**算法训练 整除问题**

问题描述

　　编写一个程序，输入三个正整数min、max和factor，然后对于min到max之间的每一个整数（包括min和max），如果它能被factor整除，就把它打印出来。  
　　输入格式：输入只有一行，包括三个整数min、max和factor。  
　　输出格式：输出只有一行，包括若干个整数。  
　　输入输出样例

样例输入

1 10 3

样例输出

3 6 9

本题的C++参考代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Powered by Graphene Richards\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//{

#define OUTPUT\_PRECISION "%.2f"

#define LF\_PRECISION 10

#define INT\_64\_MOD "%I64d"

#define UNSIGNED\_64\_MOD "%I64u"

#define CONTEST\_EXIST 1

#define CONTEST\_START\_HOUR 17

#define CONTEST\_START\_MIN 0

#define CONTEST\_START\_SEC 0

#define CONTEST\_DUR\_HOUR 2

#define CONTEST\_DUR\_MIN 30

#define CONTEST\_DUR\_SEC 0

//#pragma comment(linker,"/STACK:102400000,102400000")

#include<cmath>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<bitset>

#include<complex>

#include<vector>

#include<iomanip>

#include<iostream>

#include<list>

#include<map>

#include<queue>

#include<set>

#include<stack>

#include<string>

#include<typeinfo>

#define FAST\_RW ios\_base::sync\_with\_stdio(0),cin.tie(0);

#define ll long long

#define lf long double

#define IT(x) \_\_typeof((x).begin())

#define FS(i,a) for(int i=0;a[i];i++)

#define FE(x,ctn) for(IT(ctn)x=(ctn).begin(),CluhxSchFuDeugk=(ctn).end();x!=CluhxSchFuDeugk;x++)

#define FR(i,en) for(int i=0,pJNwFPtlXiwFoIv=(en);i<pJNwFPtlXiwFoIv;i++)

#define FOR(i,en) for(int i=1,SbKCIcakJTeYVqs=(en);i<=SbKCIcakJTeYVqs;i++)

#define FFR(i,x,y) for(int i=(x),alVDbhLBoMEGSwA=(y);i<=alVDbhLBoMEGSwA;i++)

#define DFFR(i,x,y) for(int i=(x),NWYfecAcmGBMJuU=(y);i>=NWYfecAcmGBMJuU;i--)

#define mp make\_pair

#define pb push\_back

#define pq priority\_queue

#define fi first

#define se second

#define lb(x) (x&(-x))

#define sqr(x) (x)\*(x)

#define all(x) (x).begin(),(x).end()

#define clr(x) memset((x),0,sizeof(x))

#define inf(x) memset((x),0x7F,sizeof(x))

#define \_inf(x) memset((x),0x80,sizeof(x))

#define NL puts("");

using namespace std;

template<class T1,class T2,class T3>

bool \_IN(T1 x,T2 y,T3 z){

return x<=y&&x>=z||x<=z&&x>=y;

}

template<class T>

T gcd(T a,T b){

if(!b)

return a;

while(b^=a^=b^=a%=b);

return a;

}

#ifdef wmx16835

#include<ctime>

#include<windows.h>

#define GREEN 10

#define BLUE 11

#define RED 12

#define PINK 13

#define YELLOW 14

#define WHITE 15

struct TM{

int t;

TM(int a,int b,int c):t(a\*3600+b\*60+c){}

TM(int x):t(x){}

TM(){}

operator int(){return t;}

void print(){

printf("%02d:%02d:%02d",t/3600%24,t/60%60,t%60);

}

};

void OVZmetNNpAqAVZx(){

//\_\_\_\_\_\_\_\_\_\_;

time\_t t=time(0);

srand(t);

tm\*p=localtime(&t);

TM day(24,0,0),cur(p->tm\_hour,p->tm\_min,p->tm\_sec),beg(CONTEST\_START\_HOUR,CONTEST\_START\_MIN,CONTEST\_START\_SEC),len(CONTEST\_DUR\_HOUR,CONTEST\_DUR\_MIN,CONTEST\_DUR\_SEC),en(beg+len),fen(beg+en+day>>1),seg;

if(beg>cur)

cur.t+=day;

puts(" +--------------------------+");

if(\_IN(cur,beg,en)&&CONTEST\_EXIST){

puts(" | Status: Running |");

printf(" | Progress: %5.1f %% |\n",(double)(cur-beg)/len\*100);

seg.t=en-cur;

printf(" | Remaining time: ");

seg.print();

puts(" |");

}

else{

printf(" | Current time: ");

cur.print();

puts(" | ");

if(cur<=fen||!CONTEST\_EXIST)

puts(" | Status: Ended |");

else{

puts(" | Status: Scheduled |");

cur.t-=day;

seg.t=beg-cur;

printf(" | Remaining time: ");

seg.print();

puts(" |");

}

}

puts(" +--------------------------+");

puts("................................\n");

}

const HANDLE hout=GetStdHandle(STD\_OUTPUT\_HANDLE);

void \_COLOR(unsigned x=BLUE){

SetConsoleTextAttribute(hout,x);

}

void pCJUYxWEXKaDOIC(){

int num=1;

FILE\*p=fopen("FILE\_NAME","r");

if(p){

fscanf(p,"%d",&num);

fclose(p);

}

char f\_name[1024];

sprintf(f\_name,"%d.out",num++);

\_COLOR(PINK);

printf("[NOTICE] Standard output has been redirected to \"%s\".\n",f\_name);

\_COLOR();

freopen(f\_name,"w",stdout);

if(!stdout)

exit(1);

else

p=fopen("FILE\_NAME","w");

if(p){

fprintf(p,"%d",num);

fclose(p);

}

}

#define LOG {\_COLOR(RED);unsigned ijqQjAZZNYpJYBe=clock();puts("\n----------------------");\_COLOR();

#define TEL \_COLOR(RED),printf("\n----------------------\nExecution time: %ums\n----------------------\n",clock()-ijqQjAZZNYpJYBe);\_COLOR();}

#define SHOW\_TIME \_COLOR(GREEN);OVZmetNNpAqAVZx();\_COLOR();

#define test(...) \_COLOR(YELLOW),printf(\_\_VA\_ARGS\_\_),\_COLOR()

#define TEST(x) \_COLOR(YELLOW),printf("%s = ",#x),\_P(x),puts(""),\_COLOR()

#define PF pCJUYxWEXKaDOIC();

#else

#define LOG

#define TEL

#define SHOW\_TIME

#define test(...)

#define TEST(...)

#define PF

#endif

int S(char\*a){

return scanf("%s",a)==1;

}

template<class T>

int S(T&a){

const char\*x=typeid(a).name();

if(!strcmp(x,"i")||!strcmp(x,"b"))return scanf("%d",&a)==1;

else if(!strcmp(x,"j"))return scanf("%u",&a)==1;

else if(!strcmp(x,"c"))return scanf("%c",&a)==1;

else if(!strcmp(x,"Pc")||\*x=='A')return scanf("%s",a)==1;

else if(!strcmp(x,"f"))return scanf("%f",&a)==1;

else if(!strcmp(x,"d"))return scanf("%lf",&a)==1;

else if(!strcmp(x,"x"))return scanf(INT\_64\_MOD,&a)==1;

else if(!strcmp(x,"y"))return scanf(UNSIGNED\_64\_MOD,&a)==1;

else if(!strcmp(x,"e"))return (cin>>a)!=0;

else test("Input format error!\n");

}

template<class T>

void \_P(T a){

const char\*x=typeid(a).name();

if(!strcmp(x,"i")||!strcmp(x,"b"))printf("%d",a);

else if(!strcmp(x,"j"))printf("%u",a);

else if(!strcmp(x,"c"))printf("%c",a);

else if(!strcmp(x,"Pc")||!strcmp(x,"PKc")||\*x=='A')printf("%s",a);

else if(!strcmp(x,"d")||!strcmp(x,"f"))printf(OUTPUT\_PRECISION,a);

else if(!strcmp(x,"x"))printf(INT\_64\_MOD,a);

else if(!strcmp(x,"y"))printf(UNSIGNED\_64\_MOD,a);

else if(!strcmp(x,"e"))cout<<setprecision(LF\_PRECISION)<<a;

else test("Output format error!\n");

}

template<class T1,class T2>

int S(T1&a,T2&b){

return S(a)+S(b)==2;

}

template<class T1,class T2,class T3>

int S(T1&a,T2&b,T3&c){

return S(a)+S(b)+S(c)==3;

}

template<class T1,class T2,class T3,class T4>

int S(T1&a,T2&b,T3&c,T4&d){

return S(a)+S(b)+S(c)+S(d)==4;

}

template<class T1,class T2,class T3,class T4,class T5>

int S(T1&a,T2&b,T3&c,T4&d,T5&e){

return S(a)+S(b)+S(c)+S(d)+S(e)==5;

}

template<class T>

void P(T a){

\_P(a);

putchar(' ');

}

template<class T1,class T2>

void P(T1 a,T2 b){

\_P(a);putchar(' ');

\_P(b);putchar(' ');

}

template<class T>

void PN(T a){

\_P(a);

NL

}

template<class T1,class T2>

void PN(T1 a,T2 b){

\_P(a);putchar(' ');

\_P(b);NL

}

template<class T1,class T2,class T3>

void PN(T1 a,T2 b,T3 c){

\_P(a);putchar(' ');

\_P(b);putchar(' ');

\_P(c);NL

}

template<class T1,class T2,class T3,class T4>

void PN(T1 a,T2 b,T3 c,T4 d){

\_P(a);putchar(' ');

\_P(b);putchar(' ');

\_P(c);putchar(' ');

\_P(d);NL

}

template<class T1,class T2,class T3,class T4,class T5>

void PN(T1 a,T2 b,T3 c,T4 d,T5 e){

\_P(a);putchar(' ');

\_P(b);putchar(' ');

\_P(c);putchar(' ');

\_P(d);putchar(' ');

\_P(e);NL

}

int kase;

const double pi=4\*atan(1);

const double ep=1e-9;

//}

int main(){

SHOW\_TIME

int mi,ma,f;

S(mi,ma,f);

FFR(i,mi,ma)if(i%f==0)P(i);

}

/\*\*\*\*\*\*\*\*\*Risoft corporation all rights reserved\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Template V1.33 build 20150104\*\*\*\*\*\*\*\*\*\*\*\*\*/

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int min, max , factor;

scanf("%d%d%d",&min,&max,&factor);

int i;

int res;

for(i=1;;i++)

{

res = factor\*i;

if(res<=max && res>=min)

{

printf("%d ",res);

}

else if(res>max)

break;

}

return 0;

}

**算法训练 薪水计算**

问题描述

　　编写一个程序，计算员工的周薪。薪水的计算是以小时为单位，如果在一周的时间内，员工工作的时间不超过40 个小时，那么他/她的总收入等于工作时间乘以每小时的薪水。如果员工工作的时间在40 到50 个小时之间，那么对于前40 个小时，仍按常规方法计算；而对于剩余的超额部分，每小时的薪水按1.5 倍计算。如果员工工作的时间超过了50 个小时，那么对于前40 个小时，仍按常规方法计算；对于40～50 个小时之间的部分，每小时的薪水按1.5 倍计算；而对于超出50 个小时的部分，每小时的薪水按2 倍计算。请编写一个程序，输入员工的工作时间和每小时的薪水，然后计算并显示他/她应该得到的周薪。  
　　输入格式：输入只有一行，包括一个整数和一个实数，分别表示工作时间和每小时薪水。  
　　输出格式：输出只有一个实数，表示周薪，保留小数点后2位。  
　　输入输出样例

样例输入

40 50

样例输出

2000.00

本题的C++参考代码如下：

#include<iostream>

#include<stdio.h>

using namespace std;

int main()

{

double t;

double a;

double m=0;

cin>>t>>a;

if (t<=40){

m=t\*a;

}

if (t>40&&t<=50){

m=a\*40+(t-40)\*a\*1.5;

}

if (t>50){

m=a\*40+10\*a\*1.5+(t-50)\*a\*2;

}

printf("%.2f",m);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

float a,b,sum=0.0;

scanf("%f%f",&a,&b);

if(a<=40) sum=a\*b;

else if(a>40 && a<=50) sum=40\*b+(a-40)\*1.5\*b;

else if(a>50) sum=40\*b+10\*1.5\*b+(a-50)\*2\*b;

printf("%.2f",sum);

return 0;

}

**算法训练 数位分离**

问题描述

　　编写一个程序，输入一个1000 以内的正整数，然后把这个整数的每一位数字都分离出来，并逐一地显示。  
　　输入格式：输入只有一行，即一个1000以内的正整数。  
　　输出格式：输出只有一行，即该整数的每一位数字，之间用空格隔开。  
　　输入输出样例

样例输入

769

样例输出

7 6 9

本题的C++参考代码如下：

#include <iostream>

using namespace std;

int main()

{

int a,b[5],l=0,i;

cin>>a;

while(a>0)

{

b[l]=a%10;

l++;

a/=10;

}

for(i=l-1;i>=0;i--)

cout<<b[i]<<" ";

return 0;

}

本题的C参考代码如下：

int main()

{

int n,m,i=0;

int ab[10];

scanf("%d",&n);

while(n)

{

ab[i++]=n%10;

n=n/10;

}

for(m=i-1;m>=0;m--)

{

if(m!=0)

printf("%d ",ab[m]);

else

printf("%d\n",ab[m]);

}

return 0;

}

**算法训练 Hello World!**

描述

　　本题定义本学期作业题的输出格式，请认真阅读。  
　　如无特殊说明，开头无空格，间隔符为1个空格，答案最后必须输出换行符("\n")。

输入格式

　　无

输出格式

　　Hello World!

本题的C++参考代码如下：

#include<iostream>

using namespace std;

int main()

{

cout<<"Hello World!"<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

printf("Hello World!\n");

return 0;

}

**算法训练 新生舞会**

**问题描述**  
　　新生舞会开始了。n名新生每人有三个属性：姓名、学号、性别。其中，姓名用长度不超过20的仅由大小写字母构成的字符串表示，学号用长度不超过10的仅由数字构成的字符串表示，性别用一个大写字符‘F’或‘M’表示。任意两人的姓名、学号均互不相同。换言之，每个人可被其姓名或学号唯一确定。给出m对两人的信息（姓名或学号），判断他们是否能共舞。两人能共舞的充要条件为两人性别相异。  
  
**输入**  
　　第一行一个整数n（2<=n<=1000），表示学生人数。接下来的n行每行依次包含一名新生的姓名、学号、性别，分别用一个空格隔开。  
　　之后的一行是一个整数m(1<=m<=1000)，表示询问的数目。接着的m行每行包含两个信息（姓名或学号），保证两个信息不属于同一人，中间用一个空格隔开。  
  
**输出**  
　　对于每个询问输出一行，如果两人可以共舞，输出一个大写字母‘Y’，否则输出一个大写字母‘N’。  
  
  
**样例输入**  
　　4  
　　John 10 M  
　　Jack 11 M  
　　Kate 20 F  
　　Jim 21 M  
　　3  
　　John 11  
　　20 Jack  
　　Jim Jack  
  
  
**样例输出**  
　　N  
　　Y  
　　N  
  
**提示**  
　　可以把名字和学号都当成字符串处理。可以按以下流程实现。  
  
　　#include<iostream>  
　　#include<cstring>  
　　using namespace std;  
  
　　struct tstudent  
　　{  
　　char name[21];  
　　char num[21];  
　　char sex;  
　　};  
  
　　void readdata(tstudent student[], int n)  
　　{  
**输入N个学生的信息**  
　　}  
  
　　int findstudent(tstudent student[], int n, char\* data)  
　　{  
　　if (data == NULL) return -1;  
  
**判断是否有某个学生的学号或名字等于data，如果有，函数返回该学生在student数组中的序号，否则返回-1**  
　　}  
  
　　void solve(tstudent student[], int n, int m)  
　　{  
　　char x[21], y[21];  
　　for (int i=0; i<m; i++) {  
**输入两个人的信息X、Y。通过调用findstudent函数判断这两个人能否成为舞伴**  
　　}  
　　}  
  
　　int main()  
　　{  
　　int n, m;  
　　tstudent student[1010];  
  
　　cin>>n;  
  
　　readdata(student, n);  
　　cin>>m;  
　　solve(student, n, m);  
　　}

本题的C++参考代码如下：

#include<iostream>

#include<cstring>

using namespace std;

struct tstudent

{

char name[21];

char num[21];

char sex;

};

void readdata(tstudent student[], int n)

{

//输入N个学生的信息

int i;

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

{

cin>>student[i].name>>student[i].num>>student[i].sex;

}

}

int findstudent(tstudent student[], int n, char\* data)

{

if (data == NULL)

return -1;

//判断是否有某个学生的学号或名字等于data，如果有，函数返回该学生在student数组中的序号，否则返回-1

int i;

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

{

if(strcmp(student[i].name,data)==0 || strcmp(student[i].num,data)==0){

// cout<<student[i].name<<"\t"<<student[i].num<<"\t"<<student[i].sex<<"\t"<<data<<"\t"<<strcmp(student[i].name,data)<<"\t"<<strcmp(student[i].num,data)<<endl;

return i;

}

}

return -1;

}

void solve(tstudent student[], int n, int m)

{

char x[21], y[21];

for (int i=0; i<m; i++) {

//输入两个人的信息X、Y。通过调用findstudent函数判断这两个人能否成为舞伴

cin>>x>>y;

int a = findstudent(student,n,x);

int b = findstudent(student,n,y);

if(student[a].sex == student[b].sex)

cout<<"N"<<endl;

else

cout<<"Y"<<endl;

}

}

int main(void)

{

int n, m;

tstudent student[1010];

cin>>n;

readdata(student, n);

cin>>m;

solve(student, n, m);

return 0;

}

本题的C参考代码如下：

# include <stdio.h>

# include <string.h>

struct stu{

char name[21];

char num[21];

char sex;

};

int main()

{

int i,n,m,j;

char s1[20],s2[10];

char se1,se2;

struct stu stu1[1001];

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%s %s %c",&stu1[i].name,&stu1[i].num,&stu1[i].sex);

}

scanf("%d",&m);

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

{

scanf("%s %s",&s1,&s2);

for(j=1;j<=n;j++)

{

if(strcmp(s1,stu1[j].name)==0 || strcmp(s1,stu1[j].num)==0)

{

se1=stu1[j].sex;

break;

}

}

for(j=1;j<=n;j++)

{

if(strcmp(s2,stu1[j].name)==0 || strcmp(s2,stu1[j].num)==0)

{

se2=stu1[j].sex;

break;

}

}

if(se2==se1) printf("N\n");

else printf("Y\n");

}

return 0;

}

**算法训练 反置数**

问题描述

　　一个整数的“反置数”指的是把该整数的每一位数字的顺序颠倒过来所得到的另一个整数。如果一个整数的末尾是以0结尾，那么在它的反置数当中，这些0就被省略掉了。比如说，1245的反置数是5421，而1200的反置数是21。请编写一个程序，输入两个整数，然后计算这两个整数的反置数之和sum，然后再把sum的反置数打印出来。要求：由于在本题中需要多次去计算一个整数的反置数，因此必须把这部分代码抽象为一个函数的形式。  
　　输入格式：输入只有一行，包括两个整数，中间用空格隔开。  
　　输出格式：输出只有一行，即相应的结果。  
　　输入输出样例

样例输入

435 754

样例输出

199

本题的C++参考代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Powered by Graphene Richards\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//{

#define OUTPUT\_PRECISION "%.2f"

#define INT\_64\_MOD "%I64d" ///"%I64d" or "%lld"

#define UNSIGNED\_64\_MOD "%I64u" ///"%I64d" or "%lld"

#define CONTEST\_STARTED true

#define CONTEST\_START\_HOUR 11

#define CONTEST\_START\_MIN 0

#define CONTEST\_START\_SEC 0

#define CONTEST\_DUR\_HOUR 5

#define CONTEST\_DUR\_MIN 0

#define CONTEST\_DUR\_SEC 0

//#pragma comment(linker,"/STACK:102400000,102400000")

#include<cmath>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<bitset>

#include<vector>

#include<iomanip>

#include<iostream>

#include<list>

#include<map>

#include<queue>

#include<set>

#include<stack>

#include<string>

#include<typeinfo>

#define \_\_\_\_\_\_\_\_\_\_ ios\_base::sync\_with\_stdio(0),cin.tie(0);

#define ll long long

#define IT(x) \_\_typeof((x).begin())

#define FS(i,a) for(int i=0;a[i];i++)

#define FE(x,ctn) for(IT(ctn)x=(ctn).begin(),CluhxSchFuDeugk=(ctn).end();x!=CluhxSchFuDeugk;x++)

#define FR(i,en) for(int i=0,pJNwFPtlXiwFoIv=(en);i<pJNwFPtlXiwFoIv;i++)

#define FOR(i,en) for(int i=1,SbKCIcakJTeYVqs=(en);i<=SbKCIcakJTeYVqs;i++)

#define FFR(i,x,y) for(int i=(x),alVDbhLBoMEGSwA=(y);i<=alVDbhLBoMEGSwA;i++)

#define DFFR(i,x,y) for(int i=(x),NWYfecAcmGBMJuU=(y);i>=NWYfecAcmGBMJuU;i--)

#define mp make\_pair

#define pb push\_back

#define pq priority\_queue

#define fi first

#define se second

#define lb(x) (x&(-x))

#define sqr(x) (x)\*(x)

#define all(x) (x).begin(),(x).end()

#define clr(x) memset((x),0,sizeof(x))

#define inf(x) memset((x),0x7F,sizeof(x))

#define \_inf(x) memset((x),0x80,sizeof(x))

#define NL puts("");

using namespace std;

template<class T1,class T2,class T3>

bool IN(T1 x,T2 y,T3 z){

return x<=y&&x>=z||x<=z&&x>=y;

}

template<class T>

T gcd(T a,T b){

if(!b)

return a;

while(b^=a^=b^=a%=b);

return a;

}

#ifdef wmx16835

#include<ctime>

struct Time\_wmx{

int t;

Time\_wmx(int a,int b,int c):t(a\*3600+b\*60+c){}

Time\_wmx(int x):t(x){}

Time\_wmx(){}

operator int(){return t;}

void print(){printf("%02d:%02d:%02d",t/3600%24,t/60%60,t%60);}

};

void OVZmetNNpAqAVZx(){

//\_\_\_\_\_\_\_\_\_\_;

time\_t t=time(0);

srand(t);

tm\*p=localtime(&t);

Time\_wmx day(24,0,0),cur(p->tm\_hour,p->tm\_min,p->tm\_sec),beg(CONTEST\_START\_HOUR,CONTEST\_START\_MIN,CONTEST\_START\_SEC),len(CONTEST\_DUR\_HOUR,CONTEST\_DUR\_MIN,CONTEST\_DUR\_SEC),en(beg+len),fen(beg+en+day>>1),seg;

if(beg>cur)

cur.t+=day;

puts(" +--------------------------+");

if(IN(cur,beg,en)&&CONTEST\_STARTED){

puts(" | Status: Running |");

printf(" | Progress: %5.1f %% |\n",(double)(cur-beg)/len\*100);

seg.t=en-cur;

printf(" | Remaining time: ");

seg.print();

puts(" |");

}

else{

printf(" | Current time: ");

cur.print();

puts(" | ");

if(cur<=fen||!CONTEST\_STARTED)

puts(" | Status: Ended |");

else{

puts(" | Status: Scheduled |");

cur.t-=day;

seg.t=beg-cur;

printf(" | Remaining time: ");

seg.print();

puts(" |");

}

}

puts(" +--------------------------+");

puts("................................\n");

}

void pCJUYxWEXKaDOIC(){

int num=1;

FILE\*p=fopen("FILE\_NAME","r");

if(p){

fscanf(p,"%d",&num);

fclose(p);

}

char f\_name[1024];

sprintf(f\_name,"%d.out",num++);

printf("[NOTICE] Standard output has been redirected to \"%s\".\n",f\_name);

freopen(f\_name,"w",stdout);

if(!stdout)

exit(1);

else

p=fopen("FILE\_NAME","w");

if(p){

fprintf(p,"%d",num);

fclose(p);

}

}

#define LOG {unsigned ijqQjAZZNYpJYBe=clock();

#define TEL printf("\n----------------------\nExecution time: %ums\n----------------------\n",clock()-ijqQjAZZNYpJYBe);}

#define SHOW\_TIME OVZmetNNpAqAVZx();

#define test printf

#define PF pCJUYxWEXKaDOIC();

#else

#define LOG

#define TEL

#define SHOW\_TIME

#define test

#define PF

#endif

int S(char\*a){

return scanf("%s",a)==1;

}

template<class T>

int S(T&a){

const char\*x=typeid(a).name();

if(!strcmp(x,"i")||!strcmp(x,"b"))return scanf("%d",&a)==1;

else if(!strcmp(x,"j"))return scanf("%u",&a)==1;

else if(!strcmp(x,"c"))return scanf("%c",&a)==1;

else if(!strcmp(x,"Pc")||\*x=='A')return scanf("%s",a)==1;

else if(!strcmp(x,"f"))return scanf("%f",&a)==1;

else if(!strcmp(x,"d")||!strcmp(x,"e"))return scanf("%lf",&a)==1;

else if(!strcmp(x,"x"))return scanf(INT\_64\_MOD,&a)==1;

else if(!strcmp(x,"y"))return scanf(UNSIGNED\_64\_MOD,&a)==1;

else test("Input format error!\n");

}

template<class T>

void \_P(T a){

const char\*x=typeid(a).name();

if(!strcmp(x,"i")||!strcmp(x,"b"))printf("%d",a);

else if(!strcmp(x,"j"))printf("%u",a);

else if(!strcmp(x,"c"))printf("%c",a);

else if(!strcmp(x,"Pc")||!strcmp(x,"PKc")||\*x=='A')printf("%s",a);

else if(!strcmp(x,"d")||!strcmp(x,"e")||!strcmp(x,"f"))printf(OUTPUT\_PRECISION,a);

else if(!strcmp(x,"x"))printf(INT\_64\_MOD,a);

else if(!strcmp(x,"y"))printf(UNSIGNED\_64\_MOD,a);

else test("Output format error!\n");

}

template<class T1,class T2>

int S(T1&a,T2&b){

return S(a)+S(b)==2;

}

template<class T1,class T2,class T3>

int S(T1&a,T2&b,T3&c){

return S(a)+S(b)+S(c)==3;

}

template<class T1,class T2,class T3,class T4>

int S(T1&a,T2&b,T3&c,T4&d){

return S(a)+S(b)+S(c)+S(d)==4;

}

template<class T1,class T2,class T3,class T4,class T5>

int S(T1&a,T2&b,T3&c,T4&d,T5&e){

return S(a)+S(b)+S(c)+S(d)+S(e)==5;

}

template<class T>

void P(T a){

\_P(a);

putchar(' ');

}

template<class T>

void PN(T a){

\_P(a);

NL

}

template<class T1,class T2>

void PN(T1 a,T2 b){

\_P(a);putchar(' ');

\_P(b);NL

}

template<class T1,class T2,class T3>

void PN(T1 a,T2 b,T3 c){

\_P(a);putchar(' ');

\_P(b);putchar(' ');

\_P(c);NL

}

template<class T1,class T2,class T3,class T4>

void PN(T1 a,T2 b,T3 c,T4 d){

\_P(a);putchar(' ');

\_P(b);putchar(' ');

\_P(c);putchar(' ');

\_P(d);NL

}

template<class T1,class T2,class T3,class T4,class T5>

void PN(T1 a,T2 b,T3 c,T4 d,T5 e){

\_P(a);putchar(' ');

\_P(b);putchar(' ');

\_P(c);putchar(' ');

\_P(d);putchar(' ');

\_P(e);NL

}

int kase;

const double pi=4\*atan(1);

//}

int fan(int x){

int res=0;

while(x){

res\*=10;

res+=x%10;

x/=10;

}

return res;

}

int main(){

SHOW\_TIME

int a,b;

S(a,b);

P(fan(fan(a)+fan(b)));

}

/\*\*\*\*\*\*\*\*\*Risoft corporation all rights reserved\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Template V1.26 build 20141210\*\*\*\*\*\*\*\*\*\*\*\*\*/

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

int inverse(int m);

int inverse\_sum(int a,int b);

int main()

{

int a,b,sum;

scanf("%d %d",&a,&b);

sum=inverse\_sum(a,b);

sum=inverse(sum);

printf("%d\n",sum);

return 0;

}

int inverse(int m)

{

char str[6];

int i,j;

i=0;

while(m)

{

str[i++]=m%10+'0';

m/=10;

}

str[i]='\0';

j=atoi(str);

return (j);

}

int inverse\_sum(int a,int b)

{

int j,k;

j=inverse(a);

k=inverse(b);

return j+k;

}

**算法训练 字符删除**

**问题描述**  
　　编写一个程序，先输入一个字符串str（长度不超过20），再输入单独的一个字符ch，然后程序会把字符串str当中出现的所有的ch字符都删掉，从而得到一个新的字符串str2，然后把这个字符串打印出来。  
　　输入格式：输入有两行，第一行是一个字符串（内部没有空格），第二行是一个字符。  
　　输出格式：经过处理以后的字符串。  
**输入输出样例**

样例输入

123-45-678  
-

样例输出

12345678

本题的C++参考代码如下：

#include<iostream>

#include<string>

using namespace std;

int main(){

string str1,str2;

char ch;

getline(cin,str1);

cin>>ch;

int i;

for(i=0;i<str1.size();i++)

if(str1[i]!=ch)

str2+=str1[i];

cout<<str2<<endl;

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

int main()

{

char str[21],t[21],c;

int i,len,count=0;

gets(str);

scanf("%c",&c);

len=strlen(str);

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

{

if(str[i]!=c)

t[count++]=str[i];

}

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

printf("%c",t[i]);

printf("\n");

return 0;

}

**算法训练 字串统计**

问题描述

　　给定一个长度为n的字符串S，还有一个数字L，统计长度大于等于L的出现次数最多的子串（不同的出现可以相交），如果有多个，输出最长的，如果仍然有多个，输出第一次出现最早的。

输入格式

　　第一行一个数字L。  
　　第二行是字符串S。  
　　L大于0，且不超过S的长度。

输出格式

　　一行，题目要求的字符串。  
  
　　输入样例1：  
　　4  
　　bbaabbaaaaa  
  
　　输出样例1：  
　　bbaa  
  
　　输入样例2：  
　　2  
　　bbaabbaaaaa  
  
　　输出样例2：  
　　aa

数据规模和约定

　　n<=60  
　　S中所有字符都是小写英文字母。

提示

　　枚举所有可能的子串，统计出现次数，找出符合条件的那个

本题的C++参考代码如下：

#include<stdlib.h>

#include<stdio.h>

#include<bitset>

#include<iostream>

#include<string>

using namespace std;

int main()

{

string s;

int l;

cin>>l;

cin>>s;

int map[60][60]={0};

for(int i=0;i<s.length();i++)

{

for(int j=0;j<s.length();j++)

{

if(i==j)

{

map[i][j]=0;

}

else

{

if(s[i]==s[j])

{

if(i>0 && j>0)

map[i][j]=map[i-1][j-1]+1;

else

map[i][j]=1;

}

}

}

}

int m[60][60]={0};//计算每一行某个数值出现的次数只计算大于l的数值

int max=0;

for(int i=0;i<60;i++)

{

for(int j=0;j<60;j++)

{

int val=map[i][j];//取出数值

if(val>=l)

{

m[i][val]++;//累加

if(m[i][val]>max)

{

max=m[i][val];

}

}

}

}

for(int j=59;j>=0;j--)//从后向前找保证字符是最长的

{

for(int i=0;i<60;i++)//从前向后找保证字符是最早出现的

{

if(m[i][j]==max)

{

cout<<s.substr(i-j+1,j)<<endl;

return 0;

}

}

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

char b[60][60];

int c[60];

int main()

{

int l,i1,max=1,now=0,maxn=1,i2,weizhi=0,i3,i4,weizhi1,changdu1=-1;

char a[10000];

scanf("%d",&l);

getchar();

gets(a);

int n=strlen(a);

for(i1=n;i1>=l;i1--)//长度

{

weizhi=0;

for(i3=0;i3<n-i1;i3++)

c[i3]=0;

for(i2=0;i2<n;i2++)//开始位置

{

if(i2+i1>n)

break;

int ok=1;

for(i3=0;i3<weizhi;i3++)//b的位置

{ int ko=1;

for(i4=0;i4<i1;i4++)

{

if(b[i3][i4]!=a[i2+i4])

ko=0;

}

if(ko)

{ ok=0;

c[i3]++;

if(c[i3]>max)

{max=c[i3];

changdu1=i1;

weizhi1=i2;

}

break;

}

}

if(ok)

{

for(i4=0;i4<i1;i4++)

{

b[weizhi][i4]=a[i2+i4];

}

c[weizhi]=1;

weizhi++;

}

}

}

for(i1=weizhi1;i1<weizhi1+changdu1;i1++)

printf("%c",a[i1]);

printf("\n");

return 0;

}

**算法训练 进制转换**

问题描述

　　编写一个程序，输入一个二进制的字符串（长度不超过32），然后计算出相应的十进制整数，并把它打印出来。  
　　输入格式：输入为一个字符串，每个字符都是’0’或’1’，字符串的长度不超过32。  
　　输出格式：输出一个整数。  
　　输入输出样例

样例输入

1101

样例输出

13

本题的C++参考代码如下：

#include <iostream>

#include <string>

#include <bitset>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char\*\* argv) {

bitset<32> bit;

cin>>bit;

cout<<bit.to\_ulong();

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

int main()

{

char a[100]={0};

long long int i,len,sum=0;

scanf("%s",a);

len=strlen(a);

i=1;

while(len>=1)

{

if(a[len-1]=='1')

sum+=i;

i\*=2;

len--;

}

printf("%I64d",sum);

return 0;

}

**算法训练 阶乘**

**问题描述**  
　　一个整数n的阶乘可以写成n!，它表示从1到n这n个整数的乘积。阶乘的增长速度非常快，例如，13!就已经比较大了，已经无法存放在一个整型变量中；而35!就更大了，它已经无法存放在一个浮点型变量中。因此，当n比较大时，去计算n!是非常困难的。幸运的是，在本题中，我们的任务不是去计算n!，而是去计算n!最右边的那个非0的数字是多少。例如，5! = 1\*2\*3\*4\*5 = 120，因此5!最右边的那个非0的数字是2。再如：7! = 5040，因此7!最右边的那个非0的数字是4。请编写一个程序，输入一个整数n(n<=100)，然后输出n! 最右边的那个非0的数字是多少。  
　　输入格式：输入只有一个整数n。  
　　输出格式：输出只有一个整数，即n! 最右边的那个非0的数字。  
**输入输出样例**

样例输入

6

样例输出

2

本题的C++参考代码如下：

#include <iostream>

#include <cmath>

#include <string>

#include <windows.h>

#include <stack>

#include <vector>

using namespace std;

int main(int argc, char\*\* argv) {

vector<int> cs;

int n;

cin >> n;

cs.push\_back(1);

for (int i = 2; i <= n; i++)

{

for (int j = 0; j<cs.size(); j++)

{

cs[j] \*= i;

}

for (int j = 0; j<cs.size() - 1; j++)

{

if (cs[j]>9)

{

cs[j + 1] += cs[j] / 10;

cs[j] = cs[j] % 10;

}

}

while (cs[cs.size() - 1]>9)

{

cs.push\_back(cs[cs.size() - 1] / 10);

cs[cs.size() - 2] = cs[cs.size() - 2] % 10;

}

}

for (int i = 0; i < cs.size() ; i++)

{

if (cs[i]!=0)

{

cout << (char)(cs[i] + '0');

break;

}

}

return 0;

}

本题的C参考代码如下：

//常州轻工职业技术学院七七队长提供

#include<stdio.h>

int main(void)

{

int n;

int i;

int sum=1;

scanf("%d",&n);

for(i=n;i>=1;i--)

{

sum = sum\*i;

while(sum%10==0)

sum=sum/10;

if(sum/100>0)

sum=sum%100;

}

printf("%d",sum%10);

return 0;

}

**算法训练 输出米字形**

根据输入的正整数n (1　　米字形由一个(2n-1)\*(2n-1)的矩阵组成，矩阵包含从大写A开始的n个字母  
　　例如:n=3时，包含A,B,C；n=4时，包含A,B,C,D。  
　　矩阵的正中间为n个字母中字典序最大的那个，从这个字母开始，沿着西北、正北、东北、正西、正东、西南、正南、东南八个方向各有一条由大写字母组成的直线。并且直线上的字母按字典序依次减小，直到大写字母A。  
　　矩阵的其它位置用英文句号．填充。  
  
　　样例输入一  
　　3  
  
　　样例输出一  
　　Ａ．Ａ．Ａ  
　　．ＢＢＢ．  
　　ＡＢＣＢＡ  
　　．ＢＢＢ．  
　　Ａ．Ａ．Ａ  
  
　　样例输入二  
　　4  
  
　　样例输出二  
　　Ａ．．Ａ．．Ａ  
　　．Ｂ．Ｂ．Ｂ．  
　　．．ＣＣＣ．．  
　　ＡＢＣＤＣＢＡ  
　　．．ＣＣＣ．．  
　　．Ｂ．Ｂ．Ｂ．  
　　Ａ．．Ａ．．Ａ

本题的C++参考代码如下：

#include "iostream"

using namespace std;

const int MAXN = 10000;

char g[MAXN][MAXN];

void init(int n)

{

for (int i = 0; i < 2\*n-1; ++i)

{

for (int j = 0; j < 2\*n-1; ++j)

{

g[i][j] = '.';

}

}

}

void fillg(int n)

{

int m = (2\*n-1)/2;

for (int i = 0; i <= m; ++i)

{

g[i][i] = 'A'+i;

g[m][i] = 'A'+i;

g[i][m] = 'A'+i;

}

for (int i = 0; i <= m; ++i)

{

for (int j = 2\*n-2; j > m; --j)

{

g[i][j] = g[i][2\*n-2-j];

}

}

for (int i = 2\*n-2; i > m; --i)

{

for (int j = 0; j<2\*n-1; ++j)

{

g[i][j] = g[2\*n-2-i][j];

}

}

}

int main()

{

int n;

cin>>n;

init(n);

fillg(n);

for (int i = 0; i < 2\*n-1; ++i)

{

for (int j = 0; j < 2\*n-1; ++j)

{

cout<<g[i][j];

}

cout<<endl;

}

cout<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int n,i,j,k,t;

scanf("%d",&n);

char a[2\*n-1][2\*n-1];

for(i=0;i<2\*n-1;i++)

for(j=0;j<2\*n-1;j++)

a[i][j]=0;

k=n;

for(i=0;i<n-1;i++) //行数

{

k--; //k为字符间隔

t=i;

for(j=1;j<=3;j++)

{

a[i][t]='A'+i;

t+=k;

}

}

for(i=0;i<n;i++) //中间一行

{

a[n-1][i]='A'+i;

a[n-1][2\*n-2-i]='A'+i;

}

for(i=0;i<=n-2;i++) //填补句号

for(j=0;j<2\*n-1;j++)

{

if(a[i][j]==0)

a[i][j]='.';

}

for(i=0;i<=n-2;i++) //翻转

for(j=0;j<2\*n-1;j++)

a[2\*n-2-i][j]=a[i][j];

for(i=0;i<2\*n-1;i++)

{

for(j=0;j<2\*n-1;j++)

printf("%c",a[i][j]);

printf("\n");

}

return 0;

}

**算法训练 整数平均值**

编写函数，求包含n个元素的整数数组中元素的平均值。要求在函数内部使用指针操纵数组元素，其中n个整数从键盘输入，输出为其平均值。

样例输入: （输入格式说明：5为输入数据的个数，3 4 0 0 2 是以空格隔开的5个整数）  
5   
3 4 0 0 2  
样例输出:  
1

样例输入:   
7  
3 2 7 5 2 9 1  
样例输出:  
4

本题的C++参考代码如下：

//整体平均数

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*a,n,i,sum=0;

scanf("%d",&n);

a=(int \*)malloc(n\*sizeof(int));

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

{

scanf("%d",&a[i]);

sum+=a[i];

}

printf("%d\n",sum/n);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int n,i,j,sum=0;

scanf("%d",&n);

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

{

scanf("%d",&j);

sum+=j;

}

printf("%d",sum/n);

return 0;

}

**算法训练 确定元音字母位置**

输入一个字符串，编写程序输出该字符串中元音字母的首次出现位置，如果没有元音字母输出0。英语元音字母只有‘a’、‘e’、‘i’、‘o’、‘u’五个。

样例输入:   
hello  
样例输出:  
2

样例输入:   
apple  
样例输出:  
1

样例输入:   
pmp  
样例输出:  
0

本题的C++参考代码如下：

#include<iostream>

using namespace std;

int main()

{

char a[100];

cin>>a;

for(int i=0;i<100;i++)

{

if(a[i]=='a'||a[i]=='e'||a[i]=='i'||a[i]=='o'||a[i]=='u')

{

cout<<i+1;

break;

}

if(i==99)

{

cout<<0;

break;

}

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

char s[1000]={0};

int isyuanyin(char a)

{

switch(a)

{

case 'a':return 1;

case 'e':return 1;

case 'i':return 1;

case 'o':return 1;

case 'u':return 1;

}

return 0;

}

int main()

{

int l,i,num=0;

scanf("%s",s);

l=strlen(s);

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

if(isyuanyin(s[i]))

{

num=i+1;break;

}

printf("%d",num);

return 0;

}

**算法训练 斜率计算**

　输入两个点的坐标，即p1 = (x1, y1)和p2=(x2, y2)，求过这两个点的直线的斜率。如果斜率为无穷大输出“INF”。

样例输入

1 2  
2 4

样例输出

2

样例输入

1 2  
1 4

样例输出

INF

样例输入

1 2  
3 2

样例输出

0

本题的C++参考代码如下：

#include<iostream>

using namespace std;

int main()

{

double x1,x2,y1,y2,k;

cin>>x1>>y1>>x2>>y2;

if(x1!=x2)

{

k=(y2-y1)/(x2-x1);

cout<<k;

}

else

cout<<"INF";

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main(){

int a,b,c,d;

scanf("%d %d",&a,&b);

scanf("%d %d",&c,&d);

if(a==c)

printf("INF\n");

else

printf("%d\n",(b-d)/(a-c));

return 0;

}

**算法训练 十进制数转八进制数**

　编写函数把一个十进制数输出其对应的八进制数。

样例输入

9274

样例输出

22072

本题的C++参考代码如下：

#include<iostream>

#include<stdio.h>

using namespace std;

int main()

{

int sum,k=0,b[1000],j;

cin>>sum;

while(sum>=8)

{

b[k]=(int )sum/8;

b[k]=(int )sum-b[k]\*8;

k++;

sum=sum/8;

}

b[k]=(int )sum;

j=k;

while(j>=0)

{

printf("%d",b[j]);

j--;

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

int r;

scanf("%d",&r);

printf("%o",r);

return 0;

}

**算法训练 筛选号码**

问题描述

　　有n个人围成一圈，顺序排号（编号为1到n）。从第1个人开始报数(从1到3报数)，凡报到3的人退出圈子。从下一个人开始继续报数，直到剩下最后一个人，游戏结束。  
　　问最后留下的是原来第几号的那位。  
　　举个例子，8个人围成一圈：  
　　1 2 3 4 5 6 7 8  
　　第1次报数之后，3退出，剩下：  
　　1 2 4 5 6 7 8 （现在从4开始报数）  
　　第2次报数之后，6退出，剩下：  
　　1 2 4 5 7 8 （现在从7开始报数）  
　　第3次报数之后，1退出，剩下：  
　　2 4 5 7 8 （现在从2开始报数）  
　　第4次报数之后，5退出，剩下：  
　　2 4 7 8 （现在从7开始报数）  
　　第5次报数之后，2退出，剩下：  
　　4 7 8 （现在从4开始报数）  
　　第6次报数之后，8退出，剩下：  
　　4 7 （现在从4开始报数）  
　　最后一次报数之后，4退出，剩下：  
　　7.  
　　所以，最后留下来的人编号是7。

输入格式

　　一个正整数n，(1<n<10000)

输出格式

　　一个正整数，最后留下来的那个人的编号。

样例输入

8

样例输出

7

数据规模和约定

　　对于100%的数据，1<n<10000。

本题的C++参考代码如下：

#include<stdio.h>

main()

{ int i,N,x,m,s;

x=3;

s=0;

scanf("%d",&m);

for(i=2;i<=m;i++)

s=(s+x)%i;

printf("%d\n",s+1);

}

本题的C参考代码如下：

#include <stdio.h>

#include <malloc.h>

typedef struct form{

int num;

int date;

struct form \*link;}

del;

del \*creat(int n){

int i;

del \*head,\*p1,\*p2;

head=(del \*)malloc(sizeof(del));

p1=(del \*)malloc(sizeof(del));

head->link=p1;

for(i=1;i<=n-2;i++){p2=(del \*)malloc(sizeof(del));

p1->link=p2;

p1=p2;}

p1->link=head;

return(head);

}

void dateop(del \*h,int n){

del \*p;

int i,j=1;

p=h;

for(i=1;i<=n;i++){

p->num=i;

p->date=j;j++;

if(j==4) j=1;

p=p->link;}

}

int deal(del \*h,int n){

del \*k;

int s;

int count,j=1,i;

count=n;

k=h;

while(count!=1){

if(j==3&&k->date!=0) {k->date=0;count--;}

k=k->link;

j++;

if(k->date==0) j--;

if(j==4) j=1;

}

k=h;

for(i=1;i<=n;i++){

if(k->date!=0) {s=k->num;break;}

k=k->link;}

return(s);}

int main(){

int x;

int i;

del \*p;

scanf("%d",&x);

p=creat(x);

dateop(p,x);

x=deal(p,x);

for(i=1;i<x;i++){

p=p->link;}

printf("%d",p->num);

return 0;

}

**算法训练 连接字符串**

　编程将两个字符串连接起来。例如country与side相连接成为countryside。  
　　输入两行，每行一个字符串（只包含小写字母，长度不超过100）；输出一行一个字符串。

样例输入

country  
side

样例输出

countryside

本题的C++参考代码如下：

#include <iostream>

#include <string>

#include <map>

#include <vector>

#include <algorithm>

#include <limits.h>

using namespace std;

//int a[2000001];

//int b[100010];

//int m=50000;

int main(){

string s1,s2;

cin>>s1>>s2;

cout<<s1+s2<<endl;

return 0;

}

本题的C参考代码如下：

//连接字符串

#include<stdio.h>

#include<string.h>

int main()

{

char s1[202],s2[202];

scanf("%s",s1);

scanf("%s",s2);

strcat(s1,s2);

printf("%s",s1);

printf("\n");

return 0;

}

**算法训练 统计字符次数**

　输入一个字符串(长度在100以内)，统计其中数字字符出现的次数。

样例输入

Ab100cd200

样例输出

6

本题的C++参考代码如下：

#include <iostream>

#include <string>

using namespace std;

int main(){

int n=0;

string s;

cin>>s;

for(int i=0;i<s.size();i++){

if(s[i]>='0' && s[i]<='9')

n++;

}

cout<<n<<endl;

return 0;}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

char a[101];

int i=0,s=0;

gets(a);

for(i=0;\*(a+i)!='\0';i++)

{

if(\*(a+i)>=48&&\*(a+i)<=57)

s++;

}

printf("%d",s);

return 0;

}

**算法训练 成绩的等级输出**

　　输入一个百分制的成绩t后，按下式输出它的等级。等级为：90~100为A，80~89为B，70~79为C，60~69为D，0~59为E。

样例输入

98

样例输出

A

本题的C++参考代码如下：

#include<iostream>

using namespace std;

int main()

{

int x;

cin>>x;

if(x>=90&&x<=100)

cout<<"A";

if(x>=80&&x<=89)

cout<<"B";

if(x>=70&&x<=79)

cout<<"C";

if(x>=60&&x<=69)

cout<<"D";

if(x>=0&&x<=59)

cout<<"E";

return 0;

}

本题的C参考代码如下：

//成绩的等级输出

#include<stdio.h>

int main()

{

int t;

scanf("%d",&t);

if(t>=90 && t<=100)

printf("A\n");

if(t>=80 && t<=89)

printf("B\n");

if(t>=70 && t<=79)

printf("C\n");

if(t>=60 && t<=69)

printf("D\n");

if(t<60)

printf("E\n");

return 0;

}

**算法训练 比较字符串**

　编程实现两个字符串s1和s2的字典序比较。（保证每一个字符串不是另一个的前缀，且长度在100以内）。若s1和s2相等，输出0；若它们不相等，则指出其第一个不同字符的ASCII码的差值：如果s1>s2，则差值为正；如果s1<s2，则差值为负。

样例输入

java basic

样例输出

8

本题的C++参考代码如下：

#include <stdio.h>

#include <string.h>

#include <iostream>

#include<functional>

#include <queue>

#include <string>

#include <algorithm>

using namespace std;

const int maxn = 135;

int n;

char str1[maxn],str2[maxn];

int main()

{

#ifndef ONLINE\_JUDGE

freopen("data.txt","r",stdin);

#endif

scanf("%s%s",&str1,&str2);

int len1 = strlen(str1);

int len2 = strlen(str2);

int len = len1<len2?len1:len2;

if( len1 == len2 ){

puts("0");

}

else

for( int i = 0; i <= len; i ++ ){

if( str1[i] != str2[i] ){

printf("%d\n",str1[i]-str2[i]);

break;

}

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

int min(int a,int b)

{

return a>b?b:a;

}

int main()

{

char a[1000];

char b[1000];

scanf("%s",a);

scanf("%s",b);

int k=min(strlen(a),strlen(b)),i1;

int ok=1;

for(i1=0;i1<k;i1++)

{ if(a[i1]!=b[i1])

{ printf("%d\n",a[i1]-b[i1]);

ok=0;

break;

}

}

if(ok==1)

if(strlen(a)==strlen(b))

printf("0\n");

else if(strlen(a)>strlen(b))

printf("%d\n",a[i1]);

else

printf("%d\n",b[i1]);

return 0;

}

**算法训练 最长字符串**

　　求出5个字符串中最长的字符串。每个字符串长度在100以内，且全为小写字母。

样例输入

one two three four five

样例输出

three

本题的C++参考代码如下：

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

int max = 0,k;

string s[5];

for (int i = 0; i < 5; ++i)

{

cin >> s[i];

if(s[i].size()>max)

{

max=s[i].size();

k=i;

}

}

cout<<s[k]<<endl;

return 0;

}

本题的C参考代码如下：

//最长字符串

#include<stdio.h>

#include<string.h>

int main()

{

char str[5][100];

int len[5];

int i,j,t,max;

max=0;

for(i=0;i<5;i++)

{

scanf("%s",str[i]);

len[i]=strlen(str[i]);

if(len[i]>max)

{

max=len[i];

t=i;

}

}

puts(str[t]);

return 0;

}

**算法训练 字符串逆序  、**

　　输入一个字符串，长度在100以内，按相反次序输出其中的所有字符。

样例输入

tsinghua

样例输出

auhgnist

本题的C++参考代码如下：

#include <iostream>

#include<functional>

#include <queue>

#include <string>

#include <algorithm>

using namespace std;

const int maxn = 40005;

int main()

{

#ifndef ONLINE\_JUDGE

//freopen("data.txt","r",stdin);

#endif

char str[maxn];

gets(str);

int len = strlen(str);

for( int i = len-1; i >= 0; i -- )

printf("%c",str[i]);

puts("");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

char c1[100]={0},c2[100]={0},\*p1,\*p2; //初始化之后，即使后面不改，也不会再出现乱码了

int i=0,j=0,m;

gets(c1);

p1=c1;

p2=c2;

while (\*p1++) i++;

for (;;)

{

// 方法1

p1=c1; // 指针归位，因为现在的指针已经不指向数组首地址的位置了

p2=c2;

p2[j]=p1[i-1];

// 方法2

//c2[j] = c1[i-1]; // 比较简单的方式还是直接用数据下标

j++;

i--;

if (i==0) break;

}

p2[j]='\0';

puts(p2);

return 0;

**算法训练 判定数字**

　编写函数，判断某个给定字符是否为数字。

样例输入

9

样例输出

yes

本题的C++参考代码如下：

#include <iostream>

using namespace std;

string function(char c)

{

if(c>='0' && c<='9')

return "yes";

else return "no";

}

int main()

{

char c;

cin>>c;

cout<<function(c);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char \*argv[]) {

char x;

scanf("%c",&x);

if('0'<=x&&x<='9')

{

printf("yes");

}

else

{

printf("no");

}

return 0;

}

**算法训练 最大值与最小值的计算**

　　输入11个整数，计算它们的最大值和最小值。

样例输入

0 1 2 3 4 5 6 7 8 9 10

样例输出

10 0

本题的C++参考代码如下：

#include <string>

#include <algorithm>

#include <limits.h>

using namespace std;

int main(){

int min=INT\_MAX,max=INT\_MIN;

int t;

for(int i=0;i<11;i++){

cin>>t;

if(t > max)

max=t;

if(t < min)

min=t;

}

cout<<max<<" "<<min<<endl;

return 0;}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

int n;

int a[11],i1,max=-234234,min=93759347;

for(i1=0;i1<11;i1++)

{

scanf("%d",&a[i1]);

if(max<a[i1])

max=a[i1];

if(min>a[i1])

min=a[i1];

}

printf("%d %d\n",max,min);

return 0;

}

**算法训练 字符串编辑**

问题描述

　　从键盘输入一个字符串（长度<=40个字符），并以字符 ’.’ 结束。编辑功能有：  
　　1 D：删除一个字符，命令的方式为： D a 其中a为被删除的字符，例如：D s 表示删除字符 ’s’ ，若字符串中有多个 ‘s’，则删除第一次出现的。  
　　2 I：插入一个字符，命令的格式为：I a1 a2 其中a1表示插入到指定字符前面，a2表示将要插入的字符。例如：I s d 表示在指定字符 ’s’ 的前面插入字符 ‘d’ ，若原串中有多个 ‘s’ ，则插入在最后一个字符的前面。  
　　3 R：替换一个字符，命令格式为：R a1 a2 其中a1为被替换的字符，a2为替换的字符，若在原串中有多个a1则应全部替换。  
　　在编辑过程中，若出现被改的字符不存在时，则给出提示信息。

输入格式

　　输入共两行，第一行为原串(以’.’结束)，第二行为命令（输入方式参见“问题描述” 。

输出格式

　　输出共一行，为修改后的字符串或输出指定字符不存在的提示信息。

样例输入

This is a book.  
D s

样例输出

Thi is a book.

输入输出样例解释

　　命令为删去s，第一个在字符中出现的s在This中，即得到结果。

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

using namespace std;

class myString :public string

{

public:

myString(char \*str=NULL):string(str){}

string deleteChar(char ch)

{

int index=this->find\_first\_of( ch );

if(index != -1)

{

return this->substr(0,index)+this->substr(index+1);

}

else return "指定字符不存在";

return \*this;

}

string insertChar( char ch1, char ch2)

{

int index = this -> find\_last\_of(ch1);

if(index != -1)

{

return this->substr(0,index)+ch2+this->substr(index);

}

else return "指定字符不存在";

return \*this;

}

string replaceChar(char ch1, char ch2)

{

bool isExit=false;

for(int i=0;i<(this->size()); i++)

{

if((\*this)[i]==ch1)

{

isExit=true;

(\*this)[i]=ch2;

}

}

if(isExit == false)

return "指定字符不存在";

return \*this;

}

};

int main()

{

char s[1000];

cin.getline(s,1000);

myString str(s);

char ch;

cin>>ch;

if(ch=='D')

{

cin>>ch;

cout<<str.deleteChar(ch);

}

else if(ch=='I')

{

char ch1,ch2;

cin>>ch1>>ch2;

cout<<str.insertChar(ch1,ch2);

}

else

{

char ch1,ch2;

cin>>ch1>>ch2;

cout<<str.replaceChar(ch1,ch2);

}

return 0;

}

本题的C参考代码如下：

//字符串编辑

#include<stdio.h>

#include<string.h>

int i,j;

//删除操作

void Del(char S[],char a,int len)

{

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

if(a==S[i])

break;

if(i!=len)

{

for(j=i;j<len;j++)

S[j]=S[j+1];

}

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

if(S[i]=='#')

printf(" ");

else

printf("%c",S[i]);

}

//插入操作

void Ins(char S[],char a1,char a2,int len)

{

i=len-1;

for(;i>=0;i--)

if(S[i]==a2)

break;

if(i!=0)

{

len++;

//字符右移

for(j=len-1;j>i;j--)

S[j]=S[j-1];

S[i]=a1; //最后出现的a2替换成a1

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

if(S[i]=='#')

printf(" ");

else

printf("%c",S[i]);

//printf(".");

}

}

//替换操作

void Res(char S[],char a1,char a2,int len)

{

int flag=0;

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

if(S[i]==a1)

{

S[i]=a2;

flag=1;

}

if(flag==1)

{

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

if(S[i]=='#')

printf(" ");

else

printf("%c",S[i]);

}

else

printf("指定字符不存在\n");

}

int main()

{

char S[41],Edit,a,a1,a2;

int len=0;

gets(S);

while(S[i]!='\0')

{

if(S[i]==' ')

S[i]='#';

len++;

i++;

}

//printf("len=%d\n");

scanf("%c",&Edit);

getchar();

switch(Edit)

{

case 'd':

case 'D':

{

scanf("%c",&a);

Del(S,a,len);

}break;

case 'i':

case 'I':

{

scanf("%c %c",&a1,&a2);

Ins(S,a2,a1,len);

} break;

case 'r':

case 'R':

{

scanf("%c %c",&a1,&a2);

Res(S,a1,a2,len);

}break;

}

return 0;

}

**算法训练 比赛安排**

问题描述

　　设有有2 n（n<=6）个球队进行单循环比赛，计划在2 n – 1天内完成，每个队每天进行一场比赛。设计一个比赛的安排，使在2 n – 1天内每个队都与不同的对手比赛。

输入格式

　　输入文件matchplan.in共一行，输入n的数值。

输出格式

　　输出文件matchplan.out共（2 n – 1）行，第i行输出第i天的比赛安排。  
　　格式为：<i> A-B，C-D，……。其中i是天数，A，B分别为比赛双方的编号，每行共2 n-1个比赛场次。

样例输入

2

样例输出

<1>1-2,3-4  
<2>1-3,2-4  
<3>1-4,2-3

本题的C++参考代码如下：#include "iostream"

#include "string"

#include "stdio.h"

using namespace std;

const int N =1000;

bool bArrange[N][N];

int main()

{

int n;

cin>>n;

int row=(1<<n);

int volumn=1<<(n-1);

for(int i = 0; i< row-1 ; i++ )

{

cout<<"<"<<i+1<<">"<<"1-"<<i+2;

bool isArrage[N]={false};

isArrage[1]=isArrage[i+2]=true;

bArrange[1][i+2]=bArrange[i+2][1]=true;

for(int j =1 ; j< volumn ; j++)

{

int a;

for( a=2;a<=row;a++)

if(isArrage[a]==false)

break;

for(int b=a+1;b<=row;b++)

{

if(bArrange[a][b]==false&&isArrage[b]==false)

{

bArrange[a][b]=bArrange[b][a]=true;

isArrage[a]=isArrage[b]=true;

cout<<" "<<a<<"-"<<b;

break;

}

}

}

cout<<endl;

}

return 0;

}

本题的C参考代码如下：

//求最小公倍数

#include<stdio.h>

void lcm(long long a,long long b)

{

int t;

if(a<b)

{

t=a;

a=b;

b=t;

}

long long m,n,r;

m=a,n=b;

r=a%b;

while(r!=0)

{

a=b;

b=r;

r=a%b;

}

printf("%I64d\n",(m\*n)/b);

}

int main()

{

long long a,b;

scanf("%I64d %I64d",&a,&b);

if(a==0 || b==0)

printf("0\n");

else

lcm(a,b);

return 0;

}

**算法训练 大小写判断**

问题描述

　　给定一个英文字母判断这个字母是大写还是小写。

输入格式

　　输入只包含一个英文字母c。

输出格式

　　如果c是大写字母，输出“upper”，否则输出“lower”。

样例输入

x

样例输出

lower

样例输入

B

样例输出

upper

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

#include "ctype.h"

using namespace std;

const int N =1000;

bool bArrange[N][N];

int main()

{

char ch;

cin>>ch;

if(islower(ch))

cout<<"lower";

else cout<<"upper";

return 0;

}本题的C参考代码如下：

#include <stdio.h>

int main()

{

char ch=0;

scanf("%c",&ch);

if(ch>='A' && ch<='Z')

{

printf("upper");

}

else

{

printf("lower");

}

return 0;

}

**算法训练 乘法表**

问题描述

　　输出九九乘法表。

输出格式

　　输出格式见下面的样例。乘号用“\*”表示。

样例输出

下面给出输出的前几行：  
1\*1=1  
2\*1=2 2\*2=4  
3\*1=3 3\*2=6 3\*3=9  
4\*1=4 4\*2=8 4\*3=12 4\*4=16  
……

本题的C++参考代码如下：

#include<iostream>

using namespace std;

int main()

{

for(int i=1;i<10;i++)

{

for(int j=1;j<=i;j++)

{

cout<<i<<"\*"<<j<<"="<<i\*j<<"\t";

}

cout<<"\n";

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

int i,j;

for(i=1;i<10;i++)

{

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

printf("%d\*%d=%d ",i,j,i\*j);

printf("\n");

}

return 0;

}

**算法训练 平方计算**

问题描述

　　输入正整数a, m，输出a^2%m，其中^表示乘方，即a^2表示a的平方，%表示取余。

输入格式

　　输入包含两个整数a, m，a不超过10000。

输出格式

　　输出一个整数，即a^2%m的值。

样例输入

5 6

样例输出

1

本题的C++参考代码如下：

#include <iostream>

using namespace std;

// 求数组的和，其中p为指针第一个元素的指针，n为数组中的元素个数。

int Sum(int \*p, int n)

{

int sum=0;

for(int i=0;i<n;i++)

sum+=p[i];

return sum;

}

int s[100];

int n;

int main()

{

cin >> n;

for (int i = 0; i < n; ++i)

cin >> s[i];

cout << Sum(s, n);

return 0;

}

本题的C参考代码如下：

//平方计算

#include<stdio.h>

int main()

{

int a,m;

scanf("%d %d",&a,&m);

printf("%d\n",(a\*a)%m);

return 0;

}

**算法训练 奇偶判断**

问题描述

　　能被2整除的数称为偶数，不能被2整除的数称为奇数。给一个整数x，判断x是奇数还是偶数。

输入格式

　　输入包括一个整数x，0<=x<=100000000。

输出格式

　　如果x是奇数，则输出“odd”，如果是偶数，则输出“even”。

样例输入

10

样例输出

even

样例输入

2009

样例输出

odd

本题的C++参考代码如下：

#include<iostream>

using namespace std;

int main()

{

int x;

cin>>x;

if(x%2==0) cout<<"even";

else cout<<"odd";

return 0;

}

#include <stdio.h>

#include <stdlib.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char \*argv[]) {

int a[100];

int n;

scanf("%d",&n);

if(n%2==0)

printf("even");

else

printf("odd");

return 0;

}

**算法训练 矩阵乘方**

问题描述

　　给定一个矩阵A,一个非负整数b和一个正整数m，求A的b次方除m的余数。  
　　其中一个nxn的矩阵除m的余数得到的仍是一个nxn的矩阵，这个矩阵的每一个元素是原矩阵对应位置上的数除m的余数。  
　　要计算这个问题，可以将A连乘b次，每次都对m求余，但这种方法特别慢，当b较大时无法使用。下面给出一种较快的算法(用A^b表示A的b次方)：  
　　若b=0，则A^b%m=I%m。其中I表示单位矩阵。  
　　若b为偶数，则A^b%m=(A^(b/2)%m)^2%m，即先把A乘b/2次方对m求余，然后再平方后对m求余。  
　　若b为奇数，则A^b%m=(A^(b-1)%m)\*a%m，即先求A乘b-1次方对m求余，然后再乘A后对m求余。  
　　这种方法速度较快，请使用这种方法计算A^b%m，其中A是一个2x2的矩阵，m不大于10000。

输入格式

　　输入第一行包含两个整数b, m，第二行和第三行每行两个整数，为矩阵A。

输出格式

　　输出两行，每行两个整数，表示A^b%m的值。

样例输入

2 2  
1 1  
0 1

样例输出

1 0  
0 1

本题的C++参考代码如下：

#include <iostream>

using namespace std;

// 坐标或向量的结构

struct pointer {

int x;

int y;

};

// 将A坐标按B向量平移，结果保存在C坐标中。

void move(pointer \*C, pointer \*A, pointer \*B)

{

C->x=A->x+B->x;

C->y=A->y+B->y;

}

int main()

{

pointer A, B, C;

cin >> A.x >> A.y;

cin >> B.x >> B.y;

move(&C, &A, &B);

cout << C.x << " " << C.y;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int b, m;

int a[2][2], ans[2][2], temp[2][2] = {1,1,1,1};

void play()

{

int cnt, cnt2;

for(cnt = 0; cnt < 2; ++cnt)

{

for(cnt2 = 0; cnt2 < 2; ++cnt2)

{

printf("%d ", ans[cnt][cnt2]);

}

printf("\n");

}

}

void cp(int arr1[][2], int arr2[][2])

{

int cnt, cnt2;

for(cnt = 0; cnt < 2 ;++cnt)

for(cnt2 = 0; cnt2 < 2; ++cnt2)

arr1[cnt][cnt2] = arr2[cnt][cnt2];

}

void mod(int arr[][2])

{

int cnt, cnt2;

for(cnt = 0; cnt < 2; ++cnt)

for(cnt2 = 0; cnt2 < 2; ++cnt2)

arr[cnt][cnt2] %= m;

}

void fun2(int a[][2], int b[][2])

{

int cnt, cnt2;

temp[0][0] = a[0][0]\*b[0][0]+a[0][1]\*b[1][0];

temp[0][1] = a[0][0]\*b[0][1]+a[0][1]\*b[1][1];

temp[1][0] = a[1][0]\*b[0][0]+a[1][1]\*b[1][0];

temp[1][1] = a[1][0]\*b[0][1]+a[1][1]\*b[1][1];

}

void fun(int arr[][2], int k)

{

int cnt;

if(k == 0)

{

mod(temp);

cp(ans, temp);

return;

}

if(k == 1)

{

mod(ans);

return;

}

if(k == 2)

{

fun2(a, a);

cp(ans, temp);

// printf("2\n");

// play();

mod(ans);

return;

}

if(k%2 == 0)

{

fun(arr, k/2);

fun2(ans, ans);

cp(ans, temp);

//printf("=0\n");

//play();

mod(ans);

return;

}

if(k%2 != 0)

{

fun(arr, k-1);

fun2(ans, arr);

cp(ans, temp);

//printf("!=0\n");

// play();

mod(ans);

return;

}

}

int main()

{

int cnt, cnt2;

scanf("%d%d", &b, &m);

for(cnt = 0; cnt < 2; ++cnt)

for(cnt2 = 0; cnt2 < 2; ++cnt2)

{

scanf("%d", &a[cnt][cnt2]);

ans[cnt][cnt2] = a[cnt][cnt2];

}

fun(a, b);

play();

return 0;

}

**算法训练 快速排序**

问题描述

　　快速排序是最经常使用的一种排序方式，对于给定的n个数组成的一个数组，请使用快速排序对其进行排序。  
　　现给定一序列，请用快速排序将其按升序排序并输出。

输入格式

　　第一行一个数N。  
　　第2~N+1行每行一个数，表示给定序列。

输出格式

　　共N行，每行一个数，表示所求序列。

样例输入

5  
1  
4  
2  
3  
4

样例输出

1  
2  
3  
4  
4

数据规模和约定

　　共10组数据。  
　　对100%的数据，N<=10^5，所有数均为非负数且在int范围内。

本题的C++参考代码如下：

#include <iostream>

using namespace std;

// 交换两个整数

void Swap(int \*a, int \*b)

{

int temp=\*a;

\*a=\*b;

\*b=temp;

}

int main()

{

int a, b;

cin >> a >> b;

Swap(&a, &b);

cout << a << " " << b;

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。

**算法训练 字串逆序**

问题描述

　　给定一个字符串，将这个串的所有字母逆序后输出。

输入格式

　　输入包含一个字符串，长度不超过100，字符串中不含空格。

输出格式

　　输出包含一个字符串，为上面字符串的逆序。

样例输入

tsinsen

样例输出

nesnist

本题的C++参考代码如下：

#include <iostream>

using namespace std;

int sum(int x, int y)

{

return x+y;

}

int main()

{

int a, b, c;

cin >> a >> b;

c = sum(a, b);

cout << c << endl;

return 0;

}

本题的C参考代码如下：

//字串逆序

#include<stdio.h>

#include<string.h>

int main()

{

char str[100];

int len;

scanf("%s",str);

len=strlen(str)-1;

for(;len>=0;len--)

printf("%c",str[len]);

return 0;

}

**算法训练 删除多余括号**

问题描述

　　从键盘输入一个含有括号的四则运算表达式，要求去掉可能含有的多余的括号，结果要保持原表达式中变量和运算符的相对位置不变，且与原表达式等价,不要求化简。另外不考虑'+'　　'-'用作正负号的情况，即输入表达式不会出现(+a)或(-a)的情形。

输入格式

　　表达式字符串，长度不超过255,　　并且不含空格字符。表达式中的所有变量都是单个小写的英文字母, 运算符只有加+减-乘\*除/等运算符号。

输出格式

　　去掉多余括号后的表达式

样例输入

样例一：

a+(b+c)-d

样例二：

a+b/(c+d)

样例三：

(a\*b)+c/d

样例四：

((a+b)\*f)-(i/j)

样例输出

样例一：

a+b+c-d

样例二：

a+b/(c+d)

样例三：

a\*b+c/d

样例四：

(a+b)\*f-i/j

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

#include "ctype.h"

#include "algorithm"

#include "stack"

using namespace std;

int cacluExprePriority(string str,bool &hasC)

{

int left=0;

int right=0;

bool bfind=false;

for(int i=0;i<str.size();i++)

{

if(str[i]=='(')

left++;

if(str[i]==')')

right++;

if(str[i]=='/')

hasC=true;

if(str[i]=='\*'||str[i]=='/')

{

if(left==right)

return 2;

}

if(str[i]=='+'||str[i]=='-')

{

bfind=true;

}

}

return bfind?1:-1;

}

bool vis[1000];

int safe(int i,int n)

{

if(i<0)

return 0;

if(i>=n)

return n-1;

return i;

}

void findIndexOfBrackets(string str)

{

stack<int>q;

for(int i=0;i<str.size();i++)

{

if(str[i]=='(')

{

q.push(i);

}

if(str[i]==')')

{

int s=q.top();

int t=i;

bool hasC=false;

int priority=cacluExprePriority(str.substr(s+1,t-s-1),hasC);

q.pop();

bool temp=false;

if(s-1>=0&&cacluExprePriority(str.substr(safe(s-1,str.size()),1),temp)>=cacluExprePriority(str.substr(safe(t+1,str.size()),1),temp))

{

char op=str[s-1];

{

if(op=='+')

{

vis[s]=vis[t]=true;

}

if(op=='-')

{

if(priority==2)

vis[s]=vis[t]=true;

}

if(op=='\*')

{

if(priority==2&&hasC==false)

vis[s]=vis[t]=true;

}

}

}

else if(t+1<str.size())

{

char op=str[t+1];

if(op=='+'||op=='-')

{

vis[s]=vis[t]=true;

}

if(op=='\*'||op=='/')

{

if(priority==2)

vis[s]=vis[t]=true;

}

}

}

}

}

int main()

{

string exper;

cin>>exper;

findIndexOfBrackets(exper);

for(int i=0;i<exper.size();i++)

if(vis[i]==false)

cout<<exper[i];

cout<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int q(char \*ch)

{

int i=0,z=0;

ch[i] = '#';

while (ch[i] != ')'||z!=0)

{

if (ch[i]=='(')

{

z++;

}

if (ch[i]==')')

{

z--;

}

i++;

}

ch[i] = '#';

return i;

}

int f(char \*ch, char a)

{

int i = 0,jj=0;

if (a == '+')

{

while (ch[i] != ')')

{

if (ch[i+1] == '(')

if (f(&ch[i], ch[i - 1]) == 0)

i += q(&ch[i]);

else

{

while (ch[i] != ')')

{

i++;

}

}

i++;

}

if (ch[i + 1] == '\*' || ch[i + 1] == '/')

{

return 1;

}

else

{

return 0;

}

}

if (a == '-')

{

while (ch[i] != ')')

{

if (ch[i] == '(')

if (f(&ch[i+1], ch[i - 1]) == 0)

i += q(&ch[i]);

else

{

while (ch[i] != ')')

{

i++;

}

}

if (ch[i] == '+' || ch[i] == '-')

return 1;

i++;

}

if (ch[i + 1] == '\*' || ch[i + 1] == '/')

{

return 1;

}

else

{

return 0;

}

}

if (a == '\*')

{

while (ch[i] != ')')

{

if (ch[i] == '(')

if (f(&ch[i+1], ch[i - 1]) == 0)

q(&ch[i]);

else

{

while (ch[i] != ')')

{

i++;

}

}

if (ch[i] == '+' || ch[i] == '-')

return 1;

i++;

}

return 0;

}

if (a == '/')

{

while (ch[i] != ')')

{

if (ch[i] == '(')

if (f(&ch[i+1], ch[i - 1]) == 0)

q(&ch[i]);

else

{

while (ch[i] != ')')

{

i++;

}

}

if (ch[i] == '+' || ch[i] == '-' || ch[i] == '/' || ch[i] == '\*')

return 1;

i++;

}

}

while (ch[i] != ')')

{

if (ch[i] == '(')

if (f(&ch[i+1], ch[i - 1]) == 0)

i += q(&ch[i]);

else

{

while (ch[i] != ')')

{

i++;

}

}

if (ch[i] == '+' || ch[i] == '-')

jj=1;

i++;

}

if ((ch[i + 1] == '\*' || ch[i + 1] == '/')&&jj==1)

{

return 1;

}

else

{

return 0;

}

}

void g(char \*a)

{

int i = 0;

while (a[i]!='\0')

{

if (a[i] == '(')

if (f(&a[i + 1], a[i - 1]) == 0)

q(&a[i]);

else

{

while (a[i] != ')')

{

i++;

}

}

i++;

}

}

int main()

{

int l = 0, i = 0;

char ch[100];

scanf("%s", ch);

g(ch);

while (ch[i]!='\0')

{

if (ch[i]!='#')

{

printf("%c", ch[i]);

}

i++;

}

return 0;

}

**算法训练 邮票**

问题描述

　　给定一个信封，有N（1≤N≤100）个位置可以贴邮票，每个位置只能贴一张邮票。我们现在有M(M<=100)种不同邮资的邮票，面值为X1,X2….Xm分（Xi是整数，1≤Xi≤255），每种都有N张。  
  
　　显然，信封上能贴的邮资最小值是min(X1, X2, …, Xm)，最大值是 N\*max(X1, X2, …,　　Xm)。由所有贴法得到的邮资值可形成一个集合（集合中没有重复数值），要求求出这个集合中是否存在从1到某个值的连续邮资序列，输出这个序列的最大值。  
  
　　例如，N=4，M=2，面值分别为4分，1分，于是形成1，2，3，4，5，6，7，8，9，10，12，13，16的序列，而从1开始的连续邮资序列为1，2，3，4，5，6，7，8，9，10，所以连续邮资序列的最大值为10分。

输入格式

　　第一行：最多允许粘贴的邮票张数N；第二行：邮票种数M；第三行：空格隔开的M个数字，表示邮票的面值Xi。注意：Xi序列不一定是大小有序的！

输出格式

　　从1开始的连续邮资序列的最大值MAX。若不存在从1分开始的序列（即输入的邮票中没有1分面额的邮票），则输出0.

样例输入

样例一：

4

2

4 1

样例二：

10

5

2 4 6 8 10

样例输出

样例一：

10

样例二：

0

本题的C++参考代码如下：

#include<stdio.h>

#include<string.h>

#define N 110

int d[N\*260],c[N];

int n,m,ans;

int main()

{

scanf("%d%d",&n,&m);

int i;

for(i=1;i<=m;i++) scanf("%d",&c[i]);

memset(d,0,sizeof(d));

ans=0;

while(true)

{

ans++;

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

if(ans-c[i]>=0)

if(d[ans]==0 || d[ans]>d[ans-c[i]]+1)

d[ans]=d[ans-c[i]]+1;

if(d[ans]==0 || d[ans]>n)

{

printf("%d\n",ans-1);

break;

}

}

return 0;

}本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

int main()

{

int N,M,i,Max,zuixiaozhi,t,found;//N个位置。M种邮资

int youzi[260];

int dp[26000];

scanf("%d %d",&N,&M);

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

{

scanf("%d",&youzi[i]);

}

memset(dp,0,sizeof(dp));

Max=0;

while(1)

{

Max++;

found=0;

zuixiaozhi=10000000;

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

{

t=Max-youzi[i];

if(t>=0&&dp[t]+1<zuixiaozhi)

{

dp[Max]=dp[t]+1;

zuixiaozhi=dp[t]+1;

found=1;

}

}

//printf("%d\n",dp[Max]);

if(dp[Max]>N||found==0)

{

printf("%d\n",Max-1);

break;

}

}

return 0;

}

**算法训练 矩阵加法**

问题描述

　　给定两个N×M的矩阵，计算其和。其中：  
　　N和M大于等于1且小于等于100，矩阵元素的绝对值不超过1000。

输入格式

　　输入数据的第一行包含两个整数N、M，表示需要相加的两个矩阵的行数和列数。接下来2\*N行每行包含M个数，其中前N行表示第一个矩阵，后N行表示第二个矩阵。

输出格式

　　你的程序需要输出一个N\*M的矩阵，表示两个矩阵相加的结果。注意，输出中每行的最后不应有多余的空格，否则你的程序有可能被系统认为是Presentation　　Error

样例输入

2 2

1 2

3 4

5 6

7 8

样例输出

6 8

10 12

本题的C++参考代码如下：

#include <stdio.h>

main()

{

int n,m;

int i,j;

int a[100][100];

int b[100][100];

int c[100][100];

scanf("%d %d",&n,&m);

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

{

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

{

scanf("%d",&a[i][j]);

}

}

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

{

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

{

scanf("%d",&b[i][j]);

}

}

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

{

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

{

c[i][j]=a[i][j]+b[i][j];

}

}

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

{

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

{

printf("%d\t",c[i][j]);

}

printf("\n");

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main(void)

{

static int a[100][100],b[100][100];

int n,m;

int i,j;

scanf("%d%d",&n,&m);

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

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

scanf("%d",&a[i][j]);

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

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

scanf("%d",&b[i][j]);

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

{

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

{

a[i][j]+=b[i][j];

printf("%d ",a[i][j]);

}

printf("\n");

}

return 0;

}

**算法训练 简单加法(基本型)**

问题描述

　　首先给出简单加法算式的定义：  
　　如果有一个算式(i)+(i+1)+(i+2),(i>=0)，在计算的过程中，没有任何一个数位出现了进位，则称其为简单的加法算式。  
　　例如：i=3时，3+4+5=12，有一个进位，因此3+4+5不是一个简单的加法算式；又如i=112时，112+113+114=339，没有在任意数位上产生进位，故112+113+114是一个简单的加法算式。  
  
　　问题：给定一个正整数n，问当i大于等于0且小于n时,有多少个算式(i)+(i+1)+(i+2)是简单加法算式。其中n<10000。

输入格式

　　一个整数，表示n

输出格式

　　一个整数,表示简单加法算式的个数

样例输入

4

样例输出

3

本题的C++参考代码如下：

#include<iostream>

using namespace std;

int f(int n)

{

if((3\*(n%10)+3)>=10)

{

return 0;

}

n/=10;

while(n)

{

if(3\*(n%10)>=10)

{

return 0;

}

n/=10;

}

return 1;

}

int main()

{

int n,cnt=0;

int i;

cin>>n;

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

{

if(f(i))

cnt++;

}

cout<<cnt;

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main(void)

{

int n,i;

int count=0;

int a,b,c,d;

scanf("%d",&n);

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

{

d=i%10;

c=i/10%10;

b=i/100%10;

a=i/1000;

if(a<=2&&b<=2&&c<=2&&d<=2)

count++;

}

printf("%d",count);

return 0;

}

**算法训练 排列问题**

问题描述

　　求一个0～N-1的排列（即每个数只能出现一次），给出限制条件（一张N\*N的表，第i行第j列的1或0，表示为j-1这个数不能出现在i-1这个数后面，并保证第i行第i列为0），将这个排列看成一个自然数，求从小到大排序第K个排列。

数据规模和约定

　　N<=10，K<=500000

输入格式

　　第一行为N和K,接下来的N行，每行N个数，0表示不能，1表示能

输出格式

　　所求的排列

样例输入

3 2

0 1 1

1 0 0

0 1 0

样例输出

1 0 2

解释：

对于N=3的没有任何限制的情况

第一：0 1 2

第二：0 2 1

第三：1 0 2

第四：1 2 0

第五：2 0 1

第六：2 1 0

根据题目所给的限制条件由于2不能出现在1后面，0不能出现在2后面

第一：0 2 1

第二：1 0 2

第三：2 1 0

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

#include "ctype.h"

#include "algorithm"

#include "stack"

using namespace std;

const int N =101;

int a[11];

bool map[11][11];

bool isOk(int n)

{

for(int i=0;i<n-1;i++)

if(map[ a[i] ][ a[i+1] ] == 0)

{

return false;

}

return true;

}

void init()

{

for(int i=0;i<10;i++)

a[i]=i;

}

int main()

{

int n;

int k;

cin>>n>>k;

for(int i=0;i<n;i++)

for(int j = 0; j < n; j++)

cin>>map[i][j];

init();

int num=0;

do

{

if(isOk(n))

num++;

if(num==k)

{

cout<<a[0];

for(int i=1;i<n;i++)

cout<<" "<<a[i];

cout<<endl;

break;

}

//cout<<endl;

} while (std::next\_permutation(a,a+n));

}

本题的C参考代码如下：

无

**算法训练 数组查找及替换**

问题描述

　　给定某整数数组和某一整数b。要求删除数组中可以被b整除的所有元素，同时将该数组各元素按从小到大排序。如果数组元素数值在A到Z的ASCII之间，替换为对应字母。元素个数不超过100，b在1至100之间。

输入格式

　　第一行为数组元素个数和整数b  
　　第二行为数组各个元素

输出格式

　　按照要求输出

样例输入

7 2

77 11 66 22 44 33 55

样例输出

11 33 55 M

本题的C++参考代码如下：

#include<iostream>

#include<algorithm>

using namespace std;

int arr[100];

int main()

{

int n,b;

int i,j;

cin>>n>>b;

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

{

cin>>arr[i];

if(arr[i]%b==0)

arr[i]=0;

}

sort(arr,arr+n);

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

{

if(arr[i])

{

if(arr[i]>=65 &&arr[i]<=90)

{

cout<<char(arr[i])<<" ";

}

else

cout<<arr[i]<<" ";

}

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

void del(int a[],int \*len,int m)

{

int i,j;

for(i=0;i<\*len;i++)

if(a[i]%m==0)

{

for(j=i;j<\*len;j++)

a[j]=a[j+1];

(\*len)--;

i--;

}

}

int cmp(const void \*a,const void \*b)

{

return \*(int \*)a-\*(int \*)b;

}

int main()

{

int a[101]={},i,m,n;

scanf("%d%d",&n,&m);

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

scanf("%d",&a[i]);

del(a,&n,m);

qsort(a,n,sizeof(a[0]),cmp);

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

{

if(a[i]>=65&&a[i]<=90)

printf("%c ",a[i]);

else

printf("%d ",a[i]);

}

return 0;

}

**算法训练 蜜蜂飞舞**

问题描述

　　“两只小蜜蜂呀，飞在花丛中呀……”  
  
　　话说这天天上飞舞着两只蜜蜂，它们在跳一种奇怪的舞蹈。用一个空间直角坐标系来描述这个世界，那么这两只蜜蜂初始坐标分别为(x1,y1,z1)，(x2,y2,z2)　　。在接下来它们将进行n次飞行，第i次飞行两只蜜蜂分别按照各自的速度向量飞行ti个单位时间。对于这一现象，玮玮已经观察了很久。他很想知道在蜜蜂飞舞结束时，两只蜜蜂的距离是多少。现在他就求教于你，请你写一个程序来帮他计算这个结果。

输入格式

　　第一行有且仅有一个整数n，表示两只蜜蜂将进行n次飞行。  
  
　　接下来有n行。  
  
　　第i行有7个用空格分隔开的整数ai,bi,ci,di,ei,fi,ti　　，表示第一只蜜蜂单位时间的速度向量为(ai,bi,ci) ，第二只蜜蜂单位时间的速度向量为(di,ei,fi) ，它们飞行的时间为ti 。  
  
　　最后一行有6个用空格分隔开的整数x1,y1,z1,x2,y2,z2，如题所示表示两只蜜蜂的初始坐标。

输出格式

　　输出仅包含一行，表示最后两只蜜蜂之间的距离。保留4位小数位。

样例输入

Sample 1

1

1 1 1 1 -1 1 2

3 0 1 2 0 0

Sample 2

3

1 1 1 1 -1 1 2

2 1 2 0 -1 -1 2

2 0 0 -1 1 1 3

3 0 1 2 0 0

样例输出

Sample 1

4.2426

Sample 2

15.3948

本题的C++参考代码如下：

#include<cstdio>

#include<cmath>

using namespace std;

int main(){

int x1=0,y1=0,z1=0,x2=0,y2=0,z2=0;

int a1,b1,c1,a2,b2,c2;

int t;

int n;

double s;

scanf("%d",&n);

for(int i=0;i<n;i++){

scanf("%d%d%d%d%d%d%d",&a1,&b1,&c1,&a2,&b2,&c2,&t);

x1+=a1\*t;

x2+=a2\*t;

y1+=b1\*t;

y2+=b2\*t;

z1+=c1\*t;

z2+=c2\*t;

}

scanf("%d%d%d%d%d%d",&a1,&b1,&c1,&a2,&b2,&c2);

x1+=a1;

x2+=a2;

y1+=b1;

y2+=b2;

z1+=c1;

z2+=c2;

s=sqrt((double)(abs(x2-x1)\*abs(x2-x1)+abs(y2-y1)\*abs(y2-y1)+abs(z2-z1)\*abs(z2-z1)));

printf("%.4lf\n",s);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <math.h>

int main()

{

int n;

double x1=0,y1=0,z1=0,x2=0,y2=0,z2=0;

int ai,bi,ci,di,ei,fi,ti;

int i;

scanf("%d",&n);

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

{

scanf("%d%d%d%d%d%d%d",&ai,&bi,&ci,&di,&ei,&fi,&ti);

x1+=ai\*ti;

y1+=bi\*ti;

z1+=ci\*ti;

x2+=di\*ti;

y2+=ei\*ti;

z2+=fi\*ti;

}

scanf("%d%d%d%d%d%d",&ai,&bi,&ci,&di,&ei,&fi);

x1+=ai;

y1+=bi;

z1+=ci;

x2+=di;

y2+=ei;

z2+=fi;

printf("%.4lf",sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)+(z1-z2)\*(z1-z2)));

return 0;

}

**算法训练 Hanoi问题**

问题描述

　　如果将课本上的Hanoi塔问题稍做修改：仍然是给定N只盘子，3根柱子，但是允许每次最多移动相邻的M只盘子（当然移动盘子的数目也可以小于M）,最少需要多少次？  
　　例如N=5，M=2时，可以分别将最小的2个盘子、中间的2个盘子以及最大的一个盘子分别看作一个整体，这样可以转变为N=3，M=1的情况，共需要移动7次。

输入格式

　　输入数据仅有一行，包括两个数N和M（0<=M<=N<=8）

输出格式

　　仅输出一个数，表示需要移动的最少次数

样例输入

5 2

样例输出

7

本题的C++参考代码如下：

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int n,m;

cin>>n>>m;

if(n%m==0)

n/=m;

else n=n/m+1;

cout<<pow(2,n)-1;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int N,M;

scanf("%d%d",&N,&M);

N=(N+1)/M;

printf("%d",(1<<N)-1);

return 0;

}

**算法训练 调和数列问题**

问题描述

　　输入一个实数x，求最小的n使得，1/2+1/3+1/4+...+1/(n+1)>=x。  
  
　　输入的实数x保证大于等于0.01，小于等于5.20，并且恰好有两位小数。你的程序要能够处理多组数据，即不停地读入x，如果x不等于0.00，则计算答案，否则退出程序。  
  
　　输出格式为对于一个x，输出一行n card(s)。其中n表示要计算的答案。

输入格式

　　分行输入x的具体数值

输出格式

　　分行输出n的数值，格式为n card(s)

样例输入

1.00

3.71

0.04

5.19

0.00

样例输出

3 card(s)

61 card(s)

1 card(s)

273 card(s)

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

#include "ctype.h"

#include "algorithm"

#include "stack"

#include "math.h"

using namespace std;

const int N =101;

double fen(int i)

{

return 1.0/i;

}

int main()

{

double x;

while(cin>>x, x!=0.00)

{

double ans=0;

for(int i=2;;i++)

{

ans+=fen(i);

if(ans>=x)

{

cout<<i-1<<" "<<"card(s)"<<endl;

break;

}

}

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main(void)

{

int i;

double n,k,s;

scanf("%lf",&n);

while(n!=0)

{

s=0;

for(i=2;1;i++)

{

k=1.0/i;

s+=k;

if(s>=n)

{

printf("%d card(s)\n",i-1);

break;

}

}

scanf("%lf",&n);

}

return 0;

}

**算法训练 会议中心**

　会议中心　　Siruseri政府建造了一座新的会议中心。许多公司对租借会议中心的会堂很感兴趣，他们希望能够在里面举行会议。  
　　对于一个客户而言，仅当在开会时能够独自占用整个会堂，他才会租借会堂。会议中心的销售主管认为：最好的策略应该是将会堂租借给尽可能多的客户。显然，有可能存在不止一种满足要求的策略。  
　　例如下面的例子。总共有4个公司。他们对租借会堂发出了请求，并提出了他们所需占用会堂的起止日期（如下表所示）。

|  |  |  |
| --- | --- | --- |
|  | 开始日期 | 结束日期 |
| 公司1 | 4 | 9 |
| 公司2 | 9 | 11 |
| 公司3 | 13 | 19 |
| 公司4 | 10 | 17 |

　　上例中，最多将会堂租借给两家公司。租借策略分别是租给公司1和公司3，或是公司2和公司3，也可以是公司1和公司4。注意会议中心一天最多租借给一个公司，所以公司1和公司2不能同时租借会议中心，因为他们在第九天重合了。  
　　销售主管为了公平起见，决定按照如下的程序来确定选择何种租借策略：首先，将租借给客户数量最多的策略作为候选，将所有的公司按照他们发出请求的顺序编号。对于候选策略，将策略中的每家公司的编号按升序排列。最后，选出其中字典序最小[1]的候选策略作为最终的策略。  
　　例中，会堂最终将被租借给公司1和公司3：3个候选策略是{(1,3),(2,3),(1,4)}。而在字典序中(1,3)<(1,4)<(2,3)。  
　　你的任务是帮助销售主管确定应该将会堂租借给哪些公司。

输入格式

　　输入的第一行有一个整数*N*，表示发出租借会堂申请的公司的个数。第2到第*N*+1行每行有2个整数。第*i*+1行的整数表示第*i*家公司申请租借的起始和终止日期。对于每个公司的申请，起始日期为不小于1的整数，终止日期为不大于109的整数。

输出格式

　　输出的第一行应有一个整数*M*，表示最多可以租借给多少家公司。第二行应列出*M*个数，表示最终将会堂租借给哪些公司。

数据规模和约定

　　对于50%的输入，*N*≤3000。在所有输入中，*N*≤200000。

样例输入

4  
4 9  
9 11  
13 19  
10 17

样例输出

2  
1 3  
  
[1] 字典序指在字典中排列的顺序，如果序列*l*1是序列*l*2的前缀，或者对于*l*1和*l*2的第一个不同位置*j*，*l*1[*j*]<*l*2[*j*]，则*l*1比*l*2小。

本题的C++参考代码如下：

#include <cstdio>

#include <cstdlib>

#include <algorithm>

#include <string>

#include <set>

const char fi[] = "convention.in";

const char fo[] = "convention.out";

const int maxN = 200010;

const int MAX = 0x3f3f3f3f,MIN = ~MAX;

struct Seg

{

int L,R;

Seg()

{

}

Seg(int L,int R): L(L),R(R)

{

}

bool operator<(const Seg &b) const

{

return L < b.L || L == b.L && R < b.R;

}

};

std::set <Seg> S;

std::set <Seg>::iterator iter;

Seg req[maxN],seg[maxN],tmp[maxN];

int tab[maxN << 1],next[20][maxN << 1];

int n,cnt,Lim = 1,logLim;

void init\_file()

{

return;

}

inline int getint()

{

int res = 0; char tmp;

while(!isdigit(tmp = getchar()));

do res = (res << 3) + (res << 1) + tmp - '0';

while(isdigit(tmp = getchar()));

return res;

}

void readdata()

{

n = getint();

for(int i = 0; i < n; ++i)

{

int L = getint(),R = getint();

req[i] = Seg(L,R);

tab[i << 1] = L;

tab[(i << 1) + 1] = R;

}

return;

}

int plc(int x)

{

for(int L = 0,R = Lim - 1; L < R + 1;)

{

int Mid = (L + R) >> 1;

if(x == tab[Mid]) return Mid + 1;

if(x < tab[Mid]) R = Mid - 1;

else L = Mid + 1;

}

}

bool cmp(const Seg &a,const Seg &b)

{

return a.R < b.R || a.R == b.R && a.L > b.L;

}

void discrete()

{

std::sort(tab,tab + (n << 1));

for(int i = 1; i < n << 1; ++i)

if(tab[i] != tab[i - 1])

tab[Lim++] = tab[i];

for(int i = 0; i < n; ++i)

tmp[i] = req[i] = Seg(plc(req[i].L),

plc(req[i].R));

std::sort(tmp,tmp + n,cmp);

//这里必须要用一个临时数组，

//保证左界右界同时单调递增。

int p = 0; seg[cnt++] = tmp[0];

for(int i = 1; i < n; ++i)

if(tmp[i].L > tmp[p].L)

seg[cnt++] = tmp[p = i];

return;

}

void next\_set()

{

int p = cnt; next[0][Lim + 1] = MAX;

for(int j = Lim; j; --j)

if(p > -1 && j == seg[p - 1].L)

next[0][j] = seg[--p].R + 1;

else next[0][j] = next[0][j + 1];

for(int i = 0;; ++i)

{

bool flag = 0;

next[i + 1][Lim + 1] = MAX;

for(int k = 1; k < Lim + 1; ++k)

{

if(next[i][k] == MAX)

next[i + 1][k] = MAX;

else next[i + 1][k] = next[i][next[i][k]];

if(next[i + 1][k] < MAX) flag = 1;

}

if(!flag)

{

logLim = i; break;

}

}

return;

}

int max\_time(int L,int R)

{

if(L > R++) return 0;

int ans = 0,p = L;

for(int i = logLim; i > -1 && p < R; --i)

if(next[i][p] <= R)

{

p = next[i][p]; ans += 1 << i;

}

return ans;

}

bool query(int i)

{

int L = req[i].L,R = req[i].R;

iter = S.lower\_bound(Seg(L,MAX));

if(iter-- == S.begin()) return 0;

if(iter->L > L || iter->R < R)

return 0;

int L1 = iter->L,R1 = iter->R;

if(max\_time(L1,L - 1)

+ max\_time(R + 1,R1)

+ 1 < max\_time(L1,R1))

//这里要满足放进去过后不影响总的答案。

return 0;

S.erase(iter);

if(L1 < L) S.insert(Seg(L1,L - 1));

if(R < R1) S.insert(Seg(R + 1,R1));

return 1;

}

void work()

{

printf("%d\n",max\_time(1,Lim));

S.insert(Seg(1,Lim));

for(int i = 0; i < n; ++i)

if(query(i))

printf("%d ",i + 1);

printf("\n");

return;

}

int main()

{

init\_file();

readdata();

discrete();

next\_set();

work();

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。

**算法训练 采油区域**

采油区域　　Siruseri政府决定将石油资源丰富的Navalur省的土地拍卖给私人承包商以建立油井。被拍卖的整块土地为一个矩形区域，被划分为*M*×*N*个小块。  
　　Siruseri地质调查局有关于Navalur土地石油储量的估测数据。这些数据表示为*M*×*N*个非负整数，即对每一小块土地石油储量的估计值。  
　　为了避免出现垄断，政府规定每一个承包商只能承包一个由*K*×*K*块相连的土地构成的正方形区域。  
　　AoE石油联合公司由三个承包商组成，他们想选择三块互不相交的*K*×*K*的区域使得总的收益最大。  
　　例如，假设石油储量的估计值如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 |
| 1 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 |
| 1 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 8 | 8 | 8 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 8 | 8 | 8 |
| 1 | 1 | 1 | 1 | 1 | 1 | 9 | 9 | 9 |
| 1 | 1 | 1 | 1 | 1 | 1 | 9 | 9 | 9 |

　　如果*K* = 2, AoE公司可以承包的区域的石油储量总和为100, 如果*K* = 3, AoE公司可以承包的区域的石油储量总和为208。  
　　AoE公司雇佣你来写一个程序，帮助计算出他们可以承包的区域的石油储量之和的最大值。

输入格式

　　输入第一行包含三个整数*M*, *N*, *K*，其中*M*和*N*是矩形区域的行数和列数，*K*是每一个承包商承包的正方形的大小（边长的块数）。接下来*M*行，每行有*N*个非负整数表示这一行每一小块土地的石油储量的估计值。

输出格式

　　输出只包含一个整数，表示AoE公司可以承包的区域的石油储量之和的最大值。

数据规模和约定

　　数据保证*K*≤*M*且*K*≤*N*并且至少有三个*K*×*K*的互不相交的正方形区域。其中30%的输入数据，*M*, *N*≤ 12。所有的输入数据, *M*, *N*≤ 1500。每一小块土地的石油储量的估计值是非负整数且≤ 500。

样例输入

9 9 3  
1 1 1 1 1 1 1 1 1  
1 1 1 1 1 1 1 1 1  
1 8 8 8 8 8 1 1 1  
1 8 8 8 8 8 1 1 1  
1 8 8 8 8 8 1 1 1  
1 1 1 1 8 8 8 1 1  
1 1 1 1 1 1 8 8 8  
1 1 1 1 1 1 9 9 9  
1 1 1 1 1 1 9 9 9

样例输出

208

本题的C++参考代码如下：

#include <cstdio>

#include <cstdlib>

#include <algorithm>

#include <string>

#define max(a, b) ((a) > (b) ? (a) : (b))

#define max3(a, b, c) (max((a), max(b, c)))

const char fi[] = "oil.in";

const char fo[] = "oil.out";

const int maxN = 1510;

const int MAX = 0x3f3f3f3f;

const int MIN = ~MAX;

int sum[maxN][maxN];

int zoxx[maxN][maxN]; // Left and below.

int zouh[maxN][maxN]; // Left and above.

int yzxx[maxN][maxN]; // Right and below.

int yzuh[maxN][maxN]; // Right and above.

int hh[maxN]; // Horizon.

int lp[maxN]; // Vertical.

int n,m,K;

void init\_file()

{

return;

}

inline int getint()

{

int res = 0; char tmp;

while(!isdigit(tmp = getchar()));

do res = (res << 3) + (res << 1) + tmp - '0';

while(isdigit(tmp = getchar()));

return res;

}

void readdata()

{

n = getint(); m = getint(); K = getint();

for(int i = 1; i < n + 1; ++i)

for(int j = 1; j < m + 1; ++j)

(sum[i][j] = getint()) += sum[i][j - 1];

for(int i = 1; i < n + 1; ++i)

for(int j = 1; j < m + 1; ++j)

sum[i][j] += sum[i - 1][j];

//预处理出二维的前缀和。

for(int i = K; i < n + 1; ++i)

for(int j = K; j < m + 1; ++j)

{

zouh[i][j] = sum[i][j]

+ sum[i - K][j - K]

- sum[i][j - K]

- sum[i - K][j];

hh[i] = max(hh[i],zouh[i][j]); //Horizon.

lp[j] = max(lp[j],zouh[i][j]); //Vertical.

zouh[i][j] = max3(zouh[i][j],

zouh[i - 1][j],

zouh[i][j - 1]);

} // Left and above.

for(int i = n - K + 1; i; --i)

for(int j = m - K + 1; j; --j)

yzxx[i][j] = max3(sum[i - 1][j - 1]

+ sum[i + K - 1][j + K - 1]

- sum[i - 1][j + K - 1]

- sum[i + K - 1][j - 1],

yzxx[i + 1][j],

yzxx[i][j + 1]);

//Right and below.

for(int i = K; i < n + 1; ++i)

for(int j = m - K + 1; j; --j)

yzuh[i][j] = max3(sum[i][j + K - 1]

+ sum[i - K][j - 1]

- sum[i][j - 1]

- sum[i - K][j + K - 1],

yzuh[i - 1][j],

yzuh[i][j + 1]);

//Right and above.

for(int i = n - K + 1; i; --i)

for(int j = K; j < m + 1; ++j)

zoxx[i][j] = max3(sum[i - 1][j - K]

+ sum[i + K - 1][j]

- sum[i - 1][j]

- sum[i + K - 1][j - K],

zoxx[i + 1][j],

zoxx[i][j - 1]);

//Left and below

return;

}

void work()

{

int ans = 0;

for(int i = K; i < n - (K << 1); ++i)

ans = max(ans,yzuh[i][1]

+ hh[i + K]

+ yzxx[i + K + 1][1]);

// Case 1.

for(int j = K; j < m - (K << 1); ++j)

ans = max(ans,zoxx[1][j]

+ lp[j + K]

+ yzxx[1][j + K + 1]);

// Case 2.

for(int i = K; i < n - K + 1; ++i)

for(int j = K; j < m - K + 1; ++j)

{

ans = max3(ans,zouh[i][j]

+ yzuh[i][j + 1]

+ yzxx[i + 1][1], //Case 3.

yzuh[i][1]

+ zoxx[i + 1][j]

+ yzxx[i + 1][j + 1]); //Case 4.

ans = max3(ans,zouh[i][j]

+ zoxx[i + 1][j]

+ yzxx[1][j + 1], //Case 5.

zoxx[1][j]

+ yzuh[i][j + 1]

+ yzxx[i + 1][j + 1]); //Case 6.

}

printf("%d\n",ans);

return;

}

int main()

{

init\_file();

readdata();

work();

return 0;

}

#undef max

本题的C参考代码如下：

#include <stdio.h>

#include<stdlib.h>

long n,m,K;

long map[1510][1510];

long sum1[1510][1510];//left and above

long sum2[1510][1510];//right and above

long sum3[1510][1510];//left and below

long sum4[1510][1510];//right and below

long rect1[1510][1510];//left and above

long rect2[1510][1510];//right and above

long rect3[1510][1510];//left and below

long rect4[1510][1510];//right and below

long rectA[1510][1510];

long rectB[1510][1510];

#define MAX(a,b) (a>b?a:b)

int main()

{

long i,j;

long ans = 0;

scanf("%ld%ld%ld",&n,&m,&K);

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

{

for (j=1;j<m+1;j++)

{

scanf("%ld",map[i]+j);

sum1[i][j] = sum1[i][j-1]+map[i][j];

sum3[i][j] = sum1[i][j];

}

}

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

{

for (j=m;j>0;j--)

{

sum2[i][j] = sum2[i][j+1]+map[i][j];

sum4[i][j] = sum2[i][j];

}

}

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

{

for (j=1;j<m+1;j++)

{

sum1[i][j] += sum1[i-1][j];

sum2[i][j] += sum2[i-1][j];

}

}

for (i=n;i>0;i--)

{

for (j=m;j>0;j--)

{

sum3[i][j] += sum3[i+1][j];

sum4[i][j] += sum4[i+1][j];

}

}

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

{

for (j=1;j<m+1;j++)

{

if (i-K>=0 && j-K>=0)

{

rect1[i][j] = sum1[i][j]-sum1[i][j-K]-sum1[i-K][j]+sum1[i-K][j-K];//left and above

rectA[i][j] = rect1[i][j];

}

if (i-K>=0 && j+K-1<=m)

rect2[i][j] = sum2[i][j]-sum2[i][j+K]-sum2[i-K][j]+sum2[i-K][j+K];//right and above

if (i+K-1<=n && j-K>=0)

{

rect3[i][j] = sum3[i][j]-sum3[i][j-K]-sum3[i+K][j]+sum3[i+K][j-K];//left and blow

rectB[i][j] = rect3[i][j];

}

if (i+K-1<=n && j+K-1<=m)

rect4[i][j] = sum4[i][j]-sum4[i][j+K]-sum4[i+K][j]+sum4[i+K][j+K];//right and blow

}

}

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

{

for (j=1;j<m+1;j++)

{

rect1[i][j] = MAX(rect1[i][j],rect1[i-1][j]);

rect1[i][j] = MAX(rect1[i][j],rect1[i][j-1]);

rectA[i][j] = MAX(rectA[i][j],rectA[i][j-1]);

rectB[i][j] = MAX(rectB[i][j],rectB[i-1][j]);

}

}

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

{

for (j=m;j>0;j--)

{

rect2[i][j] = MAX(rect2[i][j],rect2[i-1][j]);

rect2[i][j] = MAX(rect2[i][j],rect2[i][j+1]);

}

}

for (i=n;i>0;i--)

{

for (j=1;j<m+1;j++)

{

rect3[i][j] = MAX(rect3[i][j],rect3[i+1][j]);

rect3[i][j] = MAX(rect3[i][j],rect3[i][j-1]);

}

}

for (i=n;i>0;i--)

{

for (j=m;j>0;j--)

{

rect4[i][j] = MAX(rect4[i][j],rect4[i+1][j]);

rect4[i][j] = MAX(rect4[i][j],rect4[i][j+1]);

}

}

for (i=1+K;i<n-K+2;i++)

{

for (j=1+K;j<m-K+2;j++)

{

ans = MAX(ans,rect1[i-1][j-1]+rect2[i-1][j]+rect4[i][1]);

ans = MAX(ans,rect3[i][j-1]+rect4[i][j]+rect2[i-1][1]);

ans = MAX(ans,rect1[i-1][j-1]+rect3[i][j-1]+rect4[1][j]);

ans = MAX(ans,rect2[i-1][j]+rect4[i][j]+rect3[1][j-1]);

}

}

for (i=1+K;i<n-2\*K+1;i++)

{

ans = MAX(ans,rect2[i-1][1]+rect4[i+K][1]+rectA[i+K-1][m]);

}

for (j=1+K;j<m-2\*K+1;j++)

{

ans = MAX(ans,rect3[1][j-1]+rect4[1][j+K]+rectB[n][j+K-1]);

}

printf("%ld",ans);

system("pause");

return 0;

}

**算法训练 A+B Problem**

问题描述

　　输入A,B。  
　　输出A+B。

输入格式

　　输入包含两个整数A,B，用一个空格分隔。

输出格式

　　输出一个整数，表示A+B的值。

样例输入

5 8

样例输出

13

数据规模和约定

　　-1,000,000,000<=A,B<=1,000,000,000。

本题的C++参考代码如下：

#include<cstdio>

#include<iostream>

using namespace std;

int main()

{

int a,b;

cin>>a;

cin>>b;

cout<<a+b;

cout<<'\n';

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int sum(int a,int b)

{

return a+b;

}

int main()

{

int a,b;

scanf("%d%d",&a,&b);

//sum(a,b);

printf("%d",sum(a,b));

return 0;

}

**算法训练 数组排序去重**

问题描述

　　输入10个整数组成的序列，要求对其进行升序排序，并去掉重复元素。

输入格式

　　10个整数。

输出格式

　　多行输出，每行一个元素。

样例输入

2 2 3 3 1 1 5 5 5 5

样例输出

1  
2  
3  
5

本题的C++参考代码如下：

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

int i,t;

vector<int> v;

for (i = 0; i < 10; i++)

{

cin>>t;

v.push\_back(t);

}

sort(v.begin(),v.end());

v.erase(unique(v.begin(),v.end()),v.end());//unique(n.begin(),n.end())；unique()把相邻元素重复的甩到后面//erase()用来删掉后面的重复元素.//一定要先排序，再使用unique()，

for (i = 0; i < v.size(); i++)

cout<<v[i]<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

int A[10];

int com(const void \*a, const void \*b)

{

return \*(int\*)a-\*(int\*)b;

}

int main()

{

int i;

int last;

for(i=0;i<10;i++)

scanf("%d",&A[i]);

qsort(A,10,4,com);

for(i=0;i<10;i++)

if(i)

{

if(last!=A[i])

{

printf("%d\n",A[i]);

last=A[i];

}

}

else

{

printf("%d\n",A[i]);

last=A[i];

}

return 0;

}

**算法训练 接水问题**

问题描述

　　学校里有一个水房，水房里一共装有m 个龙头可供同学们打开水，每个龙头每秒钟的 供水量相等，均为1。 现在有n 名同学准备接水，他们的初始接水顺序已经确定。将这些同学按接水顺序从1 到n 编号，i 号同学的接水量为wi。接水开始时，1 到m 号同学各占一个水龙头，并同时打 开水龙头接水。当其中某名同学j 完成其接水量要求wj 后，下一名排队等候接水的同学k 马上接替j 同学的位置开始接水。这个换人的过程是瞬间完成的，且没有任何水的浪费。即 j 同学第x 秒结束时完成接水，则k 同学第x+1 秒立刻开始接水。若当前接水人数n’不足m， 则只有n’个龙头供水，其它m−n’个龙头关闭。 现在给出n 名同学的接水量，按照上述接水规则，问所有同学都接完水需要多少秒。

输入格式

　　第1 行2 个整数n 和m，用一个空格隔开，分别表示接水人数和龙头个数。 第2 行n 个整数w1、w2、……、wn，每两个整数之间用一个空格隔开，wi 表示i 号同 学的接水量。

输出格式

　　输出只有一行，1 个整数，表示接水所需的总时间。

样例输入

5 3  
4 4 1 2 1

样例输出

4

样例输入

8 4  
23 71 87 32 70 93 80 76

样例输出

163

输入输出样例 1 说明

　　第1 秒，3 人接水。第1 秒结束时，1、2、3 号同学每人的已接水量为1，3 号同学接完  
　　水，4 号同学接替3 号同学开始接水。  
　　第2 秒，3 人接水。第2 秒结束时，1、2 号同学每人的已接水量为2，4 号同学的已接  
　　水量为1。  
　　第3 秒，3 人接水。第3 秒结束时，1、2 号同学每人的已接水量为3，4 号同学的已接  
　　水量为2。4 号同学接完水，5 号同学接替4 号同学开始接水。  
　　第4 秒，3 人接水。第4 秒结束时，1、2 号同学每人的已接水量为4，5 号同学的已接  
　　水量为1。1、2、5 号同学接完水，即所有人完成接水。  
　　总接水时间为4 秒。

数据规模和约定

　　1 ≤ n ≤ 10000，1 ≤m≤ 100 且m≤ n；  
　　1 ≤ wi ≤ 100。

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

#include "ctype.h"

#include "algorithm"

#include "stack"

#include "list"

#include "math.h"

using namespace std;

const int N =101;

struct X

{

int total;

int now;

X(int t=0,int n=0):total(t),now(n){}

}a[10001],b[201];

int main()

{

int n,m;

scanf("%d%d",&n,&m);

for(int i=0;i<n;i++)

scanf("%d",&a[i].total);

int index=0;

for(int i=0;i<m;i++)

{

b[i]=a[ index++];

}

int time=0;

while(true)

{

time++;

for(int i=0;i<m;i++)

b[i].now++;

for(int i=0;i<m;i++)

{

if(b[i].now>=b[i].total)

{

if(index<n)

{

b[i]=a[index++];

}

}

}

int i;

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

{

if(b[i].now<b[i].total)

break;

}

if(i==m)

break;

}

cout<<time<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int max(int \*p,int n)

{

int Max = p[0];

int i;

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

{

if(p[i]>Max)

Max = p[i];

}

return Max;

}

int main(int argc, char \*argv[]) {

int m,n; //m个水龙头 n个同学

int i,j,second,min,num;

int \*p;

scanf("%d%d",&n,&m);

p = (int \*)malloc(n\*sizeof(int));

if( NULL == p )

{

printf("no enough memory!\n");

return 0;

}

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

{

scanf("%d",&p[i]);

}

if( n<=m)

{

printf("%d",max(p,n));

}

if( n>m)

{

second = 0;

num = 0;

while(n>m)

{

min = p[0];

j = 0;

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

{

if(min>p[i])

{

min = p[i];

j=i;

}

}

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

{

p[i] -= min;

}

second += min;

p[j]=p[m+num]; //第m+num同学取代取完水的同学

num++;

n--;//取水同学减少一个

if( n<=m)

{

printf("%d",max(p,n)+second);

}

}

}

return 0;

}

**算法训练 Hankson的趣味题**

问题描述

　　Hanks 博士是BT (Bio-Tech，生物技术) 领域的知名专家，他的儿子名叫Hankson。现 在，刚刚放学回家的Hankson 正在思考一个有趣的问题。 今天在课堂上，老师讲解了如何求两个正整数c1 和c2 的最大公约数和最小公倍数。现 在Hankson 认为自己已经熟练地掌握了这些知识，他开始思考一个“求公约数”和“求公 倍数”之类问题的“逆问题”，这个问题是这样的：已知正整数a0,a1,b0,b1，设某未知正整 数x 满足： 1． x 和a0 的最大公约数是a1； 2． x 和b0 的最小公倍数是b1。 Hankson 的“逆问题”就是求出满足条件的正整数x。但稍加思索之后，他发现这样的 x 并不唯一，甚至可能不存在。因此他转而开始考虑如何求解满足条件的x 的个数。请你帮 助他编程求解这个问题。

输入格式

　　输入第一行为一个正整数n，表示有n 组输入数据。  
  
　　接下来的n 行每 行一组输入数据，为四个正整数a0，a1，b0，b1，每两个整数之间用一个空格隔开。输入 数据保证a0 能被a1 整除，b1 能被b0 整除。

输出格式

　　输出共n 行。每组输入数据的输出结果占一行，为一个整数。  
　　对于每组数据：若不存在这样的 x，请输出0； 若存在这样的 x，请输出满足条件的x 的个数；

样例输入

2  
41 1 96 288  
95 1 37 1776

样例输出

6  
2

样例说明

　　第一组输入数据，x 可以是9、18、36、72、144、288，共有6 个。  
　　第二组输入数据，x 可以是48、1776，共有2 个。

数据规模和约定

　　对于 50%的数据，保证有1≤a0，a1，b0，b1≤10000 且n≤100。  
　　对于 100%的数据，保证有1≤a0，a1，b0，b1≤2,000,000,000 且n≤2000。

本题的C++参考代码如下：

#include<stdio.h>

#include<math.h>

const int maxn = 55000;

const int pmaxn = 5500;

int prime[pmaxn], p, ans = 1, tt;

bool vis[maxn];

int div(int & a, int b)

{

int cnt = 0;

while(a % b == 0)

{

a /= b;

cnt++;

}

return cnt;

}

int main()

{

int test = 50010;

int t = 0, n, m = (int)sqrt(maxn+0.5), i;

for(i = 2; i <= m; i++)

{

if(vis[i]) continue;

prime[t++] = i;

for(int j = i \* i; j < maxn; j += i)

vis[j] = true;

}

for(; i < maxn; i++)

if(!vis[i])

prime[t++] = i;

scanf("%d", &n);

while(n--)

{

int a, aa, b, bb;

ans = 1;

tt = 0;

scanf("%d%d%d%d", &a, &aa, &b, &bb);

for(i = 0; bb != 1; i++)

{

if(i > 4950)

{

prime[i] = bb;

// break;

}

if(bb % prime[i] == 0)

{

int cnt = 100000000, type;

int a1c = div(aa, prime[i]);

int a0c = div(a, prime[i]);

if(a1c > a0c) continue;

if(a1c == a0c)

{

cnt = a1c;

type = 1;

}

if(a1c < a0c)

{

type = 2;

cnt = a1c;

}

int b0c = div(b, prime[i]);

int b1c = div(bb, prime[i]);

int tot;

if(b1c == b0c)

{

if(type == 1)

tot = b1c - cnt + 1;

else

{

if(cnt > b1c)

continue;

tot = 1;

}

if(tot <= 0) continue;

}

if(b1c > b0c)

{

if(type == 2)

{

if(b1c != cnt)

{

ans = 0;break;

}

tot = 1;

}

else

{

if(cnt > b1c)

continue;

tot = 1;

}

}

ans \*= tot;

}

}

printf("%d\n", ans);

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int gcd(int x,int y)

{

return y?gcd(y,x%y):x;

}

int main()

{

int n,a0,a1,b0,b1,x,cnt;

scanf("%d",&n);

while(n--)

{

scanf("%d%d%d%d",&a0,&a1,&b0,&b1);

cnt=0;

for(x=1;x\*x<=b1;x++)

{

if(b1%x==0)//如果x确实是b1的约数

{

if(x%a1==0)//如果x确实是a1的倍数

{

if(gcd(x,a0)==a1&&(x/gcd(x,b0)\*b0==b1))//乘以b0放在后面，否则运算结果溢出，尤其是第2个输入用例

cnt++;

}

if((b1/x)%a1==0&&x\*x!=b1)//如果b1/x确实是a1的倍数,并且x与b1/x不相等

{

if(gcd(b1/x,a0)==a1&&(b1/x/gcd(b1/x,b0)\*b0==b1))//乘以b0放在后面，否则运算结果溢出

cnt++;

}

}

}

printf("%d\n",cnt);

}

return 0;

}

**算法训练 传纸条**

问题描述

　　小渊和小轩是好朋友也是同班同学，他们在一起总有谈不完的话题。一次素质拓展活动中，班上同学安排做成一个m行n列的矩阵，而小渊和小轩被安排在矩阵对角线的两端，因此，他们就无法直接交谈了。幸运的是，他们可以通过传纸条来进行交流。纸条要经由许多同学传到对方手里，小渊坐在矩阵的左上角，坐标(1,1)，小轩坐在矩阵的右下角，坐标(m,n)。从小渊传到小轩的纸条只可以向下或者向右传递，从小轩传给小渊的纸条只可以向上或者向左传递。  
　　在活动进行中，小渊希望给小轩传递一张纸条，同时希望小轩给他回复。班里每个同学都可以帮他们传递，但只会帮他们一次，也就是说如果此人在小渊递给小轩纸条的时候帮忙，那么在小轩递给小渊的时候就不会再帮忙。反之亦然。  
　　还有一件事情需要注意，全班每个同学愿意帮忙的好感度有高有低（注意：小渊和小轩的好心程度没有定义，输入时用0表示），可以用一个0-100的自然数来表示，数越大表示越好心。小渊和小轩希望尽可能找好心程度高的同学来帮忙传纸条，即找到来回两条传递路径，使得这两条路径上同学的好心程度只和最大。现在，请你帮助小渊和小轩找到这样的两条路径。

输入格式

　　输入第一行有2个用空格隔开的整数m和n，表示班里有m行n列（1<=m,n<=50）。  
　　接下来的m行是一个m\*n的矩阵，矩阵中第i行j列的整数表示坐在第i行j列的学生的好心程度。每行的n个整数之间用空格隔开。

输出格式

　　输出一行，包含一个整数，表示来回两条路上参与传递纸条的学生的好心程度之和的最大值。

样例输入

3 3  
0 3 9  
2 8 5  
5 7 0

样例输出

34

数据规模和约定

　　30%的数据满足：1<=*m*,*n*<=10  
　　100%的数据满足：1<=*m*,*n*<=50

本题的C++参考代码如下：

#include<iostream>

#include<string.h>

using namespace std ;

int dis[101][101] ;

int f[105][52][52] ;

int rows , cols , step ;

#define MAX(a,b) ((a) > (b) ? (a) : (b))

int Max( int a , int b , int c , int d){

a = MAX ( a , b ) ;

a = MAX ( a , c ) ;

a = MAX ( a , d ) ;

return a ;

}

int main(){

std::ios::sync\_with\_stdio(false) ;

int i , j , k ;

memset( f , 0 , sizeof(f) ) ;

memset( dis , 0 , sizeof(dis) ) ;

cin >> rows >> cols ;

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

for( j = 1 ; j <= cols ; j++ )

cin >> dis[i][j] ;

step = rows + cols - 2 ;

f[1][2][1] = f[1][1][2]= dis[1][2] + dis[2][1] ;

for( k = 2 ; k <= (step - 1) ; k++ )

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

for( j = 1 ; j <= rows ; j++ )

if( i != j )

f[k][i][j] = dis[i][k - i + 2] + dis[j][k - j + 2] + Max( f[k - 1][i - 1][j - 1] , f[k - 1][i - 1][j] , f[k - 1][i][j - 1] , f[k - 1][i][j]) ;

cout << f[step - 1][rows - 1][rows] << endl ;

return 0 ;

}

本题的C参考代码如下：

#include<stdio.h>

int n,m;

int i,j,k;

int Map[51][51];

int F[111][51][51];

int Max(int a,int b,int c,int d)

{

if(a>=b&&a>=c&&a>=d)

return a;

if(b>=a&&b>=c&&b>=d)

return b;

if(c>=a&&c>=b&&c>=d)

return c;

if(d>=a&&d>=b&&d>=c)

return d;

}

int main()

{

scanf("%d%d",&n,&m);

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

for(j=1;j<=m;j++)

scanf("%d",&Map[i][j]);

for(k=1;k<=n+m-2;k++)

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

for(j=1;j<=n;j++)

if(i==n&&j==n&&k==n+m-2)

F[k][i][j]=Max(F[k-1][i-1][j],F[k-1][i][j-1],F[k-1][i][j],F[k-1][i-1][j-1])+Map[i][k+2-i]+Map[j][k+2-j];

else if(i!=j&&k+2-i>=1&&k+2-j>=1)

F[k][i][j]=Max(F[k-1][i-1][j],F[k-1][i][j-1],F[k-1][i][j],F[k-1][i-1][j-1])+Map[i][k+2-i]+Map[j][k+2-j];

printf("%d",F[n+m-2][n][n]);

return 0;

}

**算法训练 传球游戏**

**【**问题描述】  
　　上体育课的时候，小蛮的老师经常带着同学们一起做游戏。这次，老师带着同学们一起做传球游戏。  
　　游戏规则是这样的：n个同学站成一个圆圈，其中的一个同学手里拿着一个球，当老师吹哨子时开始传球，每个同学可以把球传给自己左右的两个同学中的一个（左右任意），当老师再次吹哨子时，传球停止，此时，拿着球没传出去的那个同学就是败者，要给大家表演一个节目。  
　　聪明的小蛮提出一个有趣的问题：有多少种不同的传球方法可以使得从小蛮手里开始传的球，传了m次以后，又回到小蛮手里。两种传球的方法被视作不同的方法，当且仅当这两种方法中，接到球的同学按接球顺序组成的序列是不同的。比如有3个同学1号、2号、3号，并假设小蛮为1号，球传了3次回到小蛮手里的方式有1->2->3->1和1->3->2->1，共2种。

输入格式

　　共一行，有两个用空格隔开的整数n，m（3<=n<=30，1<=m<=30）。

输出格式

　　t共一行，有一个整数，表示符合题意的方法数。

样例输入

3 3

样例输出

2

数据规模和约定

　　40%的数据满足：3<=n<=30，1<=m<=20  
　　100%的数据满足：3<=n<=30，1<=m<=30

本题的C++参考代码如下：

#include <iostream>

using namespace std;

int n,m;

int fun(int x)

{

if(x<1)

return x+n;

if(x>n) return x-n;

return x;

}

int main()

{ int f[31][31]={0},i,j;

scanf("%d%d",&n,&m);

f[1][2]=f[1][n]=1;

for (i=2;i<=m;i++)

for (j=1;j<=n;j++)

f[i][j]=f[i-1][fun(j-1)]+f[i-1][fun(j+1)];

printf("%d",f[m][1]);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int n,m;

int at1(int x)

{

if(x<1) return x + n;

if(x>n) return x - n;

return x;

}int main()

{

int f[31][31] = {0},i,j;

scanf("%d%d",&n,&m);

f[1][2] = f[1][n] = 1;

for(i = 2;i <= m;i++)

for(j = 1;j <= n;j++)

f[i][j] = f[i - 1][at1(j - 1)] + f[i - 1][at1(j + 1)];

printf("%d",f[m][1]);

return 0;

}

**算法训练 纪念品分组**

问题描述

　　元旦快到了，校学生会让乐乐负责新年晚会的纪念品发放工作。为使得参加晚会的同学所获得的纪念品价值 相对均衡，他要把购来的纪念品根据价格进行分组，但每组最多只能包括两件纪念品，并且每组纪念品的价格之和不能超过一个给定的整数。为了保证在尽量短的时 间内发完所有纪念品，乐乐希望分组的数目最少。  
　　你的任务是写一个程序，找出所有分组方案中分组数最少的一种，输出最少的分组数目。

输入格式

　　输入包含*n*+2行：  
　　第1行包括一个整数*w*，为每组纪念品价格之和的上限。  
　　第2行为一个整数*n*，表示购来的纪念品的总件数。  
　　第3~*n*+2行每行包含一个正整数*pi* (5 <= *pi* <= *w*)，表示所对应纪念品的价格。

输出格式

　　输出仅一行，包含一个整数，即最少的分组数目。

样例输入

100  
9  
90  
20  
20  
30  
50  
60  
70  
80  
90

样例输出

6

数据规模和约定

　　50%的数据满足：1 <= *n* <= 15  
　　100%的数据满足：1 <= *n* <= 30000, 80 <= *w* <= 200

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

#include "ctype.h"

#include "algorithm"

#include "stack"

#include "list"

#include "math.h"

using namespace std;

const int N =30001;

int a[N];

int main()

{

int n;

int w;

cin>>w>>n;

for(int i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

std::sort(a,a+n);

int ans=0;

for(int i=0,j=n-1;i<=j;)

{

for(;i<j;j--)

{

if(a[i]+a[j]<=w)

break;

else ans++;

}

if(i!=j)

{

i++;j--;

ans++;

}

else

{

ans++;

break;

}

}

cout<<ans;

return 0;

}

本题的C参考代码如下：

/\*

算法训练 纪念品分组

问题描述

　　元旦快到了，校学生会让乐乐负责新年晚会的纪念品发放工作。为使得参加晚会的同学所获得的纪念品价值 相对均衡，他要把购来的纪念品根据价格进行分组，但每组最多只能包括两件纪念品，并且每组纪念品的价格之和不能超过一个给定的整数。为了保证在尽量短的时 间内发完所有纪念品，乐乐希望分组的数目最少。

　　你的任务是写一个程序，找出所有分组方案中分组数最少的一种，输出最少的分组数目。

输入格式

　　输入包含n+2行：

　　第1行包括一个整数w，为每组纪念品价格之和的上限。

　　第2行为一个整数n，表示购来的纪念品的总件数。

　　第3~n+2行每行包含一个正整数pi (5 <= pi <= w)，表示所对应纪念品的价格。

输出格式

　　输出仅一行，包含一个整数，即最少的分组数目。

样例输入

100

9

90

20

20

30

50

60

70

80

90

样例输出

6

数据规模和约定

　　50%的数据满足：1 <= n <= 15

　　100%的数据满足：1 <= n <= 30000, 80 <= w <= 200

\*/

#include <stdio.h>

void qsort(int i,int j);

int a[30000];

void qsort(int i,int j){

int x,p,q;

x=a[i]; p=i; q=j;

while (i<j)

{

while ((i<j)&&(a[j]>x))

j--;

if (i<j)

{

a[i]=a[j];

i++;

}

while ((i<j)&&(a[i]<x))

i++;

if (i<j)

{

a[j]=a[i];

j--;

}

}

a[i]=x;

if (p<i-1)

qsort(p,i-1);

if (i+1<q)

qsort(i+1,q);

}

main(){

int n,w,s,i,j;

scanf("%d%d",&w,&n);

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

scanf("%d",&a[i]);

qsort(0,n-1);

i=0;

j=n-1;

s=0;

while (i<j)

{

s++;

if (a[i]+a[j]<=w)

{

i++;

j--;

}

else

j--;

}

if ((i==j)&&(a[i]<=w))

s++;

printf("%d",s);

getchar();

getchar();

return(0);

}

**算法训练 数列**

问题描述

　　给定一个正整数k(3≤k≤15),把所有k的方幂及所有有限个互不相等的k的方幂之和构成一个递增的序列，例如，当k=3时，这个序列是：  
　　1，3，4，9，10，12，13，…  
　　（该序列实际上就是：30，31，30+31，32，30+32，31+32，30+31+32，…）  
　　请你求出这个序列的第N项的值（用10进制数表示）。  
　　例如，对于k=3，N=100，正确答案应该是981。

输入格式

　　只有1行，为2个正整数，用一个空格隔开：  
　　k N  
　　（k、N的含义与上述的问题描述一致，且3≤k≤15，10≤N≤1000）。

输出格式

　　计算结果，是一个正整数（在所有的测试数据中，结果均不超过2.1\*109）。（整数前不要有空格和其他符号）。

样例输入

3 100

样例输出

981

本题的C++参考代码如下：

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

int main()

{

int k, n;

cin>>k>>n;

int a[1010]={0,1,k,1+k,k\*k};

for (int i=5; i<=n; i++)

if (i%2) a[i]=a[i-1]+1; else a[i]=a[2]\*a[i/2];

cout<<a[n];

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int re[1000],k,n;

int p=0,t=1,i,j,f;

re[0]=1;j=0,f=1;

scanf("%d%d",&k,&n);

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

{

if(f==re[j])

{

f\*=k;

re[i]=f;

// printf("%d ",f);

j=0;

continue;

}

t=f+re[j++];

re[i]=t;

// printf("%d ",t);

}

printf("%d",re[i-1]);

return 0;

}

**算法训练 JAM计数法**

问题描述

　　Jam是个喜欢标新立异的科学怪人。他不使用阿拉伯数字计数，而是使用**小写英文字母**计数，他觉得这样做，会使世界更加丰富多彩。在他的计数法中，每个数字的位数都是相同的（使用相同个数的字母），英文字母按原先的顺序，排在前面的字母小于排在它后面的字母。我们把这样的“数字”称为Jam数字。在Jam数字中，每个字母互不相同，而且**从左到右是严格递增**的。每次，Jam还指定使用字母的**范围**，例如，从2到10，表示只能使用{b,c,d,e,f,g,h,i,j}这些字母。如果再规定位数为5，那么，紧接在Jam数字“bdfij”之后的数字应该是“bdghi”。（如果我们用U、V依次表示Jam数字“bdfij”与“bdghi”，则U<V< span>，且不存在Jam数字P，使U<P<V< span>）。你的任务是：对于从文件读入的一个Jam数字，按顺序输出紧接在后面的5个Jam数字，如果后面没有那么多Jam数字，那么有几个就输出几个。

输入格式

　　有2行，第1行为3个正整数，用一个空格隔开：  
　　s t w  
　　（其中s为所使用的最小的字母的序号，t为所使用的最大的字母的序号。w为数字的位数，这3个数满足：1≤s<T≤26, 2≤w≤t-s ）  
　　第2行为具有w个小写字母的字符串，为一个符合要求的Jam数字。  
　　所给的数据都是正确的，不必验证。

输出格式

　　最多为5行，为紧接在输入的Jam数字后面的5个Jam数字，如果后面没有那么多Jam数字，那么有几个就输出几个。每行只输出一个Jam数字，是由w个小写字母组成的字符串，不要有多余的空格。

样例输入

2 10 5  
bdfij

样例输出

bdghi  
bdghj  
bdgij  
bdhij  
befgh

本题的C++参考代码如下：

#include<stdio.h>

int p[26], num;

void output()

{

for(int i = 0; i < num; i++)

printf("%c", p[i]+'a');

printf("\n");

}

int main()

{

int s, end;

scanf("%d%d%d", &s, &end, &num);

getchar();

for(int i = 0; i < num; i++)

{

char c = getchar();

p[i] = c - 'a';

}

int t = 5;

while(t--)

{

int crt = num - 1, tmp = end - 1;

while(crt >= 0 && p[crt--] == tmp--);

if(crt < 0)

return 0;

p[crt] = p[++crt]++;

while(++crt < num)

p[crt] = p[crt-1] + 1;

output();

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int s,t,w;

char in[26];

int ad[26],i,j,k,flag,st;

scanf("%d%d%d",&s,&t,&w);

getchar();

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

{

scanf("%c",&in[i]);

ad[i]=in[i]-'a'+1;

}

// for(i=0;i<w;i++)

// printf("%d ",ad[i]);//+'a'-1);

// printf("\n");

for(i=0;i<5;i++)

{

if(ad[0]==(t-w+1))

break;

// flag=0;

for(j=w-1;j>=0;j--)

{

if(ad[j]==t-w+j+1)

continue;

// if(!flag)

// {

// ad[j]++;

st=++ad[j];

// k=j;

for(k=j+1;k<w;k++)

ad[k]=++st;

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

printf("%c",ad[k]+'a'-1);

printf("\n");

flag=1;

break;

// }

}

}

return 0;

}

**算法训练 开心的金明**

问题描述

　　金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎 么布置，你说了算，只要不超过N元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一 个重要度，分为5等：用整数1~5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N元（可以等于N元）的前提 下，使每件物品的价格与重要度的乘积的总和最大。  
　　设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为 j1，j2，……，jk，则所求的总和为：  
　　v[j1]\*w[j1]+v[j2]\*w[j2]+ …+v[jk]\*w[jk]。（其中\*为乘号）  
　　请 你帮助金明设计一个满足要求的购物单。

输入格式

　　输入文件 的第1行，为两个正整数，用一个空格隔开：  
　　N m  
　　（其中N（<30000）表示总钱 数，m（<25）为希望购买物品的个数。）  
　　从第2行到第m+1行，第j行给出了编号为j-1的物品的基本数据，每行有2个非负整数  
　　v p  
　　（其中v表示该物品的价格(v<=10000)，p表示该物品的重要度(1~5)）

输出格式

　　输出文件只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（<100000000）。

样例输入

1000 5  
800 2  
400 5  
300 5  
400 3  
200 2

样例输出

3900

数据规模和约定

本题的C++参考代码如下：

#include<string.h>

using namespace std;

int dp[30000];

int main()

{

int N,m,v[300],w[300];

//freopen("d://1.txt","r",stdin);

cin>>N>>m;

for(int i=0;i<m;i++)

{

cin>>v[i]>>w[i];

}

fill(dp,dp+N,0);

dp[0]=1;

for(int i=0;i<m;i++)

for(int j=N;j>=v[i];j--)

dp[j]=max(dp[j],dp[j-v[i]]+w[i]\*v[i]);

cout<<dp[N]<<endl;

return 0;

}本题的C参考代码如下：

#include <stdio.h>

long mnum[30000];

long objprice[25]; //第i件商品的价格

long objk[25]; //第i件商品的价格与权值的积

int M,N; //M表示金钱总数，N表示商品总数

int main()

{

int i,m;

int maxp = 0;

scanf("%u%u", &M, &N); //M表示金钱总数，N表示商品总数

for(i=0;i<N;++i) //依次输入第i件商品的价格与权值，并计算出商品的价格与权值之积

{

scanf("%u%u", &objprice[i], &m);

objk[i] = m\*objprice[i];

}

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

{

for(m=0;m<M-objprice[i];++m)

if (mnum[m+objprice[i]] + objk[i] > mnum[m])

mnum[m] = mnum[m+objprice[i]] + objk[i];

}

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

if (mnum[i] > maxp)

maxp = mnum[i];

printf("%u", maxp);

return 0;

}

**算法训练 入学考试**

问题描述

　　辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”  
　　如果你是辰辰，你能完成这个任务吗？

输入格式

　　第一行有两个整数T（1 <= T <= 1000）和M（1 <= M <= 100），用一个空格隔开，T代表总共能够用来采药的时间，M代表山洞里的草药的数目。接下来的M行每行包括两个在1到100之间（包括1和100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

输出格式

　　包括一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值。

样例输入

70 3  
71 100  
69 1  
1 2

样例输出

3

数据规模和约定

　　对于30%的数据，M <= 10；  
　　对于全部的数据，M <= 100。

本题的C++参考代码如下：

#include<stdio.h>

const int maxn = 1010;

int dp[maxn];

int main()

{

int V, M, w[100], v[100], i, j;

scanf("%d%d", &V, &M);

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

{

scanf("%d%d", &v[i], &w[i]);

}

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

for(j = V; j >= v[i]; j--)

dp[j] = dp[j] > dp[j-v[i]] + w[i] ? dp[j] : dp[j-v[i]] + w[i];

printf("%d\n", dp[V]);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int totalTime, medics;

int value[100];

int time[100];

int max[1001];

int main()

{

scanf("%d%d", &totalTime, &medics);

int i, t;

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

{

scanf("%d%d", &time[i], &value[i]);

}

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

{

for (t=totalTime; t>0; t--)

{

if (time[i] <= t)

{

if (value[i] + max[t-time[i]] > max[t])//第i个的价值+不选第i个且用时为t-time[i-1]时最大价值

{

max[t] = value[i] + max[t-time[i]];

}

}

}

}

printf("%d\n", max[totalTime]);

return 0;

}

**算法训练 校门外的树**

问题描述

　　某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数 轴上的每个整数点，即0，1，2，……，L，都种有一棵树。  
　　由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已 知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树 都移走后，马路上还有多少棵树。

输入格式

　　输入文件的第一行有两个整数L（1 <= L <= 10000）和 M（1 <= M <= 100），L代表马路的长度，M代表区域的数目，L和M之间用一个空格隔开。接下来的M行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点 和终止点的坐标。

输出格式

　　输出文件包括一行，这一行只包含一个整数，表示马路上剩余的树的数目。

样例输入

500 3  
150 300  
100 200  
470 471

样例输出

298

数据规模和约定

　　对于20%的数据，区域之间没有重合的部分；  
　　对于其它的数据，区域之间有重合的情况。

本题的C++参考代码如下：

#include<stdio.h>

#include<algorithm>

using namespace std;

struct Area

{

int l, r;

};

Area a[101];

bool cmp(Area a, Area b)

{

return a.l < b.l;

}

int main()

{

int L, m, i;

scanf("%d%d", &L, &m);

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

scanf("%d%d", &a[i].l, &a[i].r);

sort(a, a+m, cmp);

int tot = 0, crt = -1;

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

{

if(a[i].l > crt)

{

crt = a[i].l;

tot++;

}

if(crt < a[i].r)

{

tot += a[i].r - crt;

crt = a[i].r;

}

}

printf("%d\n", L-tot+1);

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

typedef struct

{

int start;

int end;

int flag;

}extent;

int main()

{

int L,M,i,j;

extent e[101];

scanf("%d%d",&L,&M);

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

{

scanf("%d%d",&(e[i].start),&(e[i].end));

e[i].flag=1;

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

{

if(!(e[i].end<e[j].start||e[i].start>e[j].end)&&e[j].flag)

{

e[j].flag=0;

if(e[i].start>e[j].start)

e[i].start=e[j].start;

if(e[i].end<e[j].end)

e[i].end=e[j].end;

}

}//调整区间

}

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

if(e[i].flag)

L=L-(e[i].end-e[i].start+1);

printf("%d",L+1);

return 0;

}

**算法训练 星际交流**

问题描述

　　人类终于登上了火星的土地并且见到了神秘的火星人。人类和火星人都无法理解对方的语言，但是我们的科学家发明了一种用数字交流的方法。这种交流方法是这样 的，首先，火星人把一个非常大的数字告诉人类科学家，科学家破解这个数字的含义后，再把一个很小的数字加到这个大数上面，把结果告诉火星人，作为人类的回 答。  
　　火星人用一种非常简单的方式来表示数字——掰手指。火星人只有一只手，但这只手上有成千上万的手指，这些手指排成一列，分别编号为1，2，3……。火星人的任意两根手指都能随意交换位置，他们就是通过这方法计数的。  
　　一个火星人用一个人类的手演示了如何用手指计数。如果把五根手指——拇指、食指、中指、无名指和小指分别编号为1，2，3，4和5，当它们按正常顺序排列 时，形成了5位数12345，当你交换无名指和小指的位置时，会形成5位数12354，当你把五个手指的顺序完全颠倒时，会形成54321，在所有能够形 成的120个5位数中，12345最小，它表示1；12354第二小，它表示2；54321最大，它表示120。下表展示了只有3根手指时能够形成的6个 3位数和它们代表的数字：  
　　三进制数  
　　123  
　　132  
　　213  
　　231  
　　312  
　　321  
　　代表的数字  
　　1  
　　2  
　　3  
　　4  
　　5  
　　6  
　　现在你有幸成为了第一个和火星人交流的地球人。一个火星人会让你看他的手指，科学家会告诉你要加上去的很小的数。你的任务是，把火星人用手指表示的数与科 学家告诉你的数相加，并根据相加的结果改变火星人手指的排列顺序。输入数据保证这个结果不会超出火星人手指能表示的范围。

输入格式

　　包括三行，第一行有一个正整数N，表示火星人手指的数目（1 <= N <= 10000）。第二行是一个正整数M，表示要加上去的小整数（1 <= M <= 100）。下一行是1到N这N个整数的一个排列，用空格隔开，表示火星人手指的排列顺序。

输出格式

　　只有一行，这一行含有N个整数，表示改变后的火星人手指的排列顺序。每两个相邻的数中间用一个空格分开，不能有多余的空格。

样例输入

5  
3  
1 2 3 4 5

样例输出

1 2 4 5 3

数据规模和约定

　　对于30%的数据，N<=15；  
　　对于60%的数据，N<=50；  
　　对于全部的数据，N<=10000；

本题的C++参考代码如下：

#include <iostream>

#include <cmath>

#include <cstring>

#include <algorithm>

#include <cstdio>

using namespace std;

#define du freopen("in.txt","r",stdin)

#define chu freopen("out.txt","w",stdout)

#define FOR(i,a,b) for(int i=a;i<=b;i++)

#define FD(i,a,b) for(int i=a;i>=b;i--)

#define FF(i,a) for(int i=0;i<a;i++)

#define SD(x) scanf("%d",&x)

#define PD(x) printf("%d",x)

#define PP printf(" ")

#define LN puts("")

const int maxn= 0;

int a[20001];

int main()

{

int n;

while(~scanf("%d",&n)){

int m; SD(m);

FF(i,n) SD(a[i]);

while( m&& next\_permutation(a,a+n)){

m--;

}

PD(a[0]);

FOR(i,1,n-1){

PP;PD(a[i]);

}

LN;

}

}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

int n,w,i1,i2;

int a[100000];

scanf("%d%d",&n,&w);

for(i1=0;i1<n;i1++)

scanf("%d",&a[i1]);

while(w--)

{ int max=a[n-1],min;

for(i1=n-2;i1>=0;i1--)

{

if(max<a[i1])

{max=a[i1];

continue;

}

min=max;

int xiabiao=-1;

for(i2=i1+1;i2<=n-1;i2++)

{

if(a[i2]>a[i1])

{

if(min>=a[i2])

{

min=a[i2];

xiabiao=i2;

}

}

}

int k=a[i1];

a[i1]=min;

a[xiabiao]=k;

for(i2=1;i1+i2<n-i2;i2++)

{

int kp=a[i2+i1];

a[i2+i1]=a[n-i2];

a[n-i2]=kp;

}

break;

}

}

for(i1=0;i1<n;i1++)

printf("%d ",a[i1]);

printf("\n");

return 0;

}

**算法训练 FBI树**

问题描述

　　我们可以把由“0”和“1”组成的字符串分为三类：全“0”串称为B串，全“1”串称为I串，既含“0”又含“1”的串则称为F串。  
　　FBI树是一种二叉树，它的结点类型也包括F结点，B结点和I结点三种。由一个长度为2N的“01”串S可以构造出一棵FBI树T，递归的构造方法如下：  
　　1)T的根结点为R，其类型与串S的类型相同；  
　　2)若串S的长度大于1，将串S从中间分开，分为等长的左右子串S1和S2；由左子串S1构造R的左子树T1，由右子串S2构造R的右子树T2。  
　　现在给定一个长度为2N的“01”串，请用上述构造方法构造出一棵FBI树，并输出它的后序遍历序列。

输入格式

　　第一行是一个整数N（0 <= N <= 10），第二行是一个长度为2N的“01”串。

输出格式

　　包括一行，这一行只包含一个字符串，即FBI树的后序遍历序列。

样例输入

3  
10001011

样例输出

IBFBBBFIBFIIIFF

数据规模和约定

　　对于40%的数据，N <= 2；  
　　对于全部的数据，N <= 10。  
　　注：  
　　[1] 二叉树：二叉树是结点的有限集合，这个集合或为空集，或由一个根结点和两棵不相交的二叉树组成。这两棵不相交的二叉树分别称为这个根结点的左子树和右子树。  
　　[2] 后序遍历：后序遍历是深度优先遍历二叉树的一种方法，它的递归定义是：先后序遍历左子树，再后序遍历右子树，最后访问根。

本题的C++参考代码如下：

#include<cstdio>

#include<cstdlib>

#include<cstring>

int a[2100][3],f[1100],v=1,s=1;

void g(int x,int y)

{

if(y-x+1>1)

{

int vv=v,v1,v2;

a[vv][0]=v+1;v++;v1=v;

g(x,x+(y-x)/2);

a[vv][1]=v+1;v++;v2=v;

g(x+(y-x)/2+1,y);

if(a[v1][2]==a[v2][2])a[vv][2]=a[v1][2];

else a[vv][2]=2;

}

else{a[v][0]=-1;a[v][1]=-1;a[v][2]=f[s];s++;}

}

void h(int x)

{

if(a[x][0]!=-1)h(a[x][0]);

if(a[x][1]!=-1)h(a[x][1]);

if(a[x][2]==0)printf("B");

if(a[x][2]==1)printf("I");

if(a[x][2]==2)printf("F");

}

int main()

{

int n=1,i,nn;

char c;

scanf("%d\n",&nn);

for(i=1;i<=nn;i++)n\*=2;

for(i=1;i<=n;i++){scanf("%c",&c);f[i]=c-'0';}

g(1,n);

h(1);

printf("\n");

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <malloc.h>

char in[1025];

typedef struct node

{

char data;

struct node \*lchild,\*rchild;

}Binode;

Binode \*create(int a,int b)

{

char ch;

Binode \*p;

int i,count0=0,count1=0,flag=0;

// scanf("%c",&ch);

// getchar();

if(a==b)

{

(in[a]=='0')?(ch='B'):(ch='I');

p=(Binode \*)malloc(sizeof(Binode));

p->data=ch;

p->lchild=NULL;

p->rchild=NULL;

return p;

}

for(i=a;i<=b;i++)

{

(in[i]=='0')?(count0++):(count1++);

if(count0 && count1)

{

ch='F';

flag=1;

break;

}

}

if(!flag)

{

if(!count0)

ch='I';

else

ch='B';

}

p=(Binode \*)malloc(sizeof(Binode));

p->data=ch;

p->lchild=create(a,((b+1)-a)/2+a-1);

p->rchild=create(((b+1)-a)/2+a,b);

return p;

}

void display(Binode \*p)

{

if(!p)

return;

display(p->lchild);

display(p->rchild);

printf("%c",p->data);

}

int main()

{

Binode \*r;

char c;

int p=0,n;

scanf("%d",&n);

getchar();

while(1)

{

c=getchar();

if(c=='\n')

break;

in[p++]=c;

}

r=create(0,p-1);

display(r);

return 0;

}

**算法训练 麦森数**

问题描述

　　形如2*P*-1的素数称为麦森数，这时P一定也是个素数。但反过来不一定，即如果P是个素数，2*P*-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是P=3021377，它有909526位。麦森数有许多重要应用，它与完全数密切相关。  
　　任务：从文件中输入P（1000<P<3100000），计算2*P*-1的位数和最后500位数字（用十进制高精度数表示）

输入格式

　　文件中只包含一个整数P（1000<P<3100000）

输出格式

　　第一行：十进制高精度数2*P*-1的位数。  
　　第2-11行：十进制高精度数2*P*-1的最后500位数字。（每行输出50位，共输出10行，不足500位时高位补0）  
　　不必验证2*P*-1与P是否为素数。

样例输入

1279

样例输出

386  
00000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000  
00000000000000104079321946643990819252403273640855  
38615262247266704805319112350403608059673360298012  
23944173232418484242161395428100779138356624832346  
49081399066056773207629241295093892203457731833496  
61583550472959420547689811211693677147548478866962  
50138443826029173234888531116082853841658502825560  
46662248318909188018470682222031405210266984354887  
32958028878050869736186900714720710555703168729087

本题的C++参考代码如下：

#include<iostream>

#include<vector>

#include<cmath>

#include<iomanip>

using std::cin;

using std::cout;

using std::endl;

using std::vector;

using std::setw;

using std::setfill;

vector<int> Multiply(vector<int> ivec1,vector<int> ivec2)

{

vector<int> ivec(125,0);

for(int i=0;i!=125;++i)

{

int n=0;

for(int j=0;j!=125-i;++j)

{

int tmp=ivec[i+j]+ivec1[j]\*ivec2[i]+n;

ivec[i+j]=tmp%10000;

n=tmp/10000;

}

}

return ivec;

}

int main()

{

int p=0;

cin>>p;

cout<<(int)(p\*log10(2.0))+1<<endl;

vector<int> Boolvec;

while(p)

{

Boolvec.push\_back(p%2);

p/=2;

}

vector<int> ivec;

vector<int> Result;

ivec.push\_back(2);

Result.push\_back(1);

for(int i=1;i!=125;++i)

{

ivec.push\_back(0);

Result.push\_back(0);

}

for(int i=0;i<(int)Boolvec.size();++i)

{

if(Boolvec[i])

Result=Multiply(Result,ivec);

ivec=Multiply(ivec,ivec);

}

Result[0]--;

for(int i=124;i>=0;--i)

{

if(i%25==12)

{

cout<<setfill('0')<<setw(2)<<Result[i]/100;

cout<<endl;

cout<<setfill('0')<<setw(2)<<Result[i]%100;

}

else

{

if(i%25==0)

{

cout<<setfill('0')<<setw(4)<<Result[i];

cout<<endl;

}

else

cout<<setfill('0')<<setw(4)<<Result[i];

}

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

#include <math.h>

#define LEN 125 //每数组元素存放十进制的4位，因此数组最多只要125个元素即可

//Multiply函数功能是计算高精度乘法a\*b，结果的末500位放在a中

void Multiply(int \*a,int \*b)

{

int i,j;

int nCarry; //存放进位

int nTmp;

int c[LEN]; //存放结果的末500位

memset(c,0,sizeof(int)\*LEN);

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

{

nCarry=0;

for(j=0;j<LEN-i;j++)

{

nTmp=c[i+j]+a[j]\*b[i]+nCarry;

c[i+j]=nTmp%10000;

nCarry=nTmp/10000;

}

}

memcpy(a,c,LEN\*sizeof(int));

}

int main()

{

int i,p;

int anPow[LEN]; //存放不断增长的2的次幂

int aResult[LEN]; //存放最终结果的末500位

scanf("%d",&p);

printf("%d\n",(int)(p\*log10(2))+1);

//下面将2的次幂初始化为2^(2^0)(a^b表示a的b次方),最终结果初始化为1

anPow[0]=2;

aResult[0]=1;

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

{

anPow[i]=0;

aResult[i]=0;

}

//下面计算2的p次方

while(p>0) //p=0则说明p中的有效位都用过了，不需要再计算下去

{

if(p&1) //判断此时p中最低位是否为1

Multiply(aResult,anPow);

p>>=1;

Multiply(anPow,anPow);

}

aResult[0]--; //2的p次方算出后减1

//输出结果

for(i=LEN-1;i>=0;i--)

{

if(i%25==12)

printf("%02d\n%02d",aResult[i]/100,aResult[i]%100);

else

{

printf("%04d",aResult[i]);

if(i%25==0)

printf("\n");

}

}

return 0;

}

**算法训练 Car的旅行路线**

问题描述

　　又到暑假了，住在城市A的Car想和朋友一起去城市B旅游。她知道每个城市都有四个飞机场，分别位于一个矩形的四个顶点上，同一个城市中两个机场之间有一 条笔直的高速铁路，第I个城市中高速铁路了的单位里程价格为Ti，任意两个不同城市的机场之间均有航线，所有航线单位里程的价格均为t。  
　　那么Car应如何安排到城市B的路线才能尽可能的节省花费呢?她发现这并不是一个简单的问题，于是她来向你请教。  
　　找出一条从城市A到B的旅游路线，出发和到达城市中的机场可以任意选取，要求总的花费最少。

输入格式

　　的第一行有四个正整数s，t，A，B。  
　　S表示城市的个数，t表示飞机单位里程的价格，A，B分别为城市A，B的序号，(1<=A，B<=S)。  
　　接下来有S行，其中第I行均有7个正整数xi1，yi1，xi2，yi2，xi3，yi3，Ti，这当中的(xi1，yi1)，(xi2，yi2)，(xi3，yi3)分别是第I个城市中任意三个机场的坐标，T I为第I个城市高速铁路单位里程的价格。

输出格式

　　共有n行，每行一个数据对应测试数据，保留一位小数。

样例输入

1  
1 10 1 3  
1 1 1 3 3 1 30  
2 5 7 4 5 2 1  
8 6 8 8 11 6 3

样例输出

47.55

数据规模和约定

　　0<S<=100，

本题的C++参考代码如下：

#include <float.h>

#include <math.h>

#include <stdio.h>

#include<cstring>

#define Sqr(x) ((x)\*(x))

double s[400][400], x[400], y[400];

int main()

{

int a, b, n, tt,t,i,j,k,N;

double res = DBL\_MAX;

scanf("%d%d%d%d", &n, &tt, &a, &b);

a-=1;

b-=1;

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

{

for ( j = 0; j < 3; ++j)

scanf("%lf%lf", &x[i\*4+j], &y[i\*4+j]);

scanf("%d", &t);

for (j = 0; j < 3; ++j)

{

double x1 = x[i\*4+(j+1)%3], y1 = y[i\*4+(j+1)%3],

x2 = x[i\*4+(j+2)%3], y2 = y[i\*4+(j+2)%3];

if (fabs((x[i\*4+j]-x1)\*(x[i\*4+j]-x2)+(y[i\*4+j]-y1)\*(y[i\*4+j]-y2)) < 1E-7)

x[i\*4+3] = x1+x2-x[i\*4+j], y[i\*4+3] = y1+y2-y[i\*4+j];

}

for ( j = 0; j < 4; ++j)

for (k = 0; k < 4; ++k)

s[i\*4+j][i\*4+k] = t\*sqrt(Sqr(x[i\*4+j]-x[i\*4+k])+Sqr(y[i\*4+j]-y[i\*4+k]));

for ( j = 0; j < i\*4; ++j)

for ( k = 0; k < 4; ++k)

s[i\*4+k][j] = tt\*sqrt(Sqr(x[i\*4+k]-x[j])+Sqr(y[i\*4+k]-y[j])),

s[j][i\*4+k] = s[i\*4+k][j];

}

for ( k = 0; k < n\*4; ++k)

for (i = 0; i < n\*4; ++i)

for ( j = 0; j < n\*4; ++j)

if (s[i][k]+s[k][j] < s[i][j])

s[i][j] = s[i][k]+s[k][j];

for (i = 0; i < 4; ++i)

for ( j = 0; j < 4; ++j)

if (s[a\*4+i][b\*4+j] < res)

res = s[a\*4+i][b\*4+j];

printf("%.1lf\n", res);

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<string.h>

#define BIG 999999.0

void Dijkstra(void);

float dist(float x1,float y1,float x2,float y2);

void construct(void);

typedef struct

{

float x,y;

}Point;

Point ct[100][4];

int rail[100];

int s,t,a,b;

float dis[100];

float g[400][400];

int main()

{

int i,j,k;

float answer=BIG;

scanf("%d%d%d%d",&s,&t,&a,&b);

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

scanf("%f %f %f %f %f %f %d ",&ct[i][0].x,&ct[i][0].y,&ct[i][1].x,&ct[i][1].y,

&ct[i][2].x,&ct[i][2].y,&rail[i]);

construct();//构图

Dijkstra();

for (i=0;i<4;i++){

if(dis[(b-1)\*4+i]<answer)

answer=dis[(b-1)\*4+i];

}

printf("%.1f ",answer);

system("pause");

return 0;

}

void construct(void)

{

int i,j,k,l;

for (i=0;i<s;i++){ //计算第四个点的坐标

if((ct[i][1].x-ct[i][0].x)\*(ct[i][2].x-ct[i][0].x)+

(ct[i][1].y-ct[i][0].y)\*(ct[i][2].y-ct[i][0].y)==0){

ct[i][3].x=ct[i][2].x+ct[i][1].x-ct[i][0].x;

ct[i][3].y=ct[i][2].y+ct[i][1].y-ct[i][0].y;

}else if((ct[i][0].x-ct[i][1].x)\*(ct[i][2].x-ct[i][1].x)

+(ct[i][0].y-ct[i][1].y)\*(ct[i][2].y-ct[i][1].y)==0){

ct[i][3].x=ct[i][2].x+ct[i][0].x-ct[i][1].x;

ct[i][3].y=ct[i][2].y+ct[i][0].y-ct[i][1].y;

}else if((ct[i][0].x-ct[i][2].x)\*(ct[i][1].x-ct[i][2].x)

+(ct[i][0].y-ct[i][2].y)\*(ct[i][1].y-ct[i][2].y)==0){

ct[i][3].x=ct[i][1].x+ct[i][0].x-ct[i][2].x;

ct[i][3].y=ct[i][1].y+ct[i][0].y-ct[i][2].y;

}

}

for(i=0;i<s;i++){//以城市为单位构图

for(j=0;j<4;j++)//城市内部

for(k=0;k<4;k++)

g[4\*i+j][4\*i+k]=rail[i]\*dist(ct[i][j].x,ct[i][j].y,ct[i][k].x,ct[i][k].y);

for(j=0;j<s;j++){//城市之间

if(j==i)continue;

for(k=0;k<4;k++)

for(l=0;l<4;l++)

g[4\*i+k][4\*j+l]=t\*dist(ct[i][k].x,ct[i][k].y,ct[j][l].x,ct[j][l].y);

}

}

}

float dist(float x1,float y1,float x2,float y2)

{

return sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

}

void Dijkstra()

{

int i,j;

int min=BIG,minn;

int vis[100];

memset(vis,0,sizeof(vis));

for(i=0;i<s\*4;i++)

dis[i]=BIG;

dis[a-1]=dis[a]=dis[a+1]=dis[a+2]=0;

for(i=0;i<s\*4;i++){

min=BIG;

for(j=0;j<s\*4;j++){

if(!vis[j] && dis[j]<min){

min=dis[j];

minn=j;

}

}

vis[minn]=1;

for(j=0;j<4\*s;j++){

if(min+g[minn][j]<dis[j])

dis[j]=dis[minn]+g[minn][j];

}

}

}

**算法训练 统计单词个数**

问题描述

　　给出一个长度不超过200的由小写英文字母组成的字母串(约定;该字串以每行20个字母的方式输入，且保证每行一定为20个)。要求将此字母串分成k份 (1<k<=40)，且每份中包含的单词个数加起来总数最大(每份中包含的单词可以部分重叠。当选用一个单词之后，其第一个字母不能再用。例 如字符串this中可包含this和is，选用this之后就不能包含th)。  
　　单词在给出的一个不超过6个单词的字典中。  
　　要求输出最大的个数。

输入格式

　　第一行有二个正整数(p，k)  
　　p表示字串的行数;  
　　k表示分为k个部分。  
　　接下来的p行，每行均有20个字符。  
　　再接下来有一个正整数s，表示字典中单词个数。(1<=s<=6)  
　　接下来的s行，每行均有一个单词。

输出格式

　　每行一个整数，分别对应每组测试数据的相应结果。

样例输入

1 3  
thisisabookyouareaoh  
4  
is  
a  
ok  
sab

样例输出

7

数据规模和约定

　　长度不超过200，1<k<=40，字典中的单词数不超过6。

本题的C++参考代码如下：

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

//ifstream fin("input.txt");

//#define cin fin

int p, k;

int s;

string inputString;

vector<string> words;

void getInput();

int count();

int main()

{

getInput();

int n = count();

cout << n;

return 0;

}

void getInput()

{

// 处理输入

cin >> p >> k;

string temp;

for (int i = 0; i < p; i++)

{

cin >> temp;

inputString += temp;

}

cin >> s;

for (int i = 0; i < s; i++)

{

cin >> temp;

words.push\_back(temp);

}

}

int count()

{

vector<int> Sets;

int dividedNum = 0;

int counts = 0;

int begin = 0;

int end = -1;

for (int i = 0; i < inputString.size(); i++)

{

begin = i;

bool isMatching = false;

for (int j = 0; j < words.size(); j++)

{

size\_t wordSize = words[j].size();

if (inputString.substr(i, wordSize) == words[j])

{

isMatching = true;

counts++;

if (begin <= end)

{

\*Sets.rbegin() = \*Sets.rbegin() + 1;

if (begin + wordSize - 1 > end)

end = begin + wordSize - 1;

}

else

{

Sets.push\_back(0);

end = begin + wordSize - 1;

}

}

if (isMatching == true)

break;

}

if (isMatching == false && i >= end)

dividedNum++;

}

if (k < dividedNum)

return counts;

else

return counts - (k - dividedNum);

}

本题的C参考代码如下：

#include <stdio.h>

#define INF\_MIN 1<<31

#define MAXK 41

#define STR\_SIZE 201

#define WORD\_SIZE 16

int p, k, s;

char str[STR\_SIZE];

char word[6][WORD\_SIZE];

int Cnt\_val[STR\_SIZE][STR\_SIZE]={0};

int Cnt(int from, int end)

{

int i, j, k;

int count = 0;

if(Cnt\_val[from][end] <= 0){

for(i=from; i<=end; i++)

for(j=0; j<s; j++){

if(str[i] == word[j][0]){

for(k=1; word[j][k] != '\0' && i+k<=end; k++){

if(word[j][k] != str[i+k]){

break;

}

}

if(word[j][k] == '\0'){

count ++;

break;

}

}

}

Cnt\_val[from][end] = count;

}

return Cnt\_val[from][end];

}

int main()

{

int i, j, u;

int count\_p;

char ch;

int f[STR\_SIZE][MAXK];

scanf("%d%d", &p, &k);

getchar();

count\_p = 0;

for(i=0; i<STR\_SIZE; i++){

ch = getchar();

if(ch == '\n'){

count\_p ++;

if(count\_p == p){

str[i] = '\0';

break;

}

i --;

}else{

str[i] = ch;

}

}

scanf("%d", &s);

for(i=0; i<s; i++){

scanf("%s", word[i]);

}

int temp, max;

int strlen = 20\*p;

for(i=0; i<strlen; i++){

f[i][1] = Cnt(0, i);

}

for(j=2; j<=k; j++)

{

max = INF\_MIN;

for(i=k-1; i<strlen; i++){

for(u=j-2; u<i; u++){

temp = f[u][j-1] + Cnt(u+1, i);

max = temp>max?temp:max;

}

f[i][j] = max;

}

}

printf("%d", f[strlen-1][k]);

return 0;

}

**算法训练 一元三次方程求解**

问题描述

　　有形如：ax3+bx2+cx+d=0 这样的一个一元三次方程。给出该方程中各项的系数(a，b，c，d 均为实数)，并约定该方程存在三个不同实根(根的范围在-100至100之间)，且根与根之差的绝对值>=1。要求三个实根。。

输入格式

　　四个实数：a，b，c，d

输出格式

　　由小到大依次在同一行输出这三个实根(根与根之间留有空格)，并精确到小数点后2位

样例输入

1 -5 -4 20

样例输出

-2.00 2.00 5.00

数据规模和约定

　　|a|，|b|，|c|，|d|<=10

本题的C++参考代码如下：

#include<iostream>

#include<cstdio>

#include<cmath>

using namespace std;

int main()

{

double a, b, c, d;

scanf("%lf%lf%lf%lf",&a,&b,&c,&d);

for (double x= -100.00; x <= 100.00; x+= 0.001)

{

if (fabs(a\*x\*x\*x + b\*x\*x + c\*x + d) <= 0.01)

{printf("%.2lf ",x); x+=1;}

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

float a,b,c,d,x=-100,t,y;

float f(float z)

{

return a\*z\*z\*z+b\*z\*z+c\*z+d;

}

int main()

{

scanf("%f%f%f%f",&a,&b,&c,&d);

t=f(x);

while(x<=100)

{

y=f(x);

if(y\*t<=0)printf("%.2f ",x);

x=x+0.001;t=y;

}

printf("\n");

return 0;

}

**算法训练 数的划分**

问题描述

　　将整数n分成k份，且每份不能为空，任意两份不能相同(不考虑顺序)。  
　　例如：n=7，k=3，下面三种分法被认为是相同的。  
　　1，1，5; 1，5，1; 5，1，1;  
　　问有多少种不同的分法。

输入格式

　　n，k

输出格式

　　一个整数，即不同的分法

样例输入

7 3

样例输出

4 {四种分法为：1，1，5;1，2，4;1，3，3;2，2，3;}

数据规模和约定

　　6<n<=200，2<=k<=6

本题的C++参考代码如下：

#include<iostream>

#include<cstring>

using namespace std;

int main()

{

int n,k,f[7][201];

memset(f,0,sizeof(f));

cin>>n>>k;

for(int i=0;i<=n;i++)

f[1][i]=1;

for(int i=2;i<=k;i++)

{

for(int j=0;j<=n-k;j++)

{

if(j>=i)

f[i][j]=f[i-1][j]+f[i][j-i];

else

f[i][j]=f[i-1][j];

}

}

cout<<f[k][n-k]<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int i,j,n,k,a[201][7]={0};

a[1][1] = 1;

scanf("%d%d",&n,&k);

for (i = 2; i <= n; i++)

for (j = 1; j <= k; j++)

if (i >= j)

a[i][j] = a[i - j][j] + a[i - 1][j - 1];

printf("%d",a[n][k]);

return 0;

}

**算法训练 装箱问题**

**问题描述**  
　　有一个箱子容量为V（正整数，0＜＝V＜＝20000），同时有n个物品（0＜n＜＝30），每个物品有一个体积（正整数）。  
　　要求n个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

输入格式

　　第一行为一个整数，表示箱子容量；  
　　第二行为一个整数，表示有n个物品；  
　　接下来n行，每行一个整数表示这n个物品的各自体积。

输出格式

　　一个整数，表示箱子剩余空间。  
　　**样例**输入  
　　24  
　　6  
　　8  
　　3  
　　12  
　　7  
　　9  
　　7

样例输出

0

本题的C++参考代码如下：

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

using namespace std;

bool v[21000];

int main()

{

//memset(v,0,sizeof(v));

int i,j,m,n,a;

scanf("%d%d",&m,&n);

v[0]=1;

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

{

scanf("%d",&a);

for(j=m;j>=a;j--)

v[j]=v[j]||v[j-a];

}

for(j=m;j>=0;j--)

{

if(v[j])

{

printf("%d\n",m-j);

return 0;

}

}

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

#include <malloc.h>

#define V 20001

#define N 30

int main()

{

int f[V];

int i,j;

int n,v;

int T[V];

memset(f,0,sizeof(f));

memset(T,0,sizeof(T));

scanf("%d%d",&n,&v);

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

scanf("%d",&f[i]);

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

{

for(j=n;j>=f[i];j--)

{

if(f[i]<=j&&f[i]+T[j-f[i]]>T[j])

T[j] = f[i]+T[j-f[i]];

}

}

printf("%d\n",n-T[n]);

return 0;

}

**算法训练 求先序排列**

**问题描述**  
　　给出一棵二叉树的中序与后序排列。求出它的先序排列。（约定树结点用不同的大写字母表示，长度<=8）。

输入格式

　　两行，每行一个字符串，分别表示中序和后序排列

输出格式

　　一个字符串，表示所求先序排列  
  
　　**样例**输入  
　　BADC  
　　BDCA

样例输出

ABCD

本题的C++参考代码如下：

#include<cstdio>

#include<cstring>

char m[10000];

char h[10000];

void dfs(int ai,int aj,int bi,int bj)

{

int i;

{

if(aj<ai||bj<bi)

return ;

}

if(ai==aj)

printf("%c",m[ai]);

else

{

printf("%c",h[bj]);

for(i=ai;i<=aj;i++)

{

if(m[i]==h[bj])

{

break;

}

}

dfs(ai,i-1,bi,bi+(i-ai-1));

dfs(i+1,aj,bi+i-ai,bj-1);

}

}

int main()

{

gets(m);

gets(h);

dfs(0,strlen(m)-1,0,strlen(m)-1);

puts("");

return 0;

}

本题的C参考代码如下：

#include"stdio.h"

#include"string.h"

void dg(char z[],char h[])

{

if(strlen(h)==0)

return;

printf("%c",h[strlen(h)-1]);

if(strlen(h)==1)

return;

if(strlen(h)==2)

{

printf("%c",h[0]);

return;

}

char a[9],b[9];

int i,j;

for(i=0;z[i]!=h[strlen(h)-1];i++)

a[i]=z[i],b[i]=h[i];

a[i]='\0';

b[i++]='\0';

dg(a,b);

for(j=0;i<strlen(h);i++,j++)

a[j]=z[i],b[j]=h[i-1];

a[j]='\0';

b[j]='\0';

dg(a,b);

}

int main()

{

char h[9],z[9];

scanf("%s",z);

scanf("%s",h);

dg(z,h);

return 0;

}

**算法训练 方格取数**

**问题描述**  
　　设有N\*N的方格图(N<=10),我们将其中的某些方格中填入正整数,而其他的方格中则放入数字0。  
　　某人从图的左上角的A 点(1,1)出发，可以向下行走，也可以向右走，直到到达右下角的B点(N,N)。在走过的路上，他可以取走方格中的数（取走后的方格中将变为数字0）。  
　　此人从A点到B 点共走两次，试找出2条这样的路径，使得取得的数之和为最大。  
**输入格式**  
　　输入的第一行为一个整数N（表示N\*N的方格图），接下来的每行有三个整数，前两个表示位置，第三个数为该位置上所放的数。一行单独的0表示输入结束。  
**输出格式**  
　　只需输出一个整数，表示2条路径上取得的最大的和。  
**样例输入**  
　　8  
　　2 3 13  
　　2 6 6  
　　3 5 7  
　　4 4 14  
　　5 2 21  
　　5 6 4  
　　6 3 15  
　　7 2 14  
　　0 0 0  
**样例输出**  
　　67

本题的C++参考代码如下：

#include<stdio.h>

const int maxn = 10;

int map[maxn][maxn], dp[maxn][maxn][maxn][maxn];

#define max(a, b) (a > b ? a : b)

int main()

{

int n, x, y, w;

scanf("%d", &n);

while(scanf("%d%d%d", &x, &y, &w) == 3 && x && y && w)

{

map[x][y] = w;

}

for(int i = 1; i <= n; i++)

{

for(int j = 1; j <= n; j++)

{

for(int k = 1; k <= n; k++)

{

for(int l = 1; l <= n; l++)

{

int & tmp = dp[i][j][k][l];

int p = max(max(dp[i-1][j][k-1][l], dp[i-1][j][k][l-1]), max(dp[i][j-1][k-1][l], dp[i][j-1][k][l-1]));

tmp += (i == k && j == l) ? (p + map[i][j]) : (p + map[i][j] + map[k][l]);

}

}

}

}

printf("%d\n", dp[n][n][n][n]);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define max(a,b) a>b?a:b

#define min(a,b) a<b?a:b

int main()

{

int map[11][11]={0},f[11][11]={0};

int i,j,k,N,t;

scanf("%d",&N);

while (scanf("%d%d%d",&i,&j,&k)&&(i||j||k)) map[i][j]=k;

for (i=2;i<=2\*N;i++)

for (t=min(i,N),j=t;j>0;j--)

for (k=t;k>0;k--)

{

f[j][k]=max(f[j][k],f[j-1][k-1]);

f[j][k]=max(f[j][k],f[j][k-1]);

f[j][k]=max(f[j][k],f[j-1][k]);

if (j==k) f[j][k]+=map[j][i-j];

else f[j][k]+=map[j][i-j]+map[k][i-k];

}

printf("%d",f[N][N]);

return 0;

}

**算法训练 单词接龙**

**问题描述**  
  
　　单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙”中出现两次），在两个单词相连时，其重合部分合为一部分，例如 beast和astonish，如果接成一条龙则变为beastonish，另外相邻的两部分不能存在包含关系，例如at 和 atide 间不能相连。  
  
**输入格式**  
  
　　输入的第一行为一个单独的整数n (n<=20)表示单词数，以下n 行每行有一个单词，输入的最后一行为一个单个字符，表示“龙”开头的字母。你可以假定以此字母开头的“龙”一定存在.  
  
**输出格式**  
  
　　只需输出以此字母开头的最长的“龙”的长度  
  
**样例输入**  
　　5  
　　at  
　　touch  
　　cheat  
　　choose  
　　tact  
　　a

样例输出

23

样例说明

　　连成的“龙”为atoucheatactactouchoose

本题的C++参考代码如下：

#include <cstdio>

#include <cmath>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

using namespace std ;

int map[25][25];

int n,cnt;

char c;

char st[25][25];

int visit[25];

int dfs(int s)

{

int temp,maxlen=0,t=0;

for(int i=0;i<n;i++)

{

temp=0;

if(map[s][i]&&visit[i])

{

t=1;

visit[i]--;

temp=dfs(i);

visit[i]++;

temp=temp+strlen(st[s])-map[s][i];

}

if(temp>maxlen) maxlen=temp;

}

if(!t) return strlen(st[s]);

return maxlen;

}

int main()

{

while(scanf("%d",&n)!=EOF)

{

for(int i=0;i<n;i++)

scanf("%s",st[i]);

getchar();

scanf("%c",&c);

memset(map,0,sizeof(map));

int len1,len2;

for(int i=0;i<n;i++)

for(int j=0;j<n;j++)

{

len1=strlen(st[i]);

len2=strlen(st[j]);

for(int k=0;k<len1&&k<len2;k++)

{

if(strncmp(st[i]+len1-k-1,st[j],k+1)==0)

{

map[i][j]=k+1;

break;

}

}

}

for(int i=0;i<n;i++) visit[i]=2;

cnt=0;

int temp;

for(int i=0;i<n;i++)

{

if(st[i][0]==c)

{

visit[i]--;

temp=dfs(i);

if(temp>cnt) cnt=temp;

visit[i]++;

}

}

printf("%d\n",cnt);

}

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

int Len[21],P[21],MAX,N;

char Word[21][100];

void DFS(int d,int s)

{

int i,j,k,l;

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

if (P[i]<2)

for (j=0;j<Len[d];j++)

if (Word[i][0]==Word[d][j])

{

for (l=1,k=j+1;k<Len[d]&&Word[i][l]==Word[d][k];k++,l++);

if (k<Len[d]) continue;

P[i]++;

DFS(i,s+Len[i]-l);

P[i]--;

}

if (s>MAX) MAX=s;

}

int main()

{

int i;

scanf("%d",&N);

for (i=1;i<=N;Len[i]=strlen(Word[i]),i++) scanf("%s",Word[i]);

scanf("%s",Word[0]);

Len[0]=strlen(Word[0]);

DFS(0,Len[0]);

printf("%d",MAX);

return 0;

}

**算法训练 乘积最大**

**问题描述**  
  
　　今年是国际数学联盟确定的“2000——世界数学年”，又恰逢我国著名数学家华罗庚先生诞辰90周年。在华罗庚先生的家乡江苏金坛，组织了一场别开生面的数学智力竞赛的活动，你的一个好朋友XZ也有幸得以参加。活动中，主持人给所有参加活动的选手出了这样一道题目：  
  
　　设有一个长度为N的数字串，要求选手使用K个乘号将它分成K+1个部分，找出一种分法，使得这K+1个部分的乘积能够为最大。  
  
　　同时，为了帮助选手能够正确理解题意，主持人还举了如下的一个例子：  
  
　　有一个数字串：312， 当N=3，K=1时会有以下两种分法：  
  
　　3\*12=36  
　　31\*2=62  
  
　　这时，符合题目要求的结果是：31\*2=62  
  
　　现在，请你帮助你的好朋友XZ设计一个程序，求得正确的答案。  
  
**输入格式**  
  
　　程序的输入共有两行：  
　　第一行共有2个自然数N，K（6≤N≤40，1≤K≤6）  
　　第二行是一个长度为N的数字串。  
  
  
**输出格式**  
  
　　输出所求得的最大乘积（一个自然数）。  
  
　　**样例**输入  
  
　　4 2  
　　1231

样例输出

62

本题的C++参考代码如下：

#include <iostream>

#include <string>

#include <cmath>

#include <cstring>

using namespace std;

int N,K;

string n;

int g[41] = { 0 };

long long int F[41][31] = { 0 };

inline long long int MAX(long long int a, long long int b)

{

return a>b?a:b;

}

long long int aget(int b, int c)

{

long long int temp = 0;

for(int i = b; i <= c; i++)

{

temp = (temp\*10 + g[i]);

}

return temp;

}

int main()

{

cin >> N >> K >> n;

K++;

char buf[N];

strcpy(buf,n.c\_str());

for(int i = 0; i< N; i++) g[i+1] = buf[i] - '0';

for(int i = 1; i <= N; i ++) F[i][1] = aget(1, i );

for(int j = 2; j <= K; j++)

for(int i = 1; i <= N; i++)

for(int k = 1; k < i; k++)

F[i][j] = MAX( F[i][j] , F[k][j-1]\*aget(k+1,i) );

cout << F[N][K];

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define maxN 41

#define maxK 7

#define InfiniteMin -999999999

int main()

{

int N,K;

int i,j,k,m;

int A[maxN][maxK]; /\*A[i][j]表示前i个数有j个乘号能达到的最大乘积\*/

int s[maxN];

char num[maxN];

int temp,max;

scanf("%d%d%s",&N,&K,num);

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

s[i+1]=num[i]-'0';

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

{

temp=0;

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

temp=temp\*10+s[j];

A[i][0]=temp;

}

for(j=1;j<=K;j++)

{

for(i=j+1;i<=N;i++)

{

max=InfiniteMin;

for(k=i;k-1>j-1;k--)

{

temp=0;

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

temp=temp\*10+s[m];

temp\*=A[k-1][j-1];

max=max>temp?max:temp;

}

A[i][j]=max;

}

}

printf("%d",A[N][K]);

return 0;

}

**算法训练 进制转换**

**问题描述**  
  
　　我们可以用这样的方式来表示一个十进制数： 将每个阿拉伯数字乘以一个以该数字所处位置的（值减１）为指数，以１０为底数的幂之和的形式。例如：１２３可表示为 １＊１０２＋２＊１０１＋３＊１００这样的形式。  
　　与之相似的，对二进制数来说，也可表示成每个二进制数码乘以一个以该数字所处位置的（值－１）为指数，以２为底数的幂之和的形式。一般说来，任何一个正整数Ｒ或一个负整数－Ｒ都可以被选来作为一个数制系统的基数。如果是以Ｒ或－Ｒ为基数，则需要用到的数码为 ０，１，．．．．Ｒ－１。例如，当Ｒ＝７时，所需用到的数码是０，１，２，３，４，５和６，这与其是Ｒ或－Ｒ无关。如果作为基数的数绝对值超过１０，则为了表示这些数码，通常使用英文字母来表示那些大于９的数码。例如对１６进制数来说，用Ａ表示１０，用Ｂ表示１１，用Ｃ表示１２，用Ｄ表示１３，用Ｅ表示１４，用Ｆ表示１５。  
　　在负进制数中是用－Ｒ 作为基数，例如－１５（十进制）相当于１１０００１（－２进制），并且它可以被表示为２的幂级数的和数：  
　　１１０００１＝１＊（－２）５＋１＊（－２）４＋０＊（－２）３＋０＊（－２）２＋  
　　０＊（－２）１ ＋１＊（－２）０  
　　设计一个程序，读入一个十进制数和一个负进制数的基数, 并将此十进制数转换为此负进制下的数： －Ｒ∈｛－２，－３，－４，．．．，－２０｝  
  
**输入格式**  
　　一行两个数，第一个是十进制数Ｎ（－32768＜＝Ｎ＜＝32767）， 第二个是负进制数的基数－Ｒ。  
  
**输**出格式  
　　输出所求负进制数及其基数，若此基数超过１０，则参照１６进制的方式处理。（格式参照样例）  
  
　　**样例**输入1  
　　30000 -2

样例输出

30000=11011010101110000(base-2)

样例输入

-20000 -2

样例输出

-20000=1111011000100000(base-2)

样例输入

28800 -16

样例输出

28800=19180(base-16)

样例输入

-25000 -16

样例输出

-25000=7FB8(base-16)

本题的C++参考代码如下：

#include <iostream>

#include <cstdio>

using namespace std;

int N,M,base;

inline int Div(int a,int b)

{

int n;

n = a / b;

if(n\*b <= a)

return n;

return n + 1;

}

void work()

{

int num[100];

int top = 0;

if(N == 0)

{

cout << 0;

}

int P;

while(N)

{

P = Div(N,base);

num[++top] = N - P\*base;

N = P;

}

for(;top >= 1;top--)

{

if(num[top]<10)

cout << num[top];

if(num[top] >= 10)

cout << (char)(num[top] - 10 + 'A');

}

cout << "(base" << base << ")" << endl;

}

int main()

{;

while(cin >> N)

{

cin >> base;

M = N;

cout << N << "=";

work();

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

const char nc[20]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I','J'};

char str[20];

int main()

{

int m, n, k, t, s;

int i=0;

scanf("%d%d",&m,&n);

s = m;

while(m != 0)

{

k = m % n;

t = m / n;

if(k < 0)

{

k -= n;

t++;

}

m = t;

str[i++]=nc[k];

}

printf("%d=",s);

for(i = i- 1; i >= 0; i--)

printf("%c",str[i]);

printf("(base%d)\n",n);

return 0;

}

**算法训练 旅行家的预算**

问题描述

　　一个旅行家想驾驶汽车以最少的费用从一个城市到另一个城市（假设出发时油箱是空的）。给定两个城市之间的距离D1、汽车油箱的容量C（以升为单位）、每升汽油能行驶的距离D2、出发点每升汽油价格P和沿途油站数N（N可以为零），油站i离出发点的距离Di、每升汽油价格Pi（i=1，2，……N）。计算结果四舍五入至小数点后两位。如果无法到达目的地，则输出“No Solution”。

输入格式

　　第一行为4个实数D1、C、D2、P与一个非负整数N；  
　　接下来N行，每行两个实数Di、Pi。

输出格式

　　如果可以到达目的地，输出一个实数（四舍五入至小数点后两位），表示最小费用；否则输出“No Solution”（不含引号）。

样例输入

275.6 11.9 27.4 2.8 2  
102.0 2.9  
220.0 2.2

样例输出

26.95

本题的C++参考代码如下：

#include<cstdio>

using namespace std;

int main(){

int n;

double d1,c,d2,p0;

double p[1000],d[1000];

scanf("%lf%lf%lf%lf%d",&d1,&c,&d2,&p0,&n);

for(int i=1;i<=n;i++){

scanf("%lf%lf",&d[i],&p[i]);

}

d[0]=0;

p[0]=p0;

d[n+1]=d1;

p[n+1]=0;

double sum=0,ct=0;

bool flag=1;

for(int i=0,j;i<=n;i=j){

flag=0;

if(d[i+1]-d[i]>c\*d2)break;

flag=1;

for(j=i+1;j<=n+1;j++) {

if(d[j]-d[i]>c\*d2){

j--;

break;

}

if(p[j]<=p[i]) break;

}

if(p[j]<=p[i]){

sum+=((d[j]-d[i])/d2-ct)\*p[i];

ct=0;

}else{

sum+=(c-ct)\*p[i];

ct=c-(d[j]-d[i])/d2;

}

}

if(flag)printf("%.2lf\n",sum);

else printf("No Solution\n");

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define MAX\_N 100

float D1,C,D2,P;

float A[MAX\_N],B[MAX\_N];

int N;

void solve()

{

int i;

float res = 0;

float pos,tank;

pos = B[0];

tank = C;

res += pos\*tank;

float we=0;

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

{

tank -= (A[i] - A[i-1]-we)/D2;

if(B[i] < pos)

{

we = 0;

res -= tank\*pos;

tank = C;

pos = B[i];

res += tank\*pos;

}

else

{

if(tank < (A[i+1]-A[i])/D2)

{

we=0;

we = tank\*D2;

tank = C;

pos = B[i];

res += tank\*pos;

}

if(tank < 0)

{

puts("No Solution\n");

return ;

}

}

}

tank -= (D1-A[i-1]-we)/D2;

if(tank < 0)

{

puts("No Solution\n");

return ;

}

res -= tank\*pos;

printf("%.2f\n",res);

}

int main()

{

int i;

scanf("%f%f%f%f%d",&D1,&C,&D2,&P,&N);

A[0] = 0;

B[0] = P;

N++;

A[N] = D1;

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

scanf("%f%f",&A[i],&B[i]);

if(D1 != 0)

solve();

else

puts("No Solution\n");

return 0;

}

**算法训练 回文数**

问题描述

　　若一个数（首位不为零）从左向右读与从右向左读都一样，我们就将其称之为回文数。  
　　例如：给定一个10进制数56，将56加65（即把56从右向左读），得到121是一个回文数。  
  
　　又如：对于10进制数87：  
　　STEP1：87+78 = 165 STEP2：165+561 = 726  
　　STEP3：726+627 = 1353 STEP4：1353+3531 = 4884  
  
　　在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。  
  
　　写一个程序，给定一个N（2<=N<=10或N=16）进制数M（其中16进制数字为0-9与A-F），求最少经过几步可以得到回文数。  
　　如果在30步以内（包含30步）不可能得到回文数，则输出“Impossible!”

输入格式

　　两行，N与M

输出格式

　　如果能在30步以内得到回文数，输出“STEP=xx”（不含引号），其中xx是步数；否则输出一行”Impossible!”（不含引号）

样例输入

9  
87

样例输出

STEP=6

本题的C++参考代码如下：

#include<cstdio>

#include<cstring>

#include<iostream>

using namespace std;

struct bign{

int s[1000];

int n;

int len;

bign(){

memset(s,0,sizeof(s));

n=10;

len=1;

}

bign(int n,char \*m){

int l=strlen(m);

len=l;

memset(s,0,sizeof(s));

this->n=n;

for(int i=l-1,j=0;i>=0;i--){

if(m[i]>='0' && m[i]<='9')

s[j++]=m[i]-'0';

else

s[j++]=m[i]-'A'+10;

}

}

bign operator+(const bign&x)const{

bign ans;

int len=this->len>x.len?this->len:x.len+1;

int n=x.n;

for(int i=0;i<len;i++){

ans.s[i]+=x.s[i]+this->s[i];

ans.s[i+1]=ans.s[i]/n;

ans.s[i]%=n;

}

if(len>1 && ans.s[len-1]==0)len--;

ans.n=n;

ans.len=len;

return ans;

}

bign operator=(const bign&x){

this->len=x.len;

this->n=x.n;

for(int i=0;i<x.len;i++){

this->s[i]=x.s[i];

}

return \*this;

}

bign reversal()const{

bign ans;

ans.len=this->len;

ans.n=this->n;

for(int i=this->len-1,j=0;i>=0;i--){

ans.s[j++]=this->s[i];

}

return ans;

}

bool isPalindromes()const{

int i=this->len-1;

int j=0;

while(i>j){

if(this->s[i]!=this->s[j])break;

i--;

j++;

}

return i<=j;

}

};

ostream&operator<<(ostream&out,const bign&x){

for(int i=x.len-1;i>=0;i--){

if(x.s[i]<10)

out<<x.s[i];

else

out<<x.s[i]-10+'A';

}

return out;

}

int main(){

int n;

char m[1000];

scanf("%d",&n);

getchar();

gets(m);

bign a(n,m);

int i=0;

while(!a.isPalindromes()){

if(i>30)break;

i++;

bign b=a.reversal();

a=a+b;

}

if(i<=30)printf("STEP=%d\n",i);

else printf("Impossible!\n");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<math.h>

#include <string.h>

#include <stdlib.h>

int p;

char s[1000];

int c[1000];

int reverse(int a[],int n) /\*判断a是否是回文\*/

{

int i,j;

i=0;

j=n-1;

while(i<j&&a[i]==a[j])

{

i++;

j--;

}

return i>=j;

}

int aplus(int a[],int n,int m) /\*计算a=a+a\*/

{

int \*b,i,j,kc;

b=(int \*)malloc(sizeof(int)\*1001);

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

b[n-1-i]=a[i];

kc=0;

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

{

a[i]=a[i]+b[i]+kc;

kc=a[i]/m;

a[i]=a[i]%m;

}

if(kc>0)

a[n++]=kc;

free(b);

return n;

}

int stod(char s[],int a[])

{

int i;

for(i=0;s[i]!='\0';i++)

if(s[i]>='A'&&s[i]<='F')

a[i]=10+s[i]-'A';

else

a[i]=s[i]-'0';

return i;

}

int main()

{

int n,i;

scanf("%d",&p);

scanf("%s",s);

n=stod(s,c);

for(i=0;i<30;i++)

{

n=aplus(c,n,p);

if(reverse(c,n)==1)

{

printf("STEP=%d\n",i+1);

break;

}

}

if(i>=30)

printf("Impossible!\n");

return 0;

}

**算法训练 拦截导弹**

问题描述

　　某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。  
  
　　输入导弹依次飞来的高度（雷达给出的高度数据是不大于30000的正整数），计算这套系统最多能拦截多少导弹，如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

输入格式

　　一行，为导弹依次飞来的高度

输出格式

　　两行，分别是最多能拦截的导弹数与要拦截所有导弹最少要配备的系统数

样例输入

389 207 155 300 299 170 158 65

样例输出

6  
2

本题的C++参考代码如下：

#include <cstdio>

#include <cstdlib>

#include <cstring>

using namespace std;

int a[10001],high[10001];

int n = 1, top, maw;

void work(int p)

{

int l = 1, r = n, mid;

while(l != r)

{

mid = (l + r) / 2;

if(p < high[mid]) l = mid + 1;

else r = mid;

}

if(p > high[l]) high[l] = p;

if(l > maw) maw = l;

}

void solve(int p)

{

int l = 1, r = n, mid;

while(l != r)

{

mid = (l + r) / 2;

if(p > high[mid]) l = mid + 1;

else r = mid;

}

if(p < high[l]) high[l] = p;

if(l > top) top = l;

}

int main()

{

//freopen("missile.txt","r",stdin);

//freopen("missiles.txt","w",stdout);

while(scanf("%d", &a[n]) != EOF) ++n;

--n;

for(int i = 1; i <= n; ++i)

work(a[i]);

memset(high, 1, sizeof(high));

for(int i = 1; i <= n; ++i) solve(a[i]);

printf("%d\n%d\n", maw, top);

// system("pause");

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define MAX\_N 10000

int dp[MAX\_N]={0};

int i=0;

int dao[MAX\_N];

int max(int a,int b)

{

return a>b?a:b;

}

void solve1()

{

int k,j;

int res = 0;

int n = i;

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

{

dp[j] = 1;

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

if(dao[k] > dao[j])

dp[j] = max(dp[j],dp[k]+1);

res = max(res , dp[j]);

}

printf("%d\n",res);

}

void solve2()

{

int k,j;

int res = 0;

int n = i;

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

{

dp[j] = 1;

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

if(dao[k] < dao[j])

dp[j] = max(dp[j],dp[k]+1);

res = max(res , dp[j]);

}

printf("%d\n",res);

}

int main()

{

char q;

int s=0;

while(q=getchar())

{

if(q>='0' && q<='9')

s = s\*10+q-'0';

else if(q == ' ')

{

dao[i] = s;

i++;

s = 0;

}

else

break;

}

dao[i] = s;

i++;

solve1();

solve2();

return 0;

}

**算法训练 幂方分解**

问题描述

　　任何一个正整数都可以用2的幂次方表示。例如：  
　　137=27+23+20  
　　同时约定方次用括号来表示，即ab 可表示为a（b）。  
　　由此可知，137可表示为：  
　　2（7）+2（3）+2（0）  
　　进一步：7= 22+2+20（21用2表示）  
　　3=2+20  
　　所以最后137可表示为：  
　　2（2（2）+2+2（0））+2（2+2（0））+2（0）  
　　又如：  
　　1315=210 +28 +25 +2+1  
　　所以1315最后可表示为：  
　　2（2（2+2（0））+2）+2（2（2+2（0）））+2（2（2）+2（0））+2+2（0）

输入格式

　　输入包含一个正整数N（N<=20000），为要求分解的整数。

输出格式

　　程序输出包含一行字符串，为符合约定的n的0，2表示（在表示中不能有空格）

本题的C++参考代码如下：

#include<cstdio>

#include<stack>

using namespace std;

void f(int n){

int i=0;

stack<int>s;

if(n<=2)

printf("%d",n);

else while(n>0){

if(n&1){

s.push(i);

}

n>>=1;

i++;

}

bool p=0;

while(!s.empty()){

if(p)printf("+");

p=1;

i=s.top();

s.pop();

if(i!=1){

printf("2(");

f(i);

printf(")");

}else printf("2");

}

}

int main(){

int n;

scanf("%d",&n);

f(n);

printf("\n");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

void f(int a)

{

int i=0,j,b[32],w,k;

if(a==0)printf("0");

else if(a==2)printf("2");

else if(a==1)printf("2(0)");

else

{

while(a){b[i]=a%2;a=a/2;i++;}w=i;

k=0;j=0;for(i=w-1;i>=0;i--)if(b[i])k++;

for(i=w-1;i>=0;i--)

if(b[i])

{j++;

if(i==1)printf("2");

else {printf("2(");f(i);printf(")");}

if(j!=k)printf("+");}

}

}

int main()

{

int a;scanf("%d",&a);

f(a);

return 0;

}

**算法训练 瓷砖铺放**

问题描述

　　有一长度为N(1<=Ｎ<=10)的地板，给定两种不同瓷砖：一种长度为1，另一种长度为2，数目不限。要将这个长度为N的地板铺满，一共有多少种不同的铺法？  
　　例如，长度为4的地面一共有如下5种铺法：  
　　4=1+1+1+1  
　　4=2+1+1  
　　4=1+2+1  
　　4=1+1+2  
　　4=2+2  
　　编程用递归的方法求解上述问题。

输入格式

　　只有一个数N，代表地板的长度

输出格式

　　输出一个数，代表所有不同的瓷砖铺放方法的总数

样例输入

4

样例输出

5

本题的C++参考代码如下：

#include<cstdio>

#include<cstdlib>

int a[20];

int main()

{

int n;

scanf("%d",&n);

a[0]=1; a[1]=1;

for(int i=2;i<=n;i++) a[i]=a[i-1]+a[i-2];

printf("%d\n",a[n]);

//system("pause");

}

本题的C参考代码如下：

#include <stdio.h>

int j(int a)

{

int i=1,s=1;

for(;i<=a;i++)s\*=i;

return s;

}

int main()

{

int a;scanf("%d",&a);

int s=0,b=0,i;

while(a>=0)

{

s+=j(a+b)/j(a)/j(b);

a-=2;b++;

}

printf("%d",s);

return 0;

}

**算法训练 集合运算**

问题描述

　　给出两个整数集合A、B，求出他们的交集、并集以及B在A中的余集。

输入格式

　　第一行为一个整数n，表示集合A中的元素个数。  
　　第二行有n个互不相同的用空格隔开的整数，表示集合A中的元素。  
　　第三行为一个整数m，表示集合B中的元素个数。  
　　第四行有m个互不相同的用空格隔开的整数，表示集合B中的元素。  
　　集合中的所有元素均为int范围内的整数，n、m<=1000。

输出格式

　　第一行按从小到大的顺序输出A、B交集中的所有元素。  
　　第二行按从小到大的顺序输出A、B并集中的所有元素。  
　　第三行按从小到大的顺序输出B在A中的余集中的所有元素。

样例输入

5  
1 2 3 4 5  
5  
2 4 6 8 10

样例输出

2 4  
1 2 3 4 5 6 8 10  
1 3 5

样例输入

4  
1 2 3 4  
3  
5 6 7

样例输出

1 2 3 4 5 6 7  
1 2 3 4

本题的C++参考代码如下：

#include<cstdio>

#include<algorithm>

using namespace std;

int ta[2005];

int main()

{

int n,m,a[2005],b[2005];

int i,j,k;

scanf("%d",&n);

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

scanf("%d",&a[i]);

scanf("%d",&m);

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

scanf("%d",&b[i]);

sort(a,a+n);

sort(b,b+m);

k=0;

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

{

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

{

if(a[i]<=b[j])

break;

}

if(j!=m)

{

if(a[i]==b[j])

{

ta[i]=-1;

k=1;

printf("%d ",a[i]);

}

}

}

if(k)

printf("\n");

i=0;j=0;

while(i<n&&j<m)

{

if(a[i]<b[j])

{

if(ta[i]!=-1)

printf("%d ",a[i]);

i++;

}

else

{

printf("%d ",b[j]);

j++;

}

}

for(i;i<n;i++)

{

if(ta[i]!=-1)

printf("%d ",a[i]);

}

for(j;j<m;j++)

{

printf("%d ",b[j]);

}

printf("\n");

k=0;

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

{

if(ta[i]!=-1)

{

k=1;

printf("%d ",a[i]);

}

}

if(k)

printf("\n");

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define MAX\_N 1001

int n,m;

int Partition(int x[],int low,int high)

{

int pivotkey;

pivotkey = x[low];

while(low < high)

{

while(low<high && x[high]>=pivotkey)

high--;

x[low] = x[high];

while(low<high && x[low]<=pivotkey)

low++;

x[high] = x[low];

}

x[low] = pivotkey;

return low;

}

void QSort(int x[],int low,int high)

{

int pivotloc;

if(low < high)

{

pivotloc = Partition(x,low,high);

QSort(x,low,pivotloc-1);

QSort(x,pivotloc+1,high);

}

}

void jiao(int x[],int y[],int xx,int yy)

{

int i,j;

i = 1;

j = 1;

while(i<=xx && j<=yy)

{

if(x[i] == y[j])

{

printf("%d ",x[i]);

i++;

j++;

}

else if(x[i] < y[j])

i++;

else

j++;

}

printf("\n");

}

int bing(int x[],int y[],int z[],int xx,int yy)

{

int i,j,k;

i = 1;

j = 1;

k = 1;

while(i<=xx && j<=yy)

{

if(x[i] < y[j])

{

z[k] = x[i];

i++;

}

else if(x[i] > y[j])

{

z[k] = y[j];

j++;

}

else

{

z[k] = x[i];

i++;

j++;

}

k++;

}

if(i > xx)

while(j <= yy)

{

z[k] = y[j];

k++;

j++;

}

else

while(i <= xx)

{

z[k] = x[i];

k++;

i++;

}

return k;

}

void yu(int x[],int y[],int xx,int yy)

{

int i,j;

i = 1;

j = 1;

while(i<=xx && j<=yy)

{

if(x[i] == y[j])

{

i++;

j++;

}

else if(x[i] < y[j])

{

printf("%d ",x[i]);

i++;

}

else

j++;

}

if(j > yy)

while(i <= xx)

{

printf("%d ",x[i]);

i++;

}

printf("\n");

}

int main()

{

int i,l;

int a[MAX\_N],b[MAX\_N];

int c[2014];

scanf("%d",&n);

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

scanf("%d",&a[i]);

QSort(a,1,n);

scanf("%d",&m);

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

scanf("%d",&b[i]);

QSort(b,1,m);

jiao(a,b,n,m);

l = bing(a,b,c,n,m);

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

printf("%d ",c[i]);

printf("\n");

yu(a,b,n,m);

return 0;

}

**算法训练 摆动序列**

问题描述

　　如果一个序列满足下面的性质，我们就将它称为摆动序列：  
　　1. 序列中的所有数都是不大于*k*的正整数；  
　　2. 序列中至少有两个数。  
　　3. 序列中的数两两不相等；  
　　4. 如果第*i* – 1个数比第*i* – 2个数大，则第*i*个数比第*i* – 2个数小；如果第*i* – 1个数比第*i* – 2个数小，则第*i*个数比第*i* – 2个数大。  
　　比如，当*k* = 3时，有下面几个这样的序列：  
　　1 2  
　　1 3  
　　2 1  
　　2 1 3  
　　2 3  
　　2 3 1  
　　3 1  
　　3 2  
　　一共有8种，给定*k*，请求出满足上面要求的序列的个数。

输入格式

　　输入包含了一个整数*k*。（*k*<=20）

输出格式

　　输出一个整数，表示满足要求的序列个数。

样例输入

3

样例输出

8

本题的C++参考代码如下：

#include<stdio.h>

int main()

{

int n,i,a[30],sum=0;

scanf("%d",&n);

a[2]=2;

sum+=a[2];

for(i=3;i<=n;i++)

{

a[i]=sum+i\*(i-1);

sum+=a[i];

}

printf("%d",a[n]);

return 0;

}

本题的C参考代码如下：

//天农

#include<stdio.h>

int f[21][21][21]; //f[i][j][k] i表示数的长度，j表示倒数第2位的取值，k表示最后一位的取值

int main()

{

int n,i,j,k,p;

int sum=0;

scanf("%d",&n);

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

for(j=1;j<=n;j++)

if(i!=j)

f[2][i][j]=1;

for(i=3;i<=n;i++)//从长度为3开始

{

for(j=1;j<=n;j++)

{

for(k=1;k<=n;k++)

{

for(p=1;p<=n;p++)

{

if(j>p&&k<p||j<p&&k>p)

f[i][j][k]+=f[i-1][p][j];

}

}

}

}

for(i=2;i<=n;i++)

for(j=1;j<=n;j++)

for(k=1;k<=n;k++)

sum+=f[i][j][k];

printf("%d",sum);

return 0;

}