2.猴子吃桃问题，猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了两个，第二天早上又将剩下的桃子吃掉一半，又多吃了两个，以后每天早上都吃了前一天剩下的一半零两个，到第10天早上在想吃时就只剩下两个桃子了。问第一天猴子摘下多少个桃子？

**源程序：**

#include<stdio.h>

int main()

{

int x = 2;

for (int i = 9; i >= 1; i--)

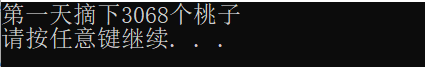
x = (x + 2) \* 2;

printf("第一天摘下%d个桃子\n", x);

return 0;

}

**运行结果:**



6.百马百担问题：有100匹马，驮100担货。大马驮3担，中马驮2担，两匹小马驮1担，问大，中，小各多少？

**源程序：**

#include<stdio.h>

int main(void)

{

int i, j, k;

for (i = 1; i <= 33; i++)

for (j = 1; j <= 50; j++)

{

k = 100 - i - j;

if (k % 2 == 0)

{

if (3 \* i + 2 \* j + k / 2 == 100)

printf("大马%d,中马%d,小马%d\n", i, j, k);

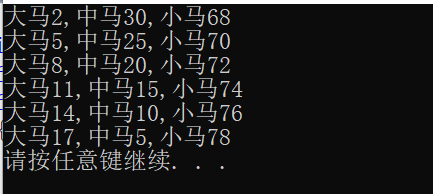
}

}

return 0;

}

**运行结果:**



9.利用分治法求一组数据中最大的两个数和最小的两个数。

**源程序：**

#include<stdlib.h>

#include <stdio.h>

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

void getminmax(int i,int j,int \*min1,int \*min2,int \*max2,int \*max1)

{

int mid;

int lmin1, lmin2, lmax1, lmax2;

int rmin1, rmin2, rmax1, rmax2;

if (i == j)

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

else if (i == j - 1)

if (a[i]<a[j]){

\*max1 = a[j];

\*max2 = a[i];

\*min1 = a[i];

\*min2 = a[j];

}

else{

\*max1 = a[i];

\*max2 = a[j];

\*min1 = a[j];

\*min2 = a[i];

}

else{

mid = (i + j) / 2;

getminmax(i, mid, &lmin1, &lmin2, &lmax2, &lmax1);

getminmax(mid + 1, j, &rmin1, &rmin2, &rmax2, &rmax1);

if (lmin1 < rmin1){

if (lmin2 < rmin1){

\*min1 = lmin1;

\*min2 = lmin2;

}

else{

\*min1 = lmin1;

\*min2 = rmin1;

}

}

else

if (lmin1 > rmin2){

\*min1 = rmin1;

\*min2 = rmin2;

}

else{

\*min1 = rmin1;

\*min2 = lmin1;

}

if (lmax1>rmax1)

if (lmax2>rmax1){

\*max1 = lmax1;

\*max2 = lmax2;

}

else{

\*max1 = lmax1;

\*max2 = rmax1;

}

else

if (rmax2>lmax1){

\*max1 = rmax1;

\*max2 = rmax2;

}

else{

\*max1 = rmax1;

\*max2 = lmax1;

}

}

}

int main()

{

int min1, min2, max1, max2;

int n = sizeof(a) / sizeof(a[0]);

getminmax(0, n - 1, &min1, &min2, &max2, &max1);

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

system("pause");

return 0;

}

**运行结果:**

F:\360MoveData\Users\寇明珠\Documents\Tencent Files\2459971164\FileRecv\MobileFile\Image\Z_O{I_3RL@J7EXTF2E[S_KH.png

11. 某旅游区的街道成网格状（见图），其中东西向的街道都是旅游街，南北向的街道都是林荫道。由于游客众多，旅游街被规定为单行道。游客在旅游街上只能从西向东走，在林荫道上既可以由南向北走，也可以从北向南走。阿隆想到这个旅游区游玩。他的好友阿福给了他一些建议，用分值表示所有旅游街相邻两个路口之间的道路值得浏览得程度，分值从-１００到１００的整数，所有林荫道不打分。所有分值不可能全是负值。

　　例如下图是被打过分的某旅游区的街道图：

　　  -50 -47 36 -30 -23

17 -19 -34 -43 -8

    -42 -3 -43 34 -45

阿隆可以从任一路口开始浏览，在任一路口结束浏览。请你写一个程序，帮助阿隆寻找一条最佳的浏览路线，使得这条路线的所有分值总和最大。

**源程序：**

#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);

system("pause");

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)

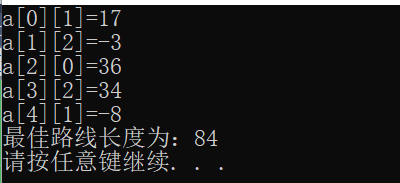
headSum = 0;

}

return maxSum;

}

**运行结果:**



13. 在一个n\*m的放个中，m为奇数，放置有n\*m个数，如下所示。方格中间的下方有一个人，此人可按照5个方向前进但不能越出方格。

人每走过一个方格必须取此方格中的数。要求找到一条从底到顶的路径，使其数相加之和为最大。输出最大和的值。

**源程序：**

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int a[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;

int temp;

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

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

{

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

{

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++)

{

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

{

max = b[i][j];

flag = j;

}

}

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

temp = c[i][flag];

printf("%5d", a[i][temp]); for (j = i; j>0; j--)

{

temp = c[j][temp];

printf("%5d", a[j - 1][temp]);

}

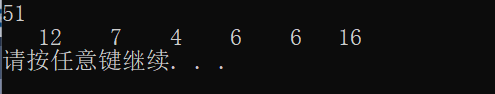
printf("\n");

system("pause");

return 0;

}

**运行结果:**



14.某工业生产部门根据国家计划的安排，拟将某种高效率的5台机器，分配给所属的A，B，C3个工厂各工厂载获得这种机器后，可以为国家盈利如下表所示，问：这5台机器如何分配给各工厂，才能使国家盈利最大？

**源程序：**

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int A[6] = { 0, 3, 7, 9, 12, 13 };

int B[6] = { 0, 5, 10, 11, 11, 11 };

int C[6] = { 0, 4, 6, 11, 12, 12 };

int AB[6][6];

int temp[6];

int abc[6];

int max;

int flag;

int i, j, k;

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

{

max = 0;

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

{

AB[i][j] = A[i - j] + B[j];

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

max = AB[i][j];

}

temp[i] = max;

}

max = 0;

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

{

abc[i] = temp[i] + C[5 - i];

if (abc[i]>max)

{

max = abc[i];

flag = i;

}

}

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

printf("c=%d\n", 5 - flag);

max = max - C[5 - flag];

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

{

if (AB[flag][i] == max)

{

printf("b=%d\n", i);

printf("a=%d\n", flag - i);

break;

}

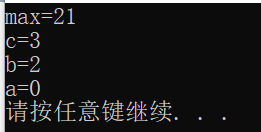
}

system("pause");

return 0;

}

**运行结果:**



17.某一印刷厂有六项加工任务，对印刷车间和装订车间所需时间见下表（时间单位：天）  
        任务 │Ｊ1 Ｊ2 Ｊ3 Ｊ4 Ｊ5 Ｊ6  
       ────┼───────────────  
        印刷车间│３ １２ ５  ２  ９  11  
       装订车间│８ １０ ９  ６  ３  １  
完成每项任务都要先去印刷车间印刷，再到装订车间装订。问如何安排这6项加工任务的加工工序，使加工时间最少。

**源程序：**

#include<stdio.h>

#include<stdlib.h>

int job[6][2] = {{ 3, 8 },{ 12, 10 },{ 5, 9 },{ 2, 6 },{ 9.3 },{ 11, 1 }};

int x[6], bestx[6], f1 = 0, bestf, f2[7] = { 0 };

void try(int i);

void swap(int a, int b);

int main(void)

{

int i, j;

bestf = 32767;

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

x[i] = i;

try(0);

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

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

printf("\nbestf=%d\n", bestf);

system("pause");

return 0;

}

void try(int i)

{

int j;

if (i == 6)

{

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

bestx[j] = x[j];

bestf = f2[i];

}

else

{

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

{

f1 = f1 + job[x[j]][0];

if (f2[i]>f1)

f2[i + 1] = f2[i] + job[x[j]][1]; else

f2[i + 1] = f1 + job[x[j]][1];

if (f2[i + 1]<bestf)

{

swap(i, j);

try(i + 1);

swap(i, j);

}

f1 = f1 - job[x[j]][0];

}

}

}

void swap(int i, int j)

{

int temp;

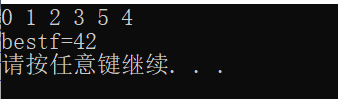
temp = x[i];

x[i] = x[j];

x[j] = temp;

}

**运行结果:**



19.编写用动态规划法求组合数的算法。

**源程序：**

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int buf[100];

int m, n;

int i, j;

buf[0] = 1;

buf[1] = 1;

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

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

{

buf[i + 1] = buf[i];

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

{

buf[j] = buf[j] + buf[j - 1];

}

}

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

system("pause");

return 0;

}

**运行结果:**

