**struct** Walsh\_Hadamard{  
  
 **long** **long** P1[MAX], P2[MAX];  
  
 **void** walsh\_transform(**long** **long**\* ar, **int** n, **int** flag , **int** invFlag){  
 **if** (n == 0) **return**;  
  
 **int** i, m = n / 2;  
 walsh\_transform(ar, m, flag, invFlag);  
 walsh\_transform(ar + m, m, flag, invFlag);  
  
 **for** (i = 0; i < m; i++){ */// Don't forget modulo if required*  
 **long** **long** x = ar[i], y = ar[i + m];  
 **if** (flag == OR) ar[i] = x, ar[i + m] = invFlag\*x + y;  
 **if** (flag == AND) ar[i] = x + invFlag\*y, ar[i + m] = y;  
 **if** (flag == XOR){  
 ar[i] = x + y, ar[i + m] = x - y;  
 **if**(invFlag==-1) ar[i]>>=1, ar[i+m]>>=1;  
 }  
 }  
 }  
  
 vector <**long** **long**> convolution(**int** n, **long** **long**\* A, **long** **long**\* B, **int** flag = XOR){  
 assert(\_\_builtin\_popcount(n) == 1); */// n must be a power of 2*  
 **for** (**int** i = 0; i < n; i++) P1[i] = A[i];  
 **for** (**int** i = 0; i < n; i++) P2[i] = B[i];  
  
 walsh\_transform(P1, n, flag, 1);  
 walsh\_transform(P2, n, flag, 1);  
 **for** (**int** i = 0; i < n; i++) P1[i] = P1[i] \* P2[i];  
 walsh\_transform(P1, n, flag, -1);  
 **return** vector<**long** **long**> (P1, P1 + n);  
 }  
  
 */// For i = 0 to n - 1, j = 0 to n - 1*  
 */// v[i or j] += A[i] \* B[j]*  
 vector <**long** **long**> or\_convolution(**int** n, **long** **long**\* A, **long** **long**\* B){  
 **return** convolution(n, A, B, OR);  
 }  
  
 */// For i = 0 to n - 1, j = 0 to n - 1*  
 */// v[i and j] += A[i] \* B[j]*  
 vector <**long** **long**> and\_convolution(**int** n, **long** **long**\* A, **long** **long**\* B){  
 **return** convolution(n, A, B, AND);  
 }  
  
 */// For i = 0 to n - 1, j = 0 to n - 1*  
 */// v[i xor j] += A[i] \* B[j]*  
 vector <**long** **long**> xor\_convolution(**int** n, **long** **long**\* A, **long** **long**\* B){  
 **return** convolution(n, A, B, XOR);  
 }  
  
}wh;