第一题

代码：

#include<stdio.h>

int main()

{

double a,n,t;

double sum=0;

printf("请输入n的值：");

scanf("%lf",&n);

t=2;

++n;

while(n--)

{

sum+=a;

a=t+a\*10;

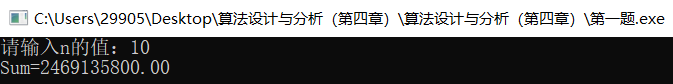
}

printf("Sum=%.2lf",sum);

return 0;

}

运行截图：



第二题

代码：

#include <stdio.h>

int main()

{

int a, i, sum = 2;

for (i = 9; i > 0; i--)

{

sum=(sum+2)\*2;

a = sum;

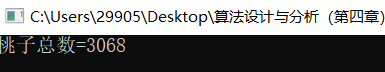
}

printf("桃子总数=%d\n", a);

return 0;

}

运行截图：



第三题

代码：

#include<stdio.h>

int main(){

int s=0; //记录总张数

int k; //记录一次出的牌张数

int t; //最后为计算机出牌时，t为1，人出牌时，t为0

t=1; //首先计算机出牌，t变为1

k=3;

printf("机:%d\n",k);

s=s+k;

do{

printf("现在剩%d张\n",54-s); //提示玩家剩余张数

t=0; //玩家出牌，t变为0

printf("人:");

scanf("%d",&k);

while(k<1||k>4) //判断玩家出牌是否符合规则

{

printf("每人每次最少取1张牌，最多取4张，请重新输入\n");

t=0;

printf("人:");

scanf("%d",&k);

};

s=s+k;

t=1;

k=5-k;

printf("机:%d\n",k);

s=s+k;

}

while(s<53);

printf("现在剩%d张\n",54-s);

t=0;

printf("人:");

scanf("%d",&k);

s=s+k;

if(s==54)

printf("%d张扑克牌已取完\n",s);

if(t==0)

printf("计算机胜!");

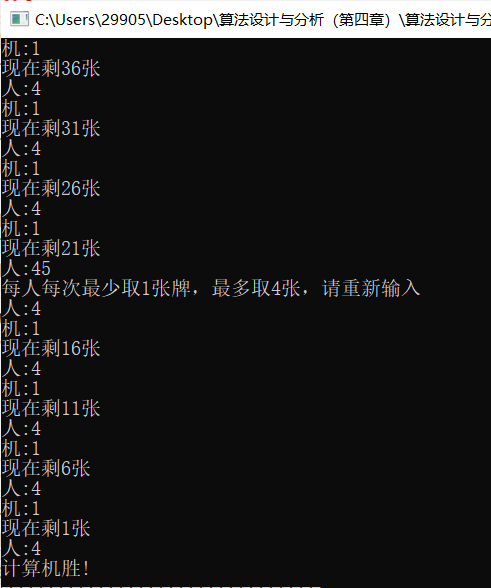
else

printf("玩家胜!");

return 0;

}

运行截图：



第四题

代码：

#include <iostream>

#include <cmath>

using namespace std;

int ak(int N, int d, int a1, int a2, int n) {

int a[60];

a[0] = a1;

a[1] = a2;

for(int i = 2; i < 60; i ++)

a[i] = a[i - 2] - 2 \* a[i - 1] + 2 \* d;

cout << a[n] << endl;

}

int main()

{

int N, d, a1, a2, n;

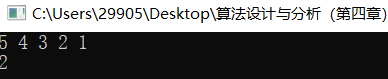
cin >> N >> d >> a1 >> a2 >> n;

ak(N, d, a1, a2, n);

return 0;

}

运行截图：



第五题

代码：

#include <stdio.h>

#include <math.h>

int check\_1\_8( long m )//检查1-8是否各出现一次，若是，则返回1，否则返回0

{

int f[10], i, flag;

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

f[i] = 0;

while ( m!=0 )

{

f[m%10]++;

m = m / 10;

}

for ( flag=1, i=1; i<=8; i++ )

{

if ( f[i]!=1 )

{

flag = 0;

break;

}

}

return flag;

}

int check\_dia( long m )//检测是否在同一对角线上，若不在，返回1，否则返回0；

{

int g[9], i, j, flag;

for ( i=8; i>=1; i-- )

{

g[i] = m % 10;

m = m / 10;

}

flag = 1;

for ( i=1; i<=7; i++ )

{

for ( j=i+1; j<=8; j++ )

{

if ( abs(g[j]-g[i]) == j-i)

{

flag = 0;

break;

}

}

}

return flag;

}

int main()

{

int n=0; //总数

long a;

for ( a=12345678; a<=87654321; a+=9 ) //枚举所有可能数

{

if ( check\_1\_8( a ) == 0 ) //检查数字1-8

continue;

if ( check\_dia( a ) == 0 ) //检测对角线

continue;

printf("%ld ", a); //得到一个解

n++;

if ( n%6==0 ) //每行打印6个

printf("\n");

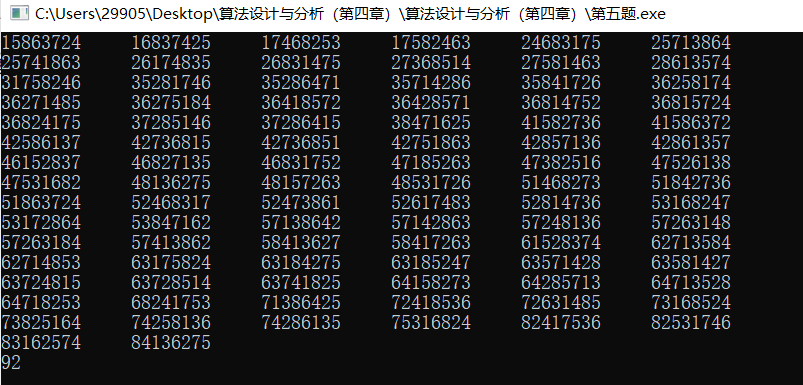
}

printf("\n%d\n", n); //输出总的方案数

return 0;

}

运行截图：



第六题

代码：

#include<stdio.h>

int main(void)

{

int Big, Middle, Small, sum; //sum指被马驼的货物总数

for (Big = 0; Big <= 33; Big++) //大马不可能超过 33 匹

{

for (Middle = 0; Middle <= 50; Middle++) //中马不可能超过 50 匹

{

Small = 100 - Big - Middle; //小马的数量通过大马和中马的数量控制就可以了

if (Small % 2 != 0) //小马数量必须为偶数

continue; //保证小马数量为偶数

sum = 3 \* Big + 2 \* Middle + Small / 2; //计算共驼了多少

if (sum == 100)

{

printf("大马:%d 中马:%d 小马:%d\n", Big, Middle, Small);

}

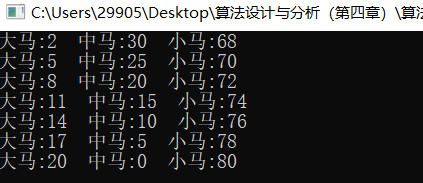
}

}

return 0;

}

运行截图：



第七题

代码：

# include<iostream>

using namespace std;

int main()

{

for(int i=1;i<25;i+=2)

{

int a=7\*i;

if(a%3==2&&a%5==4&&a%6==5)

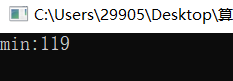
cout<<"min:"<<a<<endl;

}

return 0;

}

运行截图：



第八题

代码：

#include<stdio.h>

int main(){

for(int i=1;i<=9;i++){

for(int j=0;j<=9;j++){

if(j!=i){

for(int k=0;k<=9;k++){

if(k!=j&&k!=i){

for(int n=0;n<=9;n++){

if(n!=k&&n!=j&&n!=i){

int temp=i\*1000+j\*100+k\*10+n;

int sum=(i+j+k+n)\*(i+j+k+n);

if(temp%sum==0) printf("%d\n",temp);

}

}

}

}

}

}

}

return 0;

}

运行截图：



第九题

代码：

#include<stdio.h>

#define N 10

void max\_min(int \*a,int m,int n,int \*min1,int \*min2,int \*max1,int \*max2);

int main(void)

{

int a[N]={2,3,4,5,34,7,9,6,43,21};

int min1,min2;

int max1,max2;

max\_min(a,0,N-1,&min1,&min2,&max1,&max2);

//这四个参数的大小关系为 min1<min2<max2<max1

printf("min1=%d min2=%d\n max1=%d max2=%d\n",min1,min2,max1,max2);

return 0;

}

void max\_min(int \*a,int m,int n,int \*min1,int \*min2,int \*max1,int \*max2)

{

int lmin1,lmin2,lmax1,lmax2;

int rmin1,rmin2,rmax1,rmax2;

int mid;

if(m==n)//分治子数组中只有一个数

{

\*min1=\*min2=\*max1=\*max2=a[m];

}

else//分治子数组中不止一个数

if(m==n-1)//分治子数组中仅有2个数

{

if(a[m]<a[n])

{

\*min1=a[m];

\*min2=a[n];

\*max1=a[n];

\*max2=a[m];

}

else

{

\*min1=a[n];

\*min2=a[m];

\*max1=a[m];

\*max2=a[n];

}

}

else//分治子数组中有超过2个数

{

mid=(m+n)/2;

max\_min(a,m,mid,&lmin1,&lmin2,&lmax1,&lmax2);

max\_min(a,mid+1,n,&rmin1,&rmin2,&rmax1,&rmax2);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//确定出数组中最小的两个数

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(lmin1<rmin1)//左子数组最小数<右子数组最小数

{

if(lmin2<rmin1)

{

\*min1=lmin1;

\*min2=lmin2;

}

else

{

\*min1=lmin1;

\*min2=rmin1;

}

}

else//右子数组最小数<左子数组最小数

if(rmin2<lmin1)

{

\*min1=rmin1;

\*min2=rmin2;

}

else

{

\*min1=rmin1;

\*min2=lmin1;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//确定出数组中最大的两个数

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(lmax1>rmax1)//左子数组最大数>右子数组最大数

{

if(lmax2>rmax1)

{

\*max1=lmax1;

\*max2=lmax2;

}

else

{

\*max1=lmax1;

\*max2=rmax1;

}

}

else//右子数组最大数>左子数组最大数

if(rmax2>lmax1)

{

\*max1=rmax1;

\*max2=rmax2;

}

else

{

\*max1=rmax1;

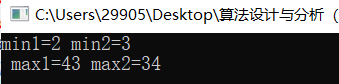
\*max2=lmax1;

}

}

}

运行截图：



第十题

代码：

#include<stdio.h>

#define N 10

int getsum(int \*a,int l,int r)//传数组a的每一个的值然后便于用来加减

{

int mid;

if(l==r)

return a[l];//如果数组只有一个数，返回这个数即可

else if(l==r-1)

return a[l]+a[r];//如果有数组只有两个数，返回值为这两个数的值相加

else

{

mid=(l+r)/2;

return getsum(a,l,mid)+getsum(a,mid+1,r);//利用递归来做分治数组求和

}

}

int main()

{

int c;

int a[N]={1,2,3,4,5,6,7,8,9,10};//定义数组，拿来求和

c=getsum(a,0,N-1);//获取要的到的值，即为数组的和

printf("最后分治法求的和为：%d",c);

}

运行截图：



第十一题

代码：

#include<stdio.h>

int maxSub(int a[]);//求最大子序列和

int main(void)

{

int a[5][3]={

{-50,17,-42},

{-47,-19,-3},

{36,-34,-43},

{-30,-43,34},

{-23,-8,-45}

};

int i,j;

int max,n;

int maxRoute=0;

int route[10];//用来存放旅游街每一列的最大分值

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

{

max=a[i][0];//max为第i行的最大分值

n=0;//最大分值列号

for(j=0;j<3;j++)//遍历每一行元素

{

if(a[i][j]>max)

{

max=a[i][j];

n=j;

}

}

route[i]=max;

printf("a[%d][%d]=%d\n",i,n,max);

}

maxRoute=maxSub(route);

printf("最佳路线长度为：%d\n",maxRoute);

return 0;

}

int maxSub(int a[])

{

int maxSum=0,headSum=0;

int i;

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

{

headSum+=a[i];

if(headSum>maxSum)

maxSum=headSum;

else if(headSum<0)

headSum=0;

}

return maxSum;

}

运行截图：



第十二题

代码：

#include <iostream>

using namespace std;

const int N=4; //常量N用来定义矩阵的大小

int main(void)

{

void STRASSEN(int n,float A[][N],float B[][N],float C[][N]);

void input(int n,float p[][N]);

void output(int n,float C[][N]); //函数声明部分

float A[N][N],B[N][N],C[N][N]; //定义三个矩阵A,B,C

cout<<"现在录入矩阵A[N][N]:"<<endl<<endl;

input(N,A);

cout<<endl<<"现在录入矩阵B[N][N]:"<<endl<<endl;

input(N,B); //录入数组

STRASSEN(N,A,B,C); //调用STRASSEN函数计算

output(N,C); //输出计算结果

getchar();

}

void input(int n,float p[][N]) //矩阵输入函数

{

int i,j;

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

{

cout<<"请输入第"<<i+1<<"行"<<endl;

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

cin>>p[i][j];

}

}

void output(int n,float C[][N]) //据矩阵输出函数

{

int i,j;

cout<<"输出矩阵:"<<endl;

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

{

cout<<endl;

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

cout<<C[i][j]<<" ";

}

cout<<endl<<endl;

}

void MATRIX\_MULTIPLY(float A[][N],float B[][N],float C[][N]) //按通常的矩阵乘法计算C=AB的子算法(仅做2阶)

{

int i,j,t;

for(i=0;i<2;i++) //计算A\*B-->C

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

{

C[i][j]=0; //计算完一个C[i][j]，C[i][j]应重新赋值为零

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

C[i][j]=C[i][j]+A[i][t]\*B[t][j];

}

}

void MATRIX\_ADD(int n,float X[][N],float Y[][N],float Z[][N]) //矩阵加法函数X+Y—>Z

{

int i,j;

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

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

Z[i][j]=X[i][j]+Y[i][j];

}

void MATRIX\_SUB(int n,float X[][N],float Y[][N],float Z[][N]) //矩阵减法函数X-Y—>Z

{

int i,j;

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

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

Z[i][j]=X[i][j]-Y[i][j];

}

void STRASSEN(int n,float A[][N],float B[][N],float C[][N]) //STRASSEN函数（递归）

{

float A11[N][N],A12[N][N],A21[N][N],A22[N][N];

float B11[N][N],B12[N][N],B21[N][N],B22[N][N];

float C11[N][N],C12[N][N],C21[N][N],C22[N][N];

float M1[N][N],M2[N][N],M3[N][N],M4[N][N],M5[N][N],M6[N][N],M7[N][N];

float AA[N][N],BB[N][N],MM1[N][N],MM2[N][N];

int i,j;//,x;

if (n==2)

MATRIX\_MULTIPLY(A,B,C);//按通常的矩阵乘法计算C=AB的子算法(仅做2阶)

else

{

for(i=0;i<n/2;i++)

for(j=0;j<n/2;j++)

{

A11[i][j]=A[i][j];

A12[i][j]=A[i][j+n/2];

A21[i][j]=A[i+n/2][j];

A22[i][j]=A[i+n/2][j+n/2];

B11[i][j]=B[i][j];

B12[i][j]=B[i][j+n/2];

B21[i][j]=B[i+n/2][j];

B22[i][j]=B[i+n/2][j+n/2];

} //将矩阵A和B式分为四块

MATRIX\_SUB(n/2,B12,B22,BB);

STRASSEN(n/2,A11,BB,M1);//M1=A11(B12-B22)

MATRIX\_ADD(n/2,A11,A12,AA);

STRASSEN(n/2,AA,B22,M2);//M2=(A11+A12)B22

MATRIX\_ADD(n/2,A21,A22,AA);

STRASSEN(n/2,AA,B11,M3);//M3=(A21+A22)B11

MATRIX\_SUB(n/2,B21,B11,BB);

STRASSEN(n/2,A22,BB,M4);//M4=A22(B21-B11)

MATRIX\_ADD(n/2,A11,A22,AA);

MATRIX\_ADD(n/2,B11,B22,BB);

STRASSEN(n/2,AA,BB,M5);//M5=(A11+A22)(B11+B22)

MATRIX\_SUB(n/2,A12,A22,AA);

MATRIX\_SUB(n/2,B21,B22,BB);

STRASSEN(n/2,AA,BB,M6);//M6=(A12-A22)(B21+B22)

MATRIX\_SUB(n/2,A11,A21,AA);

MATRIX\_SUB(n/2,B11,B12,BB);

STRASSEN(n/2,AA,BB,M7);//M7=(A11-A21)(B11+B12)

//计算M1,M2,M3,M4,M5,M6,M7（递归部分）

MATRIX\_ADD(N/2,M5,M4,MM1);

MATRIX\_SUB(N/2,M2,M6,MM2);

MATRIX\_SUB(N/2,MM1,MM2,C11);//C11=M5+M4-M2+M6

MATRIX\_ADD(N/2,M1,M2,C12);//C12=M1+M2

MATRIX\_ADD(N/2,M3,M4,C21);//C21=M3+M4

MATRIX\_ADD(N/2,M5,M1,MM1);

MATRIX\_ADD(N/2,M3,M7,MM2);

MATRIX\_SUB(N/2,MM1,MM2,C22);//C22=M5+M1-M3-M7

for(i=0;i<n/2;i++)

for(j=0;j<n/2;j++)

{

C[i][j]=C11[i][j];

C[i][j+n/2]=C12[i][j];

C[i+n/2][j]=C21[i][j];

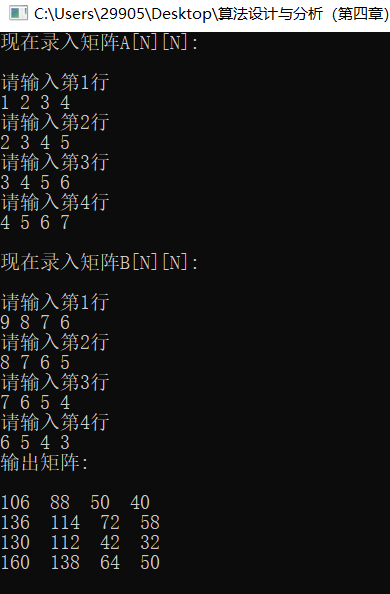
C[i+n/2][j+n/2]=C22[i][j];

} //计算结果送回C[N][N]

}

}

运行截图：



第十三题

代码：

#include<stdio.h>

int main(void)

{

int array1[6][7]={

{16,4,3,12,6,0,3},

{4,-5,6,7,0,0,2},

{6,0,-1,-2,3,6,8},

{5,3,4,0,0,-2,7},

{-1,7,4,0,7,-5,6},

{0,-1,3,4,12,4,2}

};

int b[6][7],c[6][7];

int i,j,k;

int max;

int flag;

for(i=0;i<6;i++)

for(j=0;j<7;j++)

{

b[i][j]=array1[i][j];

c[i][j]=-1;

}

for(i=1;i<5;i++)

{

for(j=0;j<7;j++)

{

max=0;

for(k=j-2;k<=j+2;k++)

{

if(k<0) continue;

else

if(k>6) break;

else

{

if(b[i][j]+b[i-1][k]>max)

{

max=b[i][j]+b[i-1][k];

flag=k;

}

}

}

b[i][j]=max;

c[i][j]=flag;

}

}

for(j=1;j<=5;j++)//i=5

{

max=0;

for(k=j-2;k<=j+2;k++)

{

if(k<0)

continue;

else

if(k>6) break;

else

{

if(b[i][j]+b[i-1][k]>max)

{

max=b[i][j]+b[i-1][k];

flag=k;

}

}

}

b[i][j]=max;

c[i][j]=flag;

}

max=0;

for(j=1;j<=5;j++) //找出

{

if(b[i][j]>max)

{

max=b[i][j];

flag=j;

}

}

printf("从底到顶最大和值为：%d\n\n",max);

printf("从底到顶分别取数：");

printf("%d",array1[i][flag]);

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

{

flag=c[j][flag];

printf("%5d",array1[j-1][flag]);

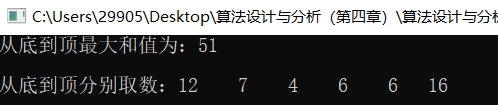
}

printf("\n");

return 0;

}

运行截图：



第十四题

代码：

#include<cstdio>

#include<iostream>

using namespace std;

int main()

{

int a[6][3]={0,0,0,3,5,4,7,10,6,9,11,11,12,11,12,13,11,12};

int i,j,k,a1,b1,c1,sum,maxm=0;

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

{

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

{

k=5-i-j;

sum=a[i][0]+a[j][1]+a[k][2];

if(sum>maxm)

{

maxm=sum;

a1=i;

b1=j;

c1=k;

}

}

}

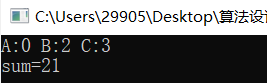
cout<<"A:"<<a1<<" B:"<<b1<<" C:"<<c1<<endl;

cout<<"sum="<<maxm;

return 0;

}

运行截图：



第十五题

代码：

#include<iostream>

using namespace std;

int main(void)

{

int x1,x2,x3;

int min[10];

int k=0,q;

for(x1=0;x1<4;x1++)

for(x2=0;x2<4;x2++)

for(x3=0;x3<4;x3++)

if(x1+x2+x3==3)

{

q=x1\*x1+2\*x2\*x2+x3\*x3-2\*x1-4\*x2-2\*x3;

min[k]=q;

k++;

}

if(k==0)

cout<<"没有满足条件的值"<<endl;

else

{

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

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

if(min[i]<min[i+1])

{

min[i]=q;

min[i+1]=min[i];

min[i]=q;

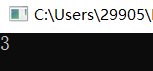
}

}

cout<<min[k]<<endl;

}

运行截图：



第十六题

代码：

#include <stdio.h>

#include <stdlib.h>

#define N 13

//计算数组的和

int sum(int coin[],int n){

int result=0;

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

result+=coin[i];

return result;

}

//判断2个硬币真假

int judge\_1(int coin[]){

int temp=0; //最小值

if(coin[0]<coin[1])

temp=coin[0];

else if(coin[0]>coin[1])

temp=coin[1];

return temp;

}

//判断三个硬币的真假

int judge\_2(int coin[]){

int temp=0; //最小值 （假币）

if(coin[0]<coin[1]&&coin[0]<coin[2]){

temp=coin[0];

}

else if(coin[1]<coin[0]&&coin[1]<coin[2]){

temp=coin[1];

}

else if(coin[2]<coin[1]&&coin[2]<coin[1]){

temp=coin[2];

}

return temp;

}

//把硬币分成3份和一个余数组

void allot(int all[],int a[],int b[],int c[],int d[],int n){

printf("\n%d 枚硬币开始分组...\n\n",n);

int j=(n-n%3)/3; //每一碓硬币数

int k=n%3; //余数组

for(int i=0;i<j;i++) //a数组（第一堆）硬币

{

a[i]=all[i];

}

printf("第一堆硬币数量为：%d 枚\n",j);

int w=0;

for(int i=j;i<2\*j;i++) //b数组（第二堆） 硬币

{

b[w]=all[i];

w++;

}

printf("第二堆硬币数量为：%d 枚\n",j);

w=0;

for(int i=2\*j;i<3\*j;i++) //c数组（第三堆） 硬币

{

c[w]=all[i];

w++;

}

printf("第三堆硬币数量为：%d 枚\n",j);

w=0;

if(k!=0){ //余数堆硬币

for(int i=3\*j;i<N;i++)

{

d[w]=all[i];

w++;

}

printf("余数堆硬币数量为：%d 枚\n",k);

}

else

{

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

d[i]=0;

printf("余数堆硬币数量为0枚\n");

}

}

//硬币个数大于3个时进行比较

int compare(int all[],int a[],int b[],int c[],int d[],int n){

int temp=0; //最小值（假币）

allot(all,a,b,c,d,n);

int m= (n-n%3)/3; //前3堆每一堆硬币数量

int u=n%3; //余数堆硬币数量

int sum\_1=sum(a,m); //第一堆硬币总质量

printf("第一堆硬币总质量为：%d\n",sum\_1);

int sum\_2=sum(b,m); //第二堆硬币总质量

printf("第二堆硬币总质量为：%d\n",sum\_2);

if(sum\_1==sum\_2){ //如果a==b

printf("第一堆硬币总质量等于第二堆硬币总质量，则计算第三堆硬币总质量...\n");

int sum\_3=sum(c,m); //第三堆硬币总质量

printf("第三堆硬币总质量为：%d\n",sum\_3);

if(sum\_1==sum\_3) // 如果a==c ，假币在d中

{

printf("第一堆硬币总质量等于第三堆硬币总质量,则假币在余数堆中\n");

if(u==1) return d[0];

if(u==2) {temp=judge\_1(d);return temp;}

if(u==3) {temp=judge\_2(d);return temp;}

if(u>3) {printf("\n");return compare(d,a,b,c,d,u);} //用减治法迭代

}

else{ //如果a不等于c,假币在c中

printf("第一堆硬币总质量不等于第三堆硬币总质量,则假币在第三堆中\n");

if(m==1) return c[0];

if(m==2) {temp=judge\_1(c);return temp;}

if(m==3) {temp=judge\_2(c);return temp;}

if(m>3) {printf("\n");return compare(c,a,b,c,d,m);}

}

}

else if(sum\_1>sum\_2){//如果a大于b,假币在b中

printf("第一堆硬币总质量大于第二堆硬币总质量,则假币在第二堆中\n");

if(m==1) return b[0];

if(m==2) {temp=judge\_1(b);return temp;}

if(m==3) {temp=judge\_2(b);return temp;}

if(m>3) {printf("\n");return compare(b,a,b,c,d,m);} //迭代

}

else {//如果a小于b,假币在a中

printf("第一堆硬币总质量小于第三堆硬币总质量,则假币在第一堆中\n");

if(m==1) return a[0];

if(m==2) {temp=judge\_1(a);return temp;}

if(m==3) {temp=judge\_2(a);return temp;}

if(m>3) {printf("\n");return compare(a,a,b,c,d,m);} //迭代

}

}

//主函数

int main()

{

int temp=0; //最小值（假币）

int n; //表示硬币个数

int all[N]; //输入的总硬币

int a[15]; //第一堆硬币

int b[15]; //第二堆硬币

int c[15]; //第三堆硬币

int d[15]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; //余数堆硬币

printf("请输入硬币个数：");

scanf("%d",&n);

printf("\n请输入硬币重量，真币质量大于假币：\n");

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

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

printf("\n");

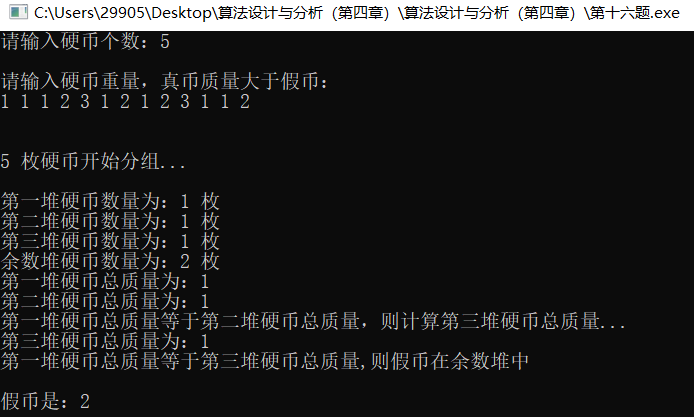
temp=compare(all,a,b,c,d,n);

printf("\n假币是：%d",temp);

return 0;

}

运行截图：



第十七题

代码：

#include<stdio.h>

//#include <algorithm>

#include<stdlib.h>

#define N 6//表示有6项加工任务

//测试数据

//3 8 12 10 5 9 2 6 9 3 11 1

struct node {

int time;

int type;

int id;

}a[N \* 2];// , d[N];

//int d[N] = { 0 };

int visit[N + 1] = { 0 };

int d[N] = { 0 };

struct d {

int fo1;

int fo2;

}p[N + 1];

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

return ((struct node\*)a)->time > ((struct node\*)b)->time;

}

void Initate() {

int flag = 1,id = 1;

printf("请输入各个工程的时间：");

for (int i = 0; i < N \* 2; i++) {

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

a[i].type = i % 2;//表示某个项目的不同工程

a[i].id = i / 2 + 1;//表示项目的id

}

for (int i = 0; i < N \* 2; i++) {

if (i % 2 == 0) {

id = a[i].id;

p[id].fo1 = a[i].time;

}

else {

id = a[i].id;

p[id].fo2 = a[i].time;

}

}

return;

}

void F() {

int s = 0; int r = N - 1;

for (int j = 0; j < N\*2; j++) {

if (visit[a[j].id] != 1) {

if (a[j].type == 0) {

d[s++] = a[j].id;

visit[a[j].id] = 1;

}

else {

d[r--] = a[j].id;

visit[a[j].id] = 1;

}

}

}

}

void Print() {

/\*int sum1 = 0; int sum2 = 0;

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

sum1 += p[d[i]].fo1;

sum2 += p[d[i]].fo2;

}

if (sum1 > sum2) {

printf("最短时间为%d\n", sum1);

}

else {

printf("最短时间为%d\n", sum2);

}

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

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

}\*/

int m1 = p[d[0]].fo1;//m1上的作业时间

int end = m1 + p[d[0]].fo2;

for (int i = 1; i < N; i++) {//其他作业

m1 += p[d[i]].fo1;

end = m1 > end ? m1 + p[d[i]].fo2 : end + p[d[i]].fo2;

}

printf("最短的时间：%d\n", end);

printf("最优的顺序：");

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

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

}

return;

}

int main() {

Initate();//初始化a数组

for (int i = 0; i < N \* 2; i++) {

printf("%d %d %d\n", a[i].id, a[i].time, a[i].type);

}

qsort(a, N\*2, sizeof(a[0]), cmp);

for (int i = 0; i < N \* 2; i++) {

printf("%d %d %d\n", a[i].id, a[i].time, a[i].type);

}

F();

Print();

return 0;

}

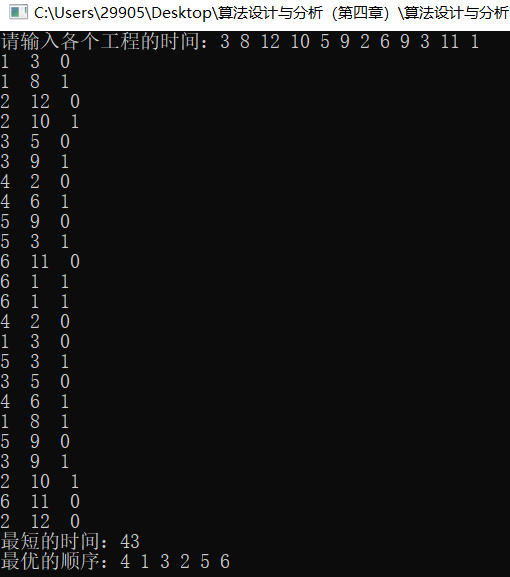
/\*

最短的时间：43

最优的顺序：4 1 3 2 5 6

\*/

运行截图：



第十八题

代码：

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

const int maxn=1000;

const int inf=0x3f3f3f3f;

int n;

char c[maxn];

int dp[maxn][maxn];

int num[maxn][maxn];

int m;

int NUM(int x,int y)

{

if(num[x][y]!=-1)

{

return num[x][y];

}

int sum=0;

for(int i=x;i<y;i++)

{

sum=sum\*10+c[i]-'0';

}

num[x][y]=sum;

return sum;

}

int DP(int p,int x)

{

if(dp[p][x]!=-1)

{

return dp[p][x];

}

if(x==0)

{

dp[p][0]=NUM(0,p);

return dp[p][0];

}

dp[p][x]=inf;

for(int k=p-1;k>=x;k--)

{

dp[p][x]=min(dp[p][x],DP(k,x-1)+NUM(k,p));

}

return dp[p][x];

}

int main() {

while(~scanf("%s",c))

{

memset(dp,-1,sizeof(dp));

memset(num,-1,sizeof(num));

scanf("%d",&m);

n=strlen(c);

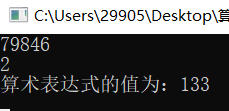
printf("算术表达式的值为：%d\n",DP(n,m));

}

return 0;

}

运行截图：



第十九题

代码：

#include<iostream>

using namespace std;

int dynamic(int m,int n){

int a[n+1][m+1];

for(int i=1;i<=n;i++){ //因为下面的双层for循环需要用到j-1，所以下面的for循环不能从j=0开始，所以在这个地方先给特殊的地方赋值

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

if(j==0 || i==j){

a[i][j]=1;

}

}

}

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

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

if(j==0 || i==j){

a[i][j]=1;

}

else if(j>i){ //特殊情况，当n<m

a[i][j]=0;

}

else{

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

}

}

}

return a[n][m];

}

int main(){

while(true){

int m,n;

cout<<"请输入组合数的m和n（m<n）:"<<endl;

cin>>m>>n;

int p;

p=dynamic(m,n);

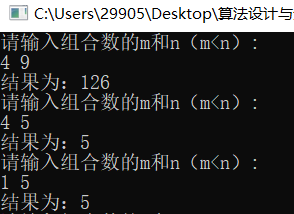
cout<<"结果为：";

cout<<p<<endl;;

}

}

运行截图：



第二十题

代码：

#include <stdio.h>

#define NUM 1001

int a[NUM];

void MaxSum(int n)

{

int begin=0, besti, bestj;

int sum=0, b=0, i, l;

int s1, s2, sum1, sum2, ii, jj;

for (i=0; i<n; i++) //情况i)的解法，动态规划方法

{

if (b>0)

b+=a[i];

else

{

b=a[i];

begin=i;

}

if (b>sum)

{

sum=b;

besti=begin;

bestj=i;

}

}

for(l=1; l<n; l++) //l<n

{

s1=0;

s2=0;

sum1=0;

sum2=0;

for(int i=n-1; i>=l; i--) //情况ii)的解法， 本质就是个蛮力法

{

s1=s1+a[i];

if(s1>sum1)

{

sum1=s1;

ii=i;

}

}

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

{

s2=s2+a[i];

if(s2>sum2)

{

sum2=s2;

jj=i;

}

}

if(sum1+sum2>sum)

{

sum=sum1+sum2; besti=ii; bestj=jj;

}

//printf("%d %d\n", sum, sum1+sum2);

}

printf("最大子段和和对应的起始及结束下标分别是：%d %d %d\n", sum, besti, bestj);

printf("最大子段和序列是：");

if(besti<=bestj)

for(int i=besti; i<=bestj; i++)

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

else

{

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

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

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

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

}

}

int main()

{

int n;

while (scanf("%d", &n) && n)

{

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

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

MaxSum(n);

}

return 0;

}

运行截图：

