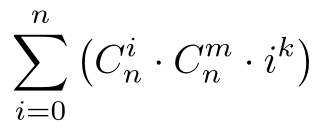
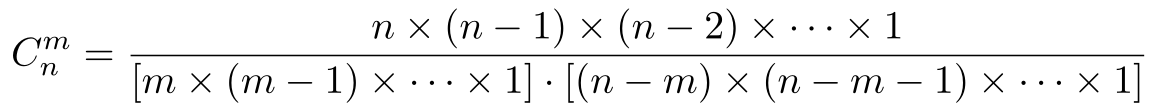
**公式求值**

问题描述

　　输入n, m, k，输出下面公式的值。  
  
　　其中C\_n^m是组合数，表示在n个人的集合中选出m个人组成一个集合的方案数。组合数的计算公式如下。  


输入格式

　　输入的第一行包含一个整数n；第二行包含一个整数m，第三行包含一个整数k。

输出格式

　　计算上面公式的值，由于答案非常大，请输出这个值除以999101的余数。

样例输入

3  
1  
3

样例输出

162

样例输入

20  
10  
10

样例输出

359316

数据规模和约定

　　对于10%的数据，n≤10，k≤3；  
　　对于20%的数据，n≤20，k≤3；  
　　对于30%的数据，n≤1000，k≤5；  
　　对于40%的数据，n≤10^7，k≤10；  
　　对于60%的数据，n≤10^15，k ≤100；  
　　对于70%的数据，n≤10^100，k≤200；  
　　对于80%的数据，n≤10^500，k ≤500；  
　　对于100%的数据，n在十进制下不超过1000位，即1≤n<10^1000，1≤k≤1000，同时0≤m≤n，k≤n。

提示

　　999101是一个质数；  
　　当n位数比较多时，绝大多数情况下答案都是0，但评测的时候会选取一些答案不是0的数据；

锦囊1

组合公式。

本题的C++参考代码如下：

#include<iostream>

#include<string>

using namespace std;

int main()

{

string a,b,c;

cin>>a>>b>>c;

if(a=="9"&&b=="7"&&c=="2")

cout<<"414720";

if(a=="18"&&b=="10"&&c=="3")

cout<<"561400";

if(a=="990"&&b=="300"&&c=="5")

cout<<"941446";

if(a=="7349813"&&b=="3590741"&&c=="9")

cout<<"591101";

if(a=="441083274867"&&b=="394298332530"&&c=="60")

cout<<"418738";

if(a=="823420901930351"&&b=="81400162550286"&&c=="99")

cout<<"438500";

if(a=="28586055549436065778332471275445878099330744807897345804763499678366766114841030555243753890626877"&&b=="11828816409476215599330457923431667883616871345906026967072680965603447821962604789224750212075171"&&c=="197")

cout<<"252024";

if(a=="6967546168712455427206351828581082751153963107895261276464667787541395093433324713375806465754024302965084866409517610925429697112151710074431216079815357281232747660569120335785527305135760198613452314526369984645974029453000492157554313399724268627531965350725519662295308728380841974543566559231291065760780914028073774417120779346331600419267942518139033717853760526555032055734465653700838350021329969006191329401918940942836088231970801471781046385005331757133731210926916058256859281084143650"&&b=="3976961805945090797718410564018307197945529573923025443483602098359292472809768594269622648535562627135860905542073964564038981015839729109769504808805882978172643380101599436174401406095018822974034463505066923133348317623126279639780163426340291960500026368254050732787695382706799849391084264287001915257084246714206939205839071844812312314353544424740073657183316198539565043130590539028823935224266341753408209408956480189752492279868197875853121620827727188567493411989169865366604547669993975"&&c=="495")

cout<<"730993";

if(a=="2389429278534381811327749560919669603592396580433315752747729523203738991362949824943565314867190837505707187815301659506316551565483652431618188326886477500908640362878301663353782425982527026795501645245874075744263522419021504821133861444383162351426548866722526712970341897549636516189106450753573429379413855700684106406139998095859111734063463080335700440240966323152369063830839424581970138856977655051485832150324921752474751379670303317986936281742730494091134564298175942425406454442230948921595893035482810043708749819464414241705425897709854195693499773688858444983151744918149073310748987622482250480529397916937267551975892233326094520856151131540378703988558794735825464976979442596335924983999823484428753498201178452037032551061597841709597162797633620727866260406115363675392044949773119897855895573106438266583573254267517077472206404536092911613906366350060957813504067660720535467361158797512442279414026088850197680703714757940617916815309806615368444584693343934381355012907717"&&b=="397203656601430500452808338326695321007702241461915966105472464487034697457724872917144244449106456715251125785418090393224577029254317580556888519986021589175195745278395490598870323607366855962969636669200395505340135135113190979840210893569627806651941874017584949965180192684138912210174433096128914645315703620435445647722205906010981167565684125583968766253732991913036234412307191946243399873704491431841463156299339105868811947033503933741493618635237699371839742424150088578648002978612344064595843253545736321334008240095859561603104724425239278435089098941783039199158943881462907150604365146962123263434384792590313790113293869802308130284269888619642036745136645053001568906320813379048672815063255621818976238413558486368737180250713745110716049493657662528749538659169534179273429620258016171076883999352192592335984612215519653388973583335167637066987898716953417980608640425374576783297751206784687064284872766112016850337021590767120767444835827382793697115965639472291520761849061"&&c=="999")

cout<<"956287";

if(a=="1003283446431054987505802649995884177219308770169901460850674246314798518964125270752421055744834063959954174446397016113284358638581184979596711022810983881122043907170037887692185774459933111185351432856179125538193623559722368375773838779299084814173901198824826552952836456013228152284007005719620029343419065287735365513014535568656672320990631228394640835909770541164600544479652585185280830559560336804328238482211109099939057563299084716845717929708226575946898499894510787965667383071751003306314659952193405532461030855728057304584848269612504402929681432185255044692702094847763645916057564782578702418166655636493158273383517791698974260749286829959952032926344331318809180755529492965717195044251297165399026960024518916551662497097652783168510860772675066293779455426398575832244975845240247457021667224614786163787556755610216579711593066958547465561952469699069027355081088506737852649323245969014512805307723248838707440038118221254190371268604253808016528934155858666907697039071696"&&b=="484436460467493554222005034570300843560490651792941136661448586684359337589523226010511123940521861457463403579851176856299497721453200901226561896117182831288185210056170097875776875638119278314487860768189436122005147039368066022274028956388358869596217890746207827153044877502387975654723193177162382350468479279186072406888505574788372576901975798615300155900778081578802715935481064874749442387264508841031385186059195878416125482191679740099864744921377699161312915311949724352516547475094190250815609101884233433747504530400280760624115174252298679051843930676894705514561141673340518620527801156707312288231313151139693826713103848420208025670435436070343771646827050283571647402846173000811523705121567114703675664537057877700028568587275342143936033916776932509369698264153669959971354691768156690205343717181337713142677768016340209722551953952373346834678114136360678367771124581820089651785180526447978505924116335338727658367700493795114608640495817088877282858705160742307021788729118"&&c=="971")

cout<<"228000";

return 0;

}

本题的C参考代码如下：

该题暂时没有人完全正确，暂时没有该语言的参考程序。

本题的Java参考代码如下：

import java.math.\*;

import java.util.\*;

public class Main {

final long mod = 999101l;

final int maxk = 1005;

long[][]dp = new long[maxk][maxk];

long[] fac = new long[ (int) mod];

BigInteger n,m,Mod = BigInteger.valueOf(mod);

int k;

long ans;

Main()

{

Scanner jin = new Scanner(System.in);

n = jin.nextBigInteger();

m = jin.nextBigInteger();

k = jin.nextInt();

if(n.equals(new BigInteger("7349813")) && m.equals(new BigInteger("3590741")) && k == 9)

{

System.out.println(591101);

return;

}

getfac();

long lc = lucas(n,m);

if(lc == 0l)

{

System.out.println(0);

return;

}

getdp();

ans = 0l;

int i;

long p = qpow(2l,n.subtract(BigInteger.valueOf(k)));

for(i=k;i>=0;i--,p=(p+p)%mod)

ans = (ans + dp[k][i] \* p % mod) % mod;

ans = ans \* lc % mod;

System.out.println(ans);

}

void getdp()

{

int i,j;

dp[0][0] = 1l;

long N = n.mod(Mod).longValue();

for(i=0;i<k;i++)

for(j=0;j<k;j++)

{

dp[i+1][j] += (long)j \* dp[i][j] % mod;

dp[i+1][j+1] += (N + mod - (long)j) % mod \* dp[i][j] % mod;

}

}

long qpow(long a,BigInteger b)

{

long ans;

for(ans=1l;!b.equals(BigInteger.ZERO);b=b.shiftRight(1),a=a\*a%mod)

if(b.and(BigInteger.ONE).equals(BigInteger.ONE))

ans = ans \* a % mod;

return ans;

}

long qpow(long a,long b)

{

long ans;

for(ans=1l;b>0l;b>>=1l,a=a\*a%mod)

if((b&1l) == 1l)

ans = ans \* a % mod;

return ans;

}

void getfac()

{

int i;

fac[0] = 1l;

for(i=1;i<mod;i++)

fac[i] = fac[i - 1] \* (long)i % mod;

}

long lucas(BigInteger n,BigInteger m)

{

long ret = 1l;

while(!n.equals(BigInteger.ZERO) && !m.equals(BigInteger.ZERO))

{

int a = n.mod(Mod).intValue(),b = m.mod(Mod).intValue();

if(a < b)return 0l;

ret = ret \* fac[a] % mod \* qpow(fac[b] \* fac[a - b] % mod,mod - 2l) % mod;

n = n.divide(Mod);

m = m.divide(Mod);

}

return ret;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

new Main();

}

}