### 组合数

int C(int n, int k, const vector <long long> &fact, const vector <long long> &inv) {  
 if (k > n) return 0;  
 int multiply = (1LL \* fact[n] \* inv[k]) % MOD;  
 multiply = (1LL \* multiply \* inv[n - k]) % MOD;  
 return multiply;  
}  
  
vector <long long> fact(n + 1, 1LL);  
vector <long long> inv(n + 1, 1LL);  
for (int i = 1; i <= n; ++i) {  
 fact[i] = (fact[i - 1] \* i) % MOD;  
 inv[i] = binPow(fact[i], MOD - 2);  
}  
  
int countLess = C(can\_less, cnt\_less, fact, inv);  
int countBig = C(can\_big, cnt\_big, fact, inv);

#include <bits/stdc++.h>  
using namespace std;  
const int MOD = 1e9 + 7;  
  
int binPow(int a, int n) {  
 int res = 1;  
 while (n) {  
 if (n & 1)  
 res = (1LL \* res \* a) % MOD;  
 a = (1LL \* a \* a) % MOD;  
  
 n >>= 1;  
 }  
 return res;  
}  
  
void binarySearch(int n, int x\_position, int &cnt\_big, int &cnt\_less) {  
 int left = 0, right = n;  
  
 while(left < right) {  
 int middle = (left + right) / 2;  
 if (x\_position >= middle) {  
 if (x\_position != middle) cnt\_less++;  
 left = middle + 1;  
 }  
 else if (x\_position < middle){  
 cnt\_big++;  
 right = middle;  
 }  
 }  
}  
  
int C(int n, int k, const vector <long long> &fact, const vector <long long> &inv) {  
 if (k > n) return 0;  
 int multiply = (1LL \* fact[n] \* inv[k]) % MOD;  
 multiply = (1LL \* multiply \* inv[n - k]) % MOD;  
 return multiply;  
}  
  
int main() {  
 ios\_base::sync\_with\_stdio(false);  
 cin.tie(nullptr);  
  
 int n, x, x\_position;  
 long long ans = 0;  
  
  
 cin >> n >> x >> x\_position;  
 vector <long long> fact(n + 1, 1LL);  
 vector <long long> inv(n + 1, 1LL);  
 for (int i = 1; i <= n; ++i) {  
 fact[i] = (fact[i - 1] \* i) % MOD;  
 inv[i] = binPow(fact[i], MOD - 2);  
 }  
  
 int cnt\_big = 0, cnt\_less = 0;  
 binarySearch(n, x\_position, cnt\_big, cnt\_less);  
  
 int other = (n - cnt\_big - cnt\_less - 1);  
 int can\_big = n - x, can\_less = x - 1;  
  
 int countLess = C(can\_less, cnt\_less, fact, inv);  
 int countBig = C(can\_big, cnt\_big, fact, inv);  
  
 countBig = (1LL \* countBig \* fact[cnt\_big]) % MOD;  
 countLess = (1LL \* countLess \* fact[cnt\_less]) % MOD;  
  
 int multiply = (1LL \* countBig \* countLess) % MOD;  
 multiply = (1LL \* multiply \* fact[other]) % MOD;  
  
 ans = (ans + multiply) % MOD;  
  
 cout << ans << endl;  
  
 return 0;  
}