Problem I
Digits of Unity
Time limit: 6 seconds

At the beginning of the school year, the students in the International College of Paper Cutters (ICPC) choose their student IDs. The students can choose any positive integer less than or equal to some maximum number for their IDs, but no two students can choose the same student ID.

After some deliberation among the ranks, the students decided they wanted to find some common ground between all their IDs. In particular, they want to choose their IDs such that the bitwise AND of all of their student IDs has at least some minimum number of 1-bits. The students of the ICPC are asking you to write a program to compute the number of ways to do this. Two assignments are different if there is at least one student that has a different student ID in each assignment.

The bitwise AND of two integers a and b is an integer c whose binary representation is as follows: the ith bit of c is 1 if and only if the ith bits of both a and b are 1. C, C++, Java, and Python all support computing the bitwise AND of two integers using the `&` operator.

This definition generalizes to sets of numbers. The bitwise AND of a set of integers S is an integer c whose binary representation is as follows: the ith bit of c is 1 if and only if the ith bit of each element of S is 1.

Input

The single line of input contains three integers n ($1 \leq n \leq 5 \times 10^5$), k ($1 \leq k \leq 5 \times 10^5$), and m ($n \leq m \leq 5 \times 10^6$), where n is the number of students, k is the required minimum number of common bits, and m is the maximum number any student ID could be.

Output

Output a single integer, which is the number of ways to choose n distinct student IDs from the range $[1, m]$ such that the number of 1-bits in the bitwise AND of the student IDs has at least k 1-bits. Since the answer may be large, output it modulo 998,244,353.

Sample Explanation

There are 2 students, they want the bitwise AND of their student IDs to have at least 2 1-bits and the maximum allowed student ID is 10. The valid ID assignments are $\{(3, 7), (5, 7), (6, 7), (7, 3), (7, 5), (7, 6)\}$.

Sample Input 1	Sample Output 1
2 2 10	6

Sample Input 2	Sample Output 2
3 4 14	0

Sample Input 3	Sample Output 3
2 1 100000	910073387