0){
cout<<"no"<<endl;
exit(0);
}
n=n/2;
}
cout<<"yes"<<endl;
}
int main() {
 int n;
cin>>n;
power(n);
 return 0;
}
```

Pro(brian cunningham algorithm)

```
#include <bits/stdc++.h>
using namespace std;
void power(int n){
if(n==0){
cout<<"no"<<endl;
exit(0);
}

while(n>0){

if((n&(n-1))!=0){
cout<<"no"<<endl; exit(0);

}

n=n/2;
}

cout<<"yes";
}

int main() {

int n;
```

```
 cin>>n; power(n); return 0;

}
```

Q8)one odd occurring

Naïve

```
#include <bits/stdc++.h>
 using namespace std;
int power(int arr[],int n){
 for(int i = 0; i < n; i++){
int count = 0;
for(int j = 0; j < n; j++){
if(arr[i] == arr[j])
count++;
}
if(count % 2 != 0)
return i;
}
}
int main() {
 int n;
 cin>>n;
int arr[n];
for(int i=0;i<n;i++){
cin>>arr[i];
}
cout<<power(arr,n); return 0;
}
```

Pro(using xor operator)

```
 #include <bits/stdc++.h>
 using namespace std;
 void power(int arr[],int n){
 int res = 0;
for(int i = 0; i < n; i++){
 res = res ^ arr[i];
}
cout<<res;
}
int main() {
int n; cin>>n;
int arr[n];
for(int i=0;i<n;i++){
cin>>arr[i];
}
power(arr,n);
return 0;
}
```

Q9)Two odd occurring

Pro

```
#include <bits/stdc++.h>
using namespace std;
void oddAppearing(int arr[], int n){
      int xors = 0, res1 = 0, res2 = 0;

      for (int i = 0; i < n; i++){
      xors = xors ^ arr[i];
      }//intialiasing all the things

      int sn = xors & (~(xors - 1));
 //helps to find the last element

      for (int i = 0; i < n; i++){
         if ((arr[i] & sn) != 0)//first part
            res1 = res1 ^ arr[i];
         else//second part
            res2 = res2 ^ arr[i];
      }



      cout <<  res1 << " " << res2;
}
 int main() {
int arr[]= {3, 4, 3, 4, 5, 4, 4, 6, 7, 7}, n = 10;
   oddAppearing(arr, n);
}
```

Q10)Divide two integers and find quotient without using multiplication,divison and modulo..

https://leetcode.com/problems/divide-two-integers/description/

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve(ll dividend,ll divisor){
   bool
sign=((dividend)>=0)==((divisor)>=0)?true:false;
   dividend=abs(dividend);
   divisor=abs(divisor);
   ll result=0;
   while(dividend-divisor>0){
      ll count=0;
      while(dividend-(divisor<<1<<count)>=0){
         count++;
      }
      result+=(1<<count);
      dividend-=(divisor<<count);
      }
      cout<<result;
}
int main(){
   ll m,temp,n;
   cin>>m>>n;
   solve(m,n);
```

```
}
```

Q11)power set

Important for having all the permutations..

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve(string &s){
   int n=s.length();
   int total=(1<<n);
   for(ll i=0;i<total;i++){
      for(ll j=0;j<n;j++){
         if((i&(1<<j))!=0){
            cout<<s[j];
         }
      }
      cout<<endl;
   }
}
int main(){
   string s;
   cin>>s;
   solve(s);
}
```

Q12)Storing permutations and printing one of the them..

For int:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;

vector<vector<ll>> subsets(vector<ll>& nums){
   int n=nums.size();
   ll temp;
   int size=(1<<n);
   vector<vector<ll>>subsets;
   for(ll i=0;i<size;i++){
      vector<ll>subset;
      for(ll j =0;j<n;j++){
         if((i&(1<<j))!=0){//bracket imp
            temp=nums[j];
            subset.push_back(temp);
         }
      }
      subsets.push_back(subset);
   }
   return subsets;
}
int main(){
   ll m,temp;
   cin>>m;
   vector<ll>v;
   for(ll i=0;i<m;i++){
      cin>>temp;
      v.push_back(temp);
   }
   auto allsubs = subsets(v);
   for(auto ele : allsubs){
      for(ll hell : ele){
      cout<<hell<<" ";
   }
   cout<<endl;
  }

}
```

For string:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve(vector<string>&v){
   ll n=v.size();
   ll size=(1<<n);
   for(ll i=0;i<size;i++){
      vector<string>h;
      for(ll j=0;j<n;j++){
         if((i&(1<<j))!=0){
            h.push_back(v[j]);
         }
      }
      for (const string& element : h) {
         cout << element << " ";
      }
      cout<<endl;
   }

}

int main(){
   ll m,temp;
   string s;
   vector<string>v;
      cin>>s;
      for(ll i=0;i<s.length();i++){
      string temp;
      //converting char into string
      temp=temp+s[i];
      v.push_back(temp);
      }
   solve(v);

}
```

Q13)Print binary of the number

```
#include<bits/stdc++.h>
#define ll long long
```

```
using namespace std;
void solve(ll m){
   for(ll i=10;i>=0;i--){
      cout<<((m<<i)&1);
   }
   cout<<endl;
}
int main(){
   ll n,m;
   cin>>m;
   solve(m);
}
```

Q14)short tricks with bit manipulation
--odd even check,divide multiply by 2,upper case lower case conversion and toggling

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve(ll m){
   //odd-even check
   if(m&1!=0){
      cout<<"odd"<<endl;
   }else{
      cout<<"even"<<endl;
   }
   //divide or multiply by 2
   cout<<(m>>1)<<endl;//divide
   cout<<(m<<1)<<endl;//multiply
   //conversion from upper case to lower case
   //difference between A=1000001 a=1100001
   //basically toggle and change
   char A='A';
   char b=A|(1<<5);//toggling
   cout<<b<<endl;
   char B='a';
   char C=(B&(~(1<<5)));//toggling
   cout<<C<<endl;

   //shortrick
   char d='A'|32;
   cout<<d<<endl;
   char e='a'^32;
   cout<<e<<endl;
}
int main(){
   ll n,m;
   cin>>m;
   solve(m);
}
```

Q15)check LSB,MSB,Swap operation

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve3(ll m){
   for(ll i=10;i>=0;i--){
      cout<<((m>>i)&1);
   }
   cout<<endl;
}

void solve1(ll a){
 //clear LSB
 int i=4;
 solve3(a);
 int k=(a&(~((1<<i+1)-1)));
 solve3(k);
 //clear MSB
 int j=(a&((1<<i+1)-1));
 solve3(j);
}
void solve2(ll a,ll b){
   //swap
   a=a^b;
   b=b^a;
   a=a^b;
   cout<<a<<" "<<b;
}
int main(){
  ll a,b;
  cin>>a>>b;
  solve1(a);
  solve2(a,b);
}
```

BIT-MASKING

Q16)input

5(total workers)
4(total days first worker has worked)
1,4,7,9..
6
2,9,1,7,25,29
7
1,23,4,7,9,11,29

Etc…

Find the maximum intersection of two workers..

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;

int main(){
   ll n,m,mask;
   cin>>m;
   vector<int>masks(n,0);//making it intial to 0
   for(ll i=0;i<m;i++){
      ll workers;
```

```
    cin>>workers;
    for(ll i=0;i<workers;i++){
      int day;
      cin>>day;
      mask=(mask|(1<<day));//bitmasking
    }
    masks[i]=mask;//storing the mask
  }
  //intersection
  ll maxdays=0,person1,person2;
  for(ll i=0;i<n;i++){
    for(ll j=i+1;j<n;j++){
      ll intersection=(masks[i]&masks[j]);
      ll
common=__builtin_popcount(intersection);
      //how to know which is common..
      if(common>maxdays){
        maxdays=common;
        person1=i;
        person2=j;
      }

    }
  }
  cout<<person1<<" "<<person2<<" "<<maxdays;

}
```

Q17)Product of two xor operators ex-A^B=C find the product of such that the product is maximum

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;

int main(){
  ll m,temp,n;
  cin>>n;
  int k=(int)log2(n)+1;//ex-2 has 2 bits needed
  //32 bit is there but ex-c=10^5 so 16 bit
  vector<ll>v;//checks the number of set bits
  ll a=0,b=0;
  for(ll i=0;i<k;i++){
    if((n&(1<<i))){
      v.push_back(i);
    }else{//we know the 0 case will be 1,1 for both
      a|=(1<<i);
      b|=(1<<i);
    }
  }
 //we generate subset for the rest cases..
 ll size=1<<v.size();
 ll ans=-1;
 for(ll i=0;i<size;i++){
   ll acopy=a,bcopy=b;
   for(ll j=0;j<v.size();j++){
     if((i&(1<<j))!=0){
       acopy|=(1<<v[j]);
     }else{
       bcopy|=(1<<v[j]);
     }
   }
   ans=max(ans,acopy*1LL*bcopy);
 }

  cout<<ans;

}
```