**題目1：**

[2] Sweet Child Makes Trouble

**題目翻譯說明：**

給定一正整數 Ｎ（Ｎ ≦ ８００），代表有Ｎ個物件，編號為１～Ｎ，排在１ ～Ｎ的位置。求重新排列，使得Ｎ個物件都不在自己原來的位置上的方法數為何？當Ｎ＝－１時，輸入結束。

**程式碼片段：**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 800

#define LEN 2000

int p[SIZE + 1][LEN + 1] = {0};

char output[SIZE + 1][LEN + 1];

void create()

{

int i;

p[0][0] = 1;

p[1][0] = 0;

output[0][0] = '1';

output[0][1] = '\0';

output[1][0] = '0';

output[1][1] = '\0';

int j, k, s, count = 0;

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

{

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

{

p[i][j] += (p[i - 1][j] + p[i - 2][j]) \* (i - 1);

p[i][j + 1] += p[i][j] / 10, p[i][j] %= 10;

}

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

if (p[i][j] > 0) break;

for (k = j, s = 0; k >= 0; k --, s ++)

output[i][s] = p[i][k] + '0';

output[i][s] = '\0';

}

}

int main()

{

int n;

create();

while (scanf("%d", &n) == 1 && n != -1)

puts(output[n]);

return 0;

}

**輸出畫面截圖：**



**題目2：**

[3] Polygon

**題目翻譯說明：**

給定半徑r(0<r<20000)以及多邊形數量n，求多邊形的面積。

**程式碼片段：**

#include <stdio.h>

#include <stdlib.h>

#include<math.h>

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

float r;

float n;

float area;

float pi = 3.14159265;

printf("Please input the radius'r' where (0<r<20000)\n");

scanf("%f", &r);

printf("Please input the sides of the polygon 'n' where (2<n<20000)\n");

scanf("%f", &n);

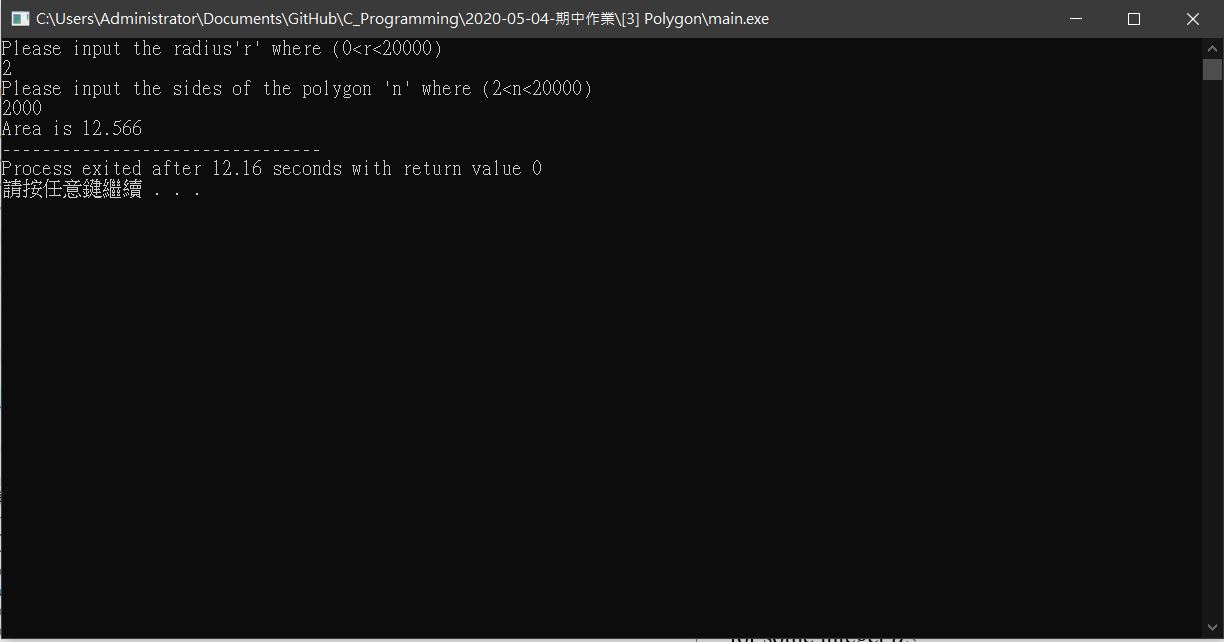
printf("Area is %.3f", 0.5\*sin(2\*pi/n)\*n\*r\*r);

**// 正n邊形面積公式 = 0.5\*sin(2π/n)\*n\*R\*R**

return 0;

}

**輸出畫面截圖：**



**題目3：**

[4] Basic Remains

**題目翻譯說明：**

給三個數，一是進位制，二是被除數，三是除數。求出在此進位制時的餘數為多少。

**程式碼片段：**

#include <stdio.h>

#include <string.h>

#define SIZE 1000

char p[SIZE + 1], m[10], print[SIZE + 1];

int b, r;

int get10Basic()

{

int n = 0, i;

int len = strlen(m);

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

n = n \* b + (m[i] - '0');

return n;

}

void printbBasic(int n)

{

int i = 0, j;

for ( ; n; n /= b)

print[i ++] = n % b;

if (i == 0) printf("0");

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

printf("%c", print[j] + '0');

printf("\n");

}

int main()

{

while (scanf("%d", &b) == 1 && b)

{

scanf("%s %s", p, m);

int len = strlen(p);

int i;

r = get10Basic();

int n = 0;

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

{

n \*= b;

n += p[i] - '0';

n %= r;

}

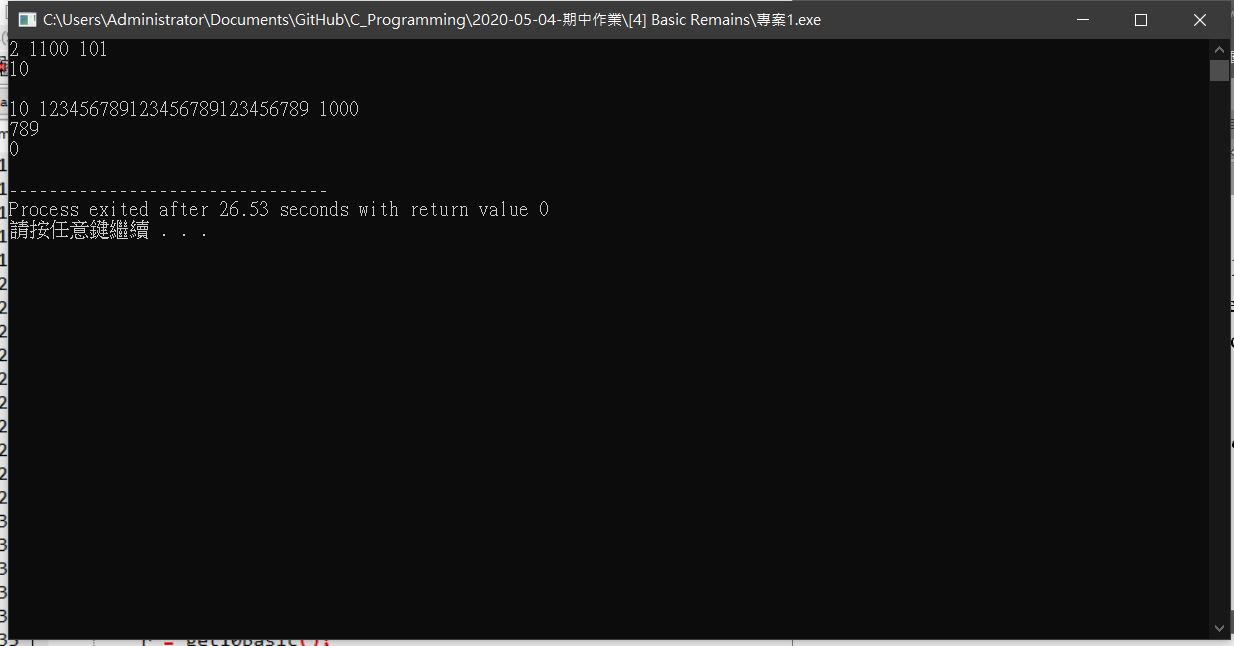
printbBasic(n);

}

return 0;

}

**輸出畫面截圖：**

****

**題目4：**

[5] Geometry Paradox

**題目翻譯說明：**

圖中的2個小圓彼此接觸，而大圓則又與2個小圓接觸。2個小圓的半徑分別為 r1 和 r2，而2小圓間的共同切線於大圓中的長度為 t。我們要算出灰色區域的面積。

**程式碼片段：**

#include <stdio.h>

#include <math.h>

int main() {

int t, a, b;

const double pi = acos(-1);

char cmd[50];

scanf("%d", &t);

gets(cmd);

while(t--) {

gets(cmd);

if(sscanf(cmd, "%d %d", &a, &b) == 2) {

printf("%.4lf\n", 2\*a\*b\*pi);

} else {

sscanf(cmd, "%d", &a);

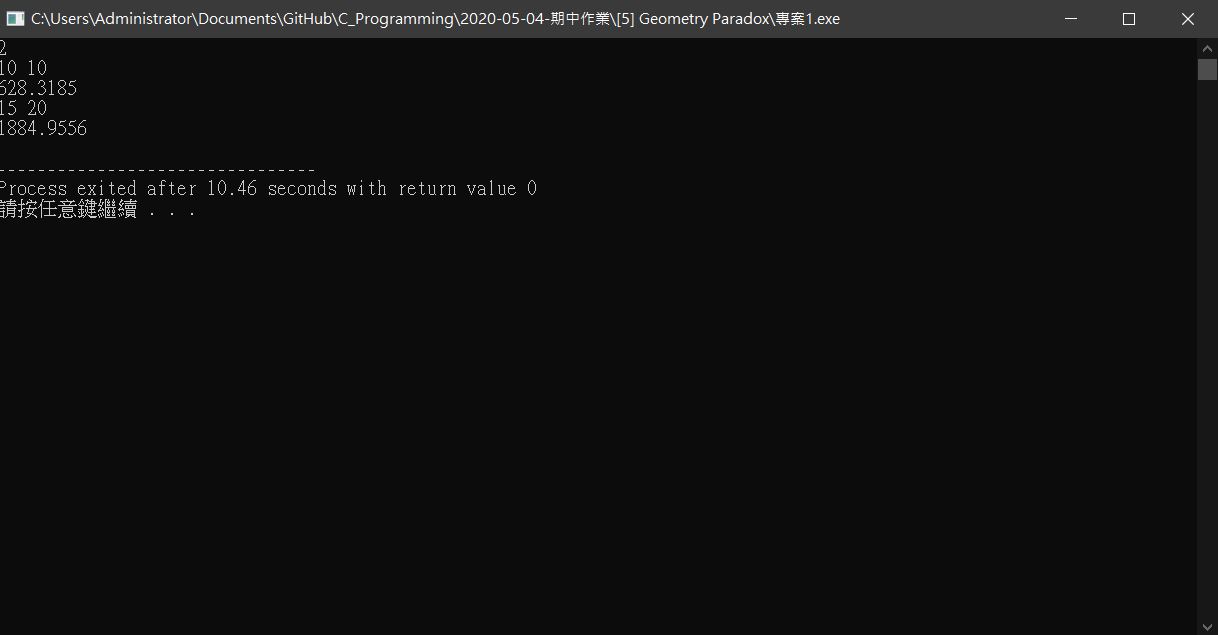
printf("%.4lf\n", a\*a\*pi/8);

}

}

return 0;

}

輸出畫面截圖：

**題目5：**

[7] The jackpot

**題目翻譯說明：**

有一個人想要快速的致富而且不用太辛苦工作，他決定要在以賭場作為生涯職場。所以他開始研究別的賭客輸贏的情況，這樣他才能制訂一個贏的策略。由於他不懂電腦，所以他雇用你寫一個程式來幫助他。

寫一個程式在一連串的賭局中辨識出連續的賭局最大可能贏的錢。每場賭局以一個整數來表示贏或輸的金額。正數代表贏，負數代表輸。例如：有6局賭局金額如下：

-99 10 -9 10 -5 4

連續的賭局最大可能贏的錢為 11，出現在第2,3,4這連續三局中。

**程式碼片段：**

#include <stdio.h>

int main() {

int i;

printf("Input your numbers:\n");

while(scanf("%d", &i) == 1 && i) {

int ans = 0, tmp = 0;

int x;

while(i--) {

scanf("%d", &x);

tmp += x;

if(tmp < 0)

tmp = 0;

if(tmp > ans)

ans = tmp;

}

if(ans)

printf("The maximum winning streak is %d.\n", ans);

else

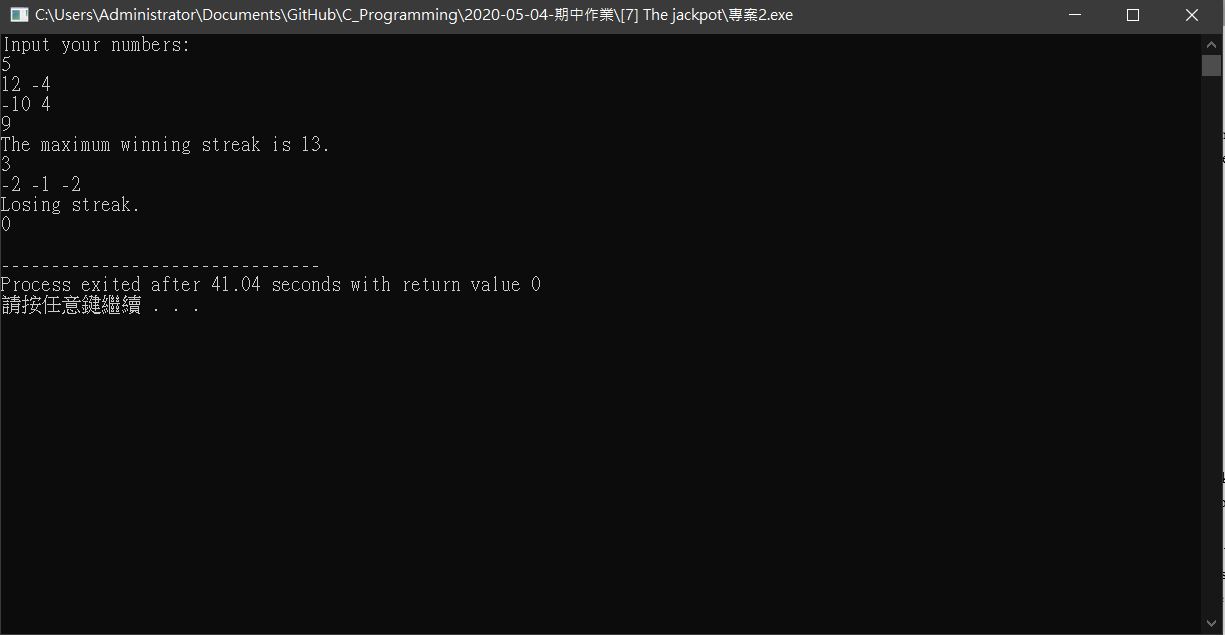
printf("Losing streak.\n");

}

return 0;

}

**輸出畫面截圖：**



**題目6：**

[11] Cheapest Base

**題目翻譯說明：**

列印文字時並不是每個字，都需要同樣的墨水，因此成本也不同。例如'W'，'M'，'8'要比'i'，'c'，'1'來的多。

數字可以被表示成不同的進位制，當我們把數字表示成n進位時（2 <= n <= 36），我們需要用到字串'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'的前 n 項。

每個字元列印時，都有不同的費用，以一個1~128的整數表示，對每一個數字來說，被印出的費用是：所有組成他的數字的費用總和。現在給一個數字，計算用哪種進位制輸出最省錢。

**程式碼片段：**

#include <stdio.h>

#include <ctype.h>

#define SIZE 37

int index, best[SIZE];

int money[SIZE];

int getMoney(int n, int base)

{

int nMoney = 0;

for (; n; n /= base)

{

nMoney += money[n % base];

}

return nMoney;

}

int main()

{

int n, i, m, j, k, min, count, caseNum;

scanf("%d", &n);

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

{

if (caseNum != 1) printf("\n");

index = 0;

for (i = 0; i < 36; i ++)

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

printf("Case %d:\n", caseNum);

scanf("%d", &m);

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

{

scanf("%d", &k);

index = 0;

min = getMoney(k, 2);

best[index ++] = 2;

for (j = 3; j <= 36; j ++)

{

count = getMoney(k, j);

if (min == count) best[index ++] = j;

if (min > count) min = count, index = 0, best[index ++] = j;

}

printf("Cheapest base(s) for number %d:", k);

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

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

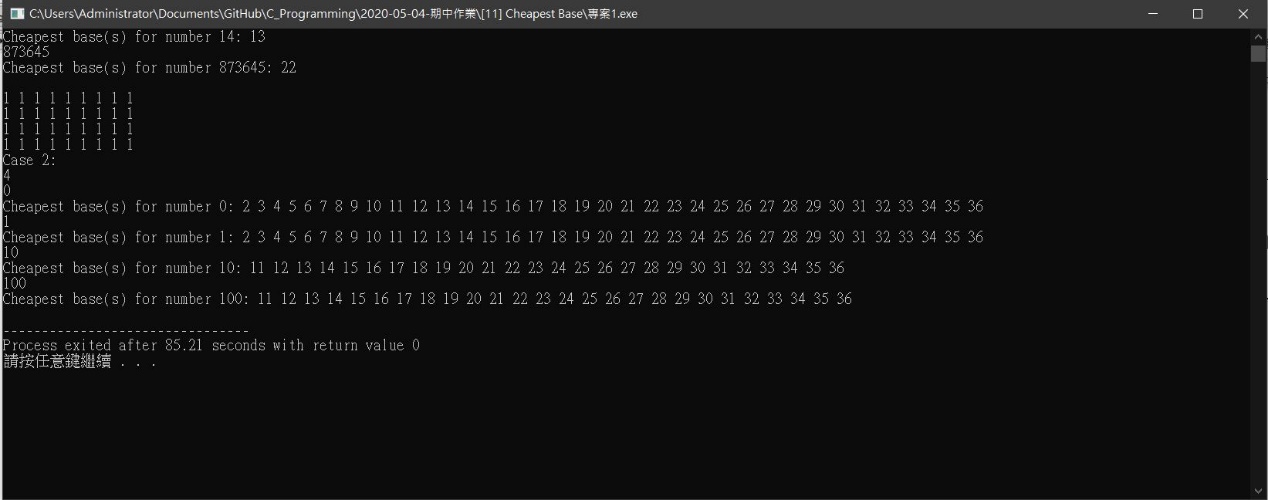
printf("\n");

}

}

return 0;

}

**輸出畫面截圖：**

**題目7：**

[14] Joseph's Cousin

**題目翻譯說明：**

某部落酋長有n個俘虜（編號從1,2,3,……,n），他叫他們排成一個圈圈，然後開始數，第m個人要被煮來吃掉（第一次從編號1的人開始數），按照此規則繼續下去，直到只剩下一個人，那一個人可以保留性命。例如：n=6, m=5則被吃掉的人的編號依序是5,4,6,2,3最號只有編號1活了下來。Joseph是個很聰明的人，他總是能挑到最後存留的位置，所以這件事才被披露出來。在這之後，救了不少被抓的程式設計師。

Joseph's Cousin很討厭程式設計師，所以他寫了一封信介紹了一種新的算法給酋長，以增加計算的難度。在他的算法中 m 不是固定的，而是根據質數的順序（2,3,5,7,11,13,……）產生的。所以當酋長要吃第 i 個人時，他就算第 i 個質數。

你的任務就是要找出最後一個存活者的位置。

**程式碼片段：**

#include <stdio.h>

int Pt = 1, P[5500];

int sieve() {

char mark[50000] = {};

int i, j;

for(i = 2; i < 50000; i++) {

if(!mark[i]) {

P[Pt++] = i;

for(j = 2\*i; j < 50000; j += i)

mark[j] = 1;

}

}

}

int main() {

sieve();

int n, i, j, st[8193];

while(scanf("%d", &n) == 1 && n) {

int M;

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

for(i = 2\*M-1; i > 0; i--) {

if(i >= M)

st[i] = 1;

else

st[i] = st[i<<1]+st[i<<1|1];

}

int m, last, prev = 0, s;

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

m = (P[i]+prev)%(n-i+1);

if(m == 0)

m = n-i+1;

prev = m-1;

for(s = 1; s < M;) {

if(st[s<<1] < m)

m -= st[s<<1], s = s<<1|1;

else

s = s<<1;

}

last = s-M+1;

while(s) {

st[s] --;

s >>= 1;

}

}

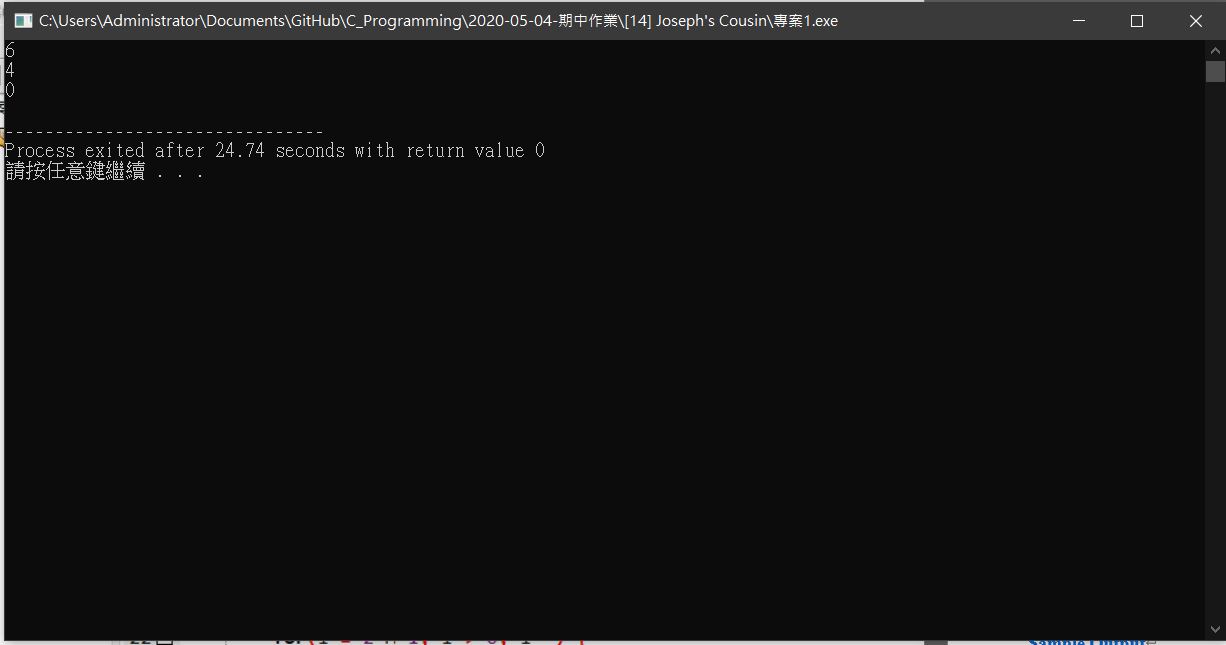
printf("%d\n", last);

}

return 0;

}

**輸出畫面截圖：**



**題目8：**

[15] Page Hopping

**題目翻譯說明：**

最新的報導指出，平均而言只需19次點擊即可從任一網頁連結到www上的任何其他網頁，也就是說，如果把每一個網頁視為圖論中的一個節點，則任兩個節點的平均路徑長度為19。

給定一張圖，可由該圖中任一啟始節點拜訪其他所有節點，你的工作是要從中找出任兩個節點的平均最短路徑長度，以下圖為例，各節點間以有向邊所連接，這表示由 a 節點連到 b 節點的邊，並不表示 b 節點連得到 a 節點。

**程式碼片段：**

#include <stdio.h>

#define min(x, y) ((x) < (y) ? (x) : (y))

int node(int mark[], int \*size, int \*v) {

if(mark[\*v] == 0)

mark[\*v] = ++(\*size);

return mark[\*v];

}

int main() {

int x, y, Case = 0, i, j, k, m;

int map[101][101], in[100000][2];

while(scanf("%d %d", &x, &y) == 2) {

if(x == 0 && y == 0)

break;

int mark[101] = {}, size = 0;

m = 0;

x = node(mark, &size, &x);

y = node(mark, &size, &y);

in[m][0] = x, in[m][1] = y;

m++;

while(scanf("%d %d", &x, &y) == 2) {

if(x == 0 && y == 0)

break;

x = node(mark, &size, &x);

y = node(mark, &size, &y);

in[m][0] = x, in[m][1] = y;

m++;

}

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

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

map[i][j] = 10000;

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

map[in[i][0]][in[i][1]] = 1;

for(k = 1; k <= size; k++)

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

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

map[i][j] = min(map[i][j], map[i][k]+map[k][j]);

int sum = 0;

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

for(j = 1; j <= size; j++) {

if(i != j)

sum += map[i][j];

}

}

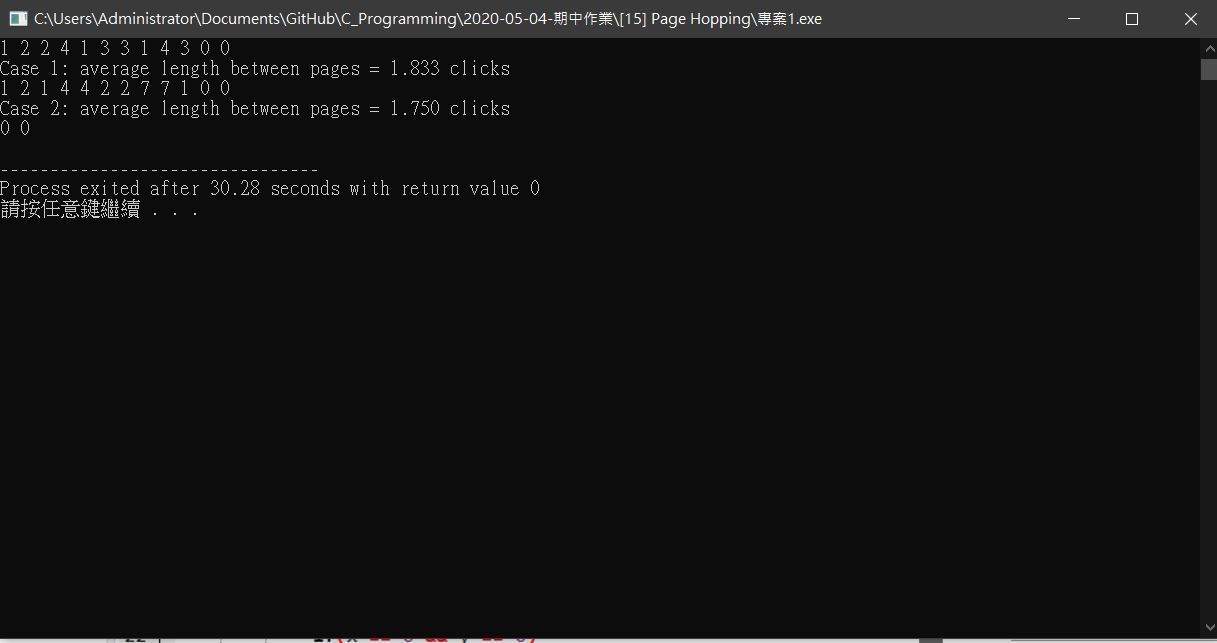
printf("Case %d: average length between pages = %.3lf clicks\n", ++Case, (double)sum/size/(size-1));

}

return 0;

}

**輸出畫面截圖：**

****

**題目9：**

[16] Bee Maja

**題目翻譯說明：**

Maja是一隻蜜蜂，她和幾千隻蜜蜂住在一個蜂巢裡。蜂巢是由正6邊形的小蜂窩所組成的，蜂蜜就存放在裡面。現在Maja有一個問題。另一隻蜜蜂，Willi告訴她在哪裡見面，但是因為Willi是一隻雄蜂，Maja是一隻雌蜂，而他們有不同的座標系統。寫一個程式幫助Maja把Willi的座標轉換成Maja的座標。

Input：每組測試資料1列，有1個整數n（n<100000），代表Willi的座標系統的某一個位置。

Output：輸出對應於Maja座標系統的位置。請參考sample output。

**程式碼片段：**

#include <stdlib.h>

#define SIZE 100000

struct point

{

int x, y;

};

struct point p[SIZE + 1];

int index;

void create()//暴力法建表

{

p[1].x = 0, p[1].y = 0;

index = 2;

int nowI = 0, nowJ = 0, count, num;

for (num = 1; index <= SIZE; num ++)

{

for (count = 0; count < num && index <= SIZE; count ++, index ++)//下

p[index].x = nowI, p[index].y = ++ nowJ;

for (count = 0; count < num - 1 && index <= SIZE; count ++, index ++)//左下 每次都比別人少1 所以第一次沒有

p[index].x = -- nowI, p[index].y = ++ nowJ;

for (count = 0; count < num && index <= SIZE; count ++, index ++)//上

p[index].x = -- nowI, p[index].y = nowJ;

for (count = 0; count < num && index <= SIZE; count ++, index ++)//右上

p[index].x = nowI, p[index].y = -- nowJ;

for (count = 0; count < num && index <= SIZE; count ++, index ++)//右

p[index].x = ++ nowI, p[index].y = -- nowJ;

for (count = 0; count < num && index <= SIZE; count ++, index ++)//右下

p[index].x = ++ nowI, p[index].y = nowJ;

}

}

int main()

{

create();

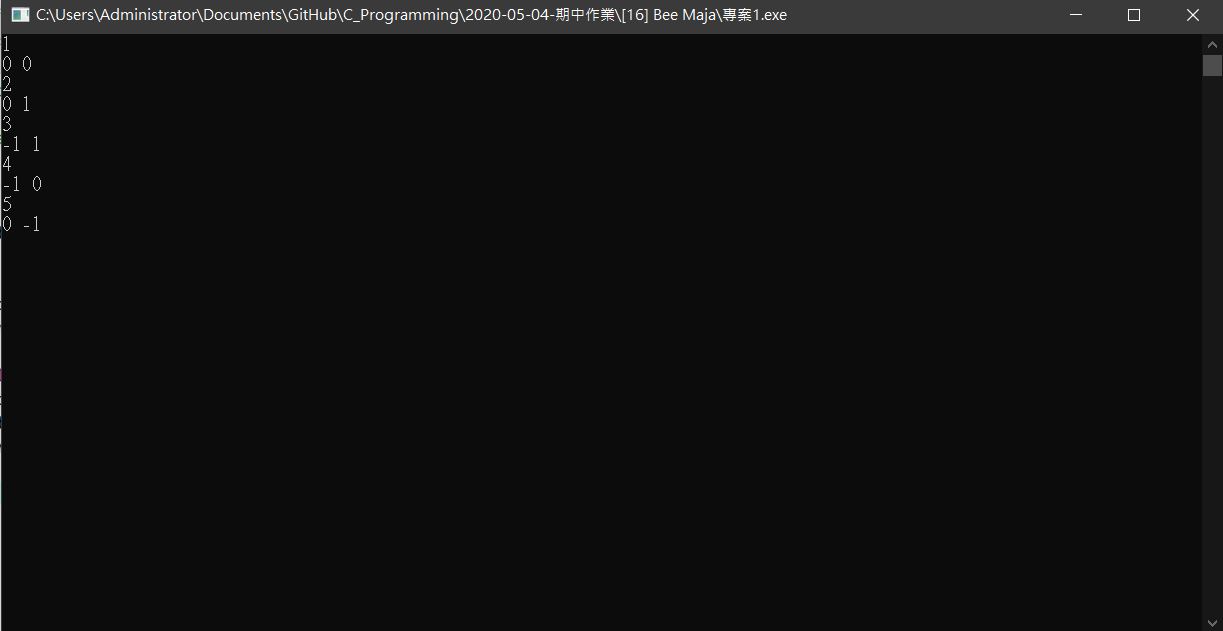
int n;

while (scanf("%d", &n) == 1)

printf("%d %d\n", p[n].x, p[n].y);

}

**輸出畫面截圖：**

****

**題目10：**

[18] Magic Numbers

**題目翻譯說明：**

寫一支程式找出並輸出每一對整數S1及S2符合以下條件：

無論S1或S2都不可以有任何一位數字重複出現﹔

S1 / S2 = N, N為輸入的數目。

**程式碼片段：**

#include <stdio.h>

const long long limit = 9876543210LL;

int Check(long long n) {

char digit[10] = {0}, s[11], i = 0;

sprintf(s, "%lld", n);

while(s[i]) {

digit[s[i]-'0']++;

if(digit[s[i]-'0'] == 2)

return 0;

i++;

}

return 1;

}

int main() {

int T;

long long n, m, i;

scanf("%d", &T);

while(T--) {

scanf("%lld", &n);

for(i = 1; n\*i <= limit; i++) {

m = n\*i;

if(Check(i) && Check(m))

printf("%lld / %lld = %lld\n", m, i, n);

}

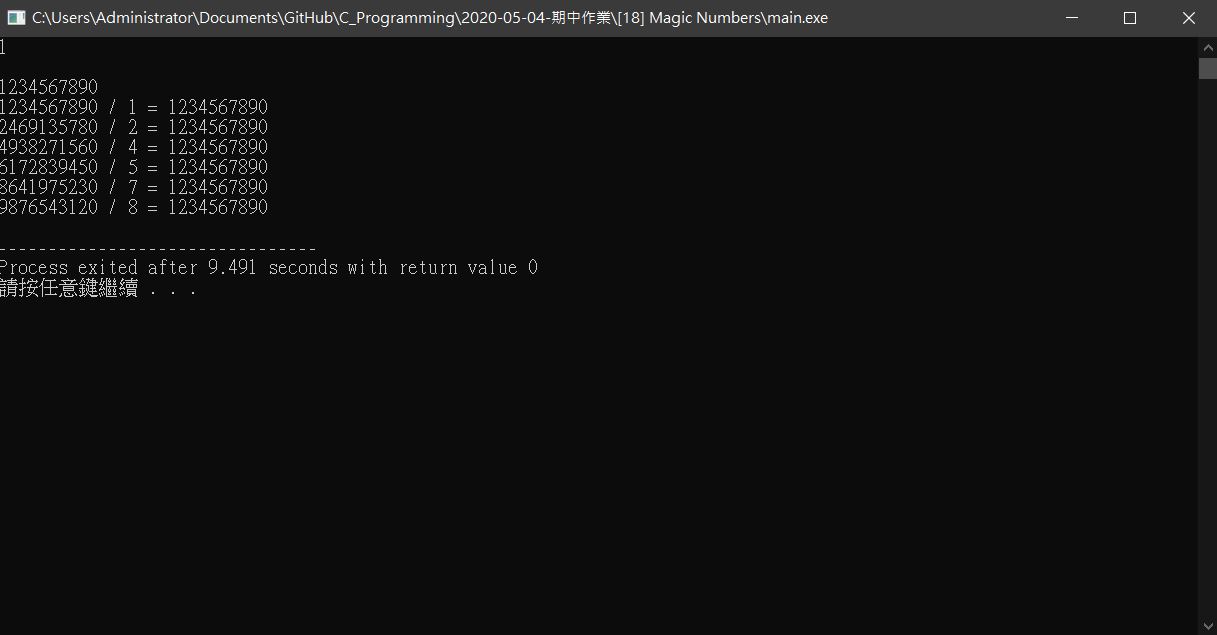
if(T) puts("");

}

return 0;

}

**輸出畫面截圖：**



**題目11：**

[19] Graph Connectivity

**題目翻譯說明：**

一個圖形G是由許多節點及相連的邊所組成。如果在G中的每個節點都可以藉相連的邊，走0步或多步而到達，我們就說G是一個相連（connected）的圖形。給一個圖形，寫一個程式來找出在圖形中有多少個子圖是相連且節點的個數是最大的。

**程式碼片段：**

#include <stdio.h>

int p[26], r[26];

void init(int n) {

while(n >= 0) {

p[n] = n, r[n] = 1;

n--;

}

}

int find(int x) {

return p[x] == x ? x : (p[x]=find(p[x]));

}

int joint(int x, int y) {

x = find(x), y = find(y);

if(x != y) {

if(r[x] > r[y])

r[x] += r[y], p[y] = x;

else

r[y] += r[x], p[x] = y;

return 1;

}

return 0;

}

int main() {

int t;

char s[10];

scanf("%d ", &t);

while(t--) {

gets(s);

init(s[0]-'A');

int ans = s[0]-'A'+1;

while(gets(s)) {

if(s[0] == '\0')

break;

ans -= joint(s[0]-'A', s[1]-'A');

}

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

if(t)

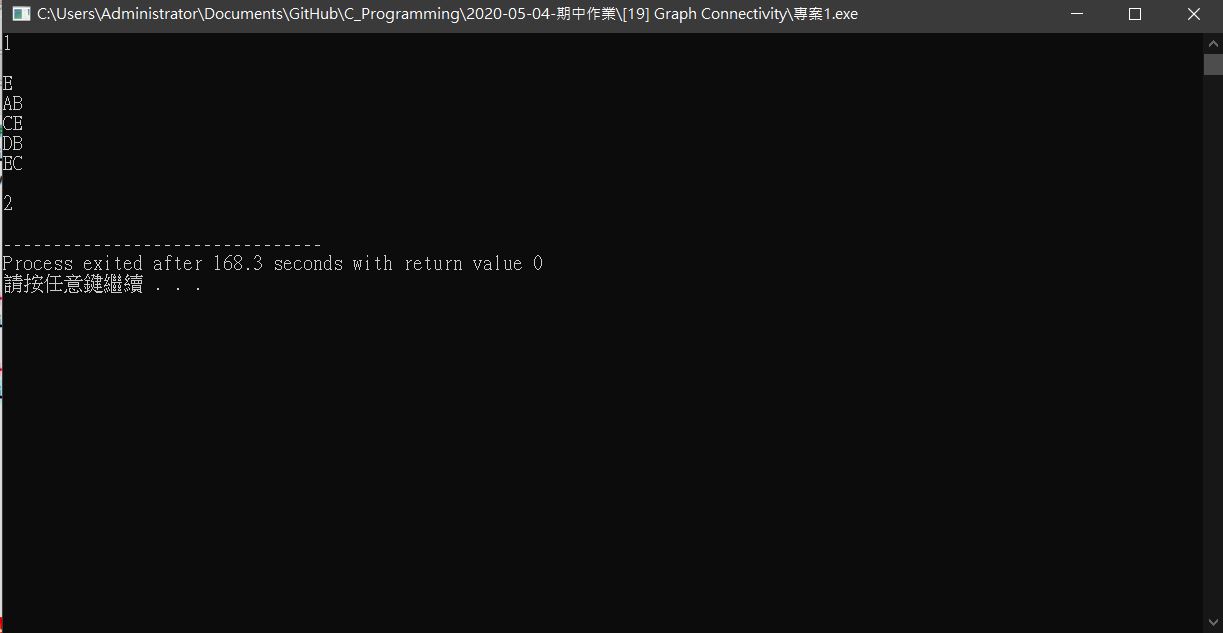
puts("");

}

return 0;

}

**輸出畫面截圖：**



**題目12：**

[20] Humble Numbers

**題目翻譯說明：**

如果一個數字它的質因數只有2,3,5或7則我們稱這個數字為Humble number。下列數字代表前20個Humble numbers：1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 24, 25, 27,...，寫一個程式去找出在這個序列中的第N個元素。

**程式碼片段：**

#include<stdio.h>

int humble[5843];

int min(int a, int b)

{

return a > b ? b : a;

}

void create()

{

int num = 0, p2, p3, p5, p7, n;

humble[1] = p2 = p3 = p5 = p7 = n = 1;

while (1)

{

humble[++ n] = min(2 \* humble[p2],

min(3 \* humble[p3],

min(5 \* humble[p5], 7 \* humble[p7]) ));

if (humble[n] == 2 \* humble[p2]) p2 ++;

if (humble[n] == 3 \* humble[p3]) p3 ++;

if (humble[n] == 5 \* humble[p5]) p5 ++;

if (humble[n] == 7 \* humble[p7]) p7 ++;

if (n == 5842) break;

}

}

void printNumber(int n)

{

int s = n % 10;

int h = n % 100;

printf("The %d", n);

if (h == 11) printf("th");

else if (h == 12) printf("th");

else if (h == 13) printf("th");

else if (s == 1) printf("st");

else if (s == 2) printf("nd");

else if (s == 3) printf("rd");

else printf("th");

printf(" humble number is %d.\n", humble[n]);

}

int main()

{

create();

int n;

while (scanf("%d", &n) == 1 && n)

{

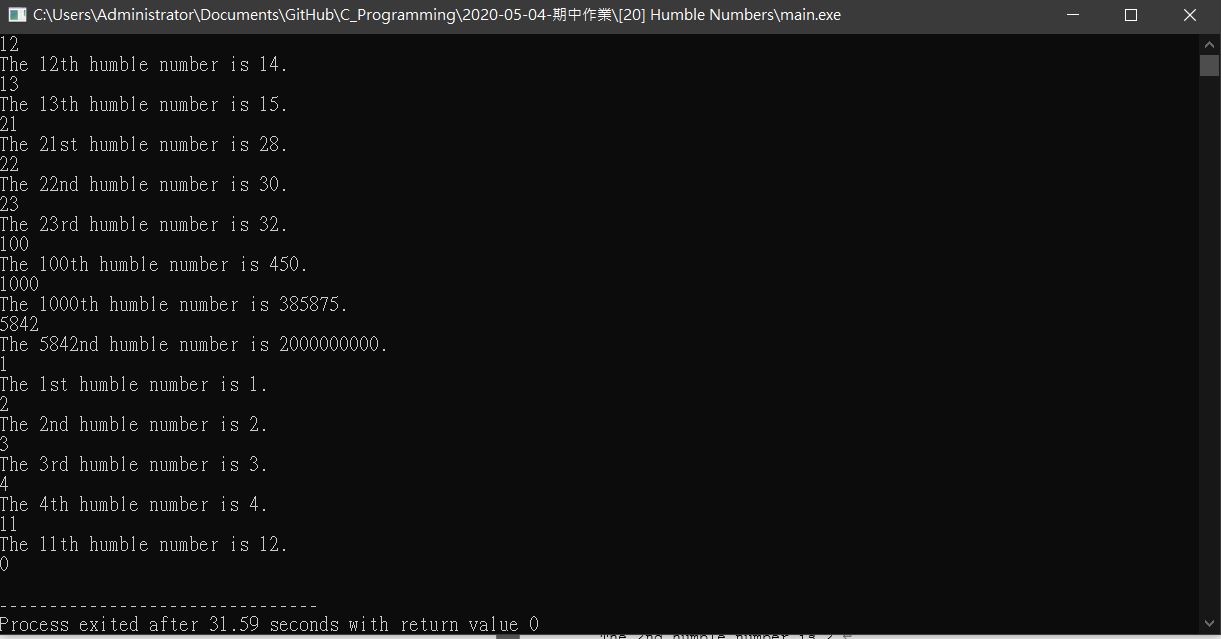
printNumber(n);

}

return 0;

}

**輸出畫面截圖：**

****

**題目13：**

[21] CD

**題目翻譯說明：**

你的空白錄音帶長共N分鐘，你如何選擇CD上的歌使得盡可能的利用錄音帶的空間。以下是一些此問題的假設：

* CD上的歌最多不會超過20首。
* 沒有任何一首歌的長度超過N分鐘。
* 要錄在錄音帶上的歌不能重複。
* 每首歌的長度以一整數表達。
* N也是一個整數。

你的程式必須找出該放哪些CD上的歌到錄音帶上（按CD上的順序），使得錄音帶空白的空間最小。

**程式碼片段：**

#include <stdio.h>

#include <string.h>

int main() {

int N, t, i, j;

int cdTime[20], DP[20001], source[20001];

while(scanf("%d %d", &N, &t) == 2) {

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

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

memset(DP, 0, sizeof(DP));

memset(source, 0, sizeof(source));

DP[0] = 1, source[0] = 0;

for(i = 0; i < t; i++) {

for(j = N-cdTime[i]; j >= 0; j--) {

if(DP[j+cdTime[i]] == 0 && DP[j] == 1) {

DP[j+cdTime[i]] = 1;

source[j+cdTime[i]] = cdTime[i];

}

}

}

int tmpN = N;

while(tmpN >= 0 && DP[tmpN] == 0) tmpN--;

N = tmpN;

int ans[20], aidx = 0;

while(tmpN) {

ans[aidx++] = source[tmpN];

tmpN -= source[tmpN];

}

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

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

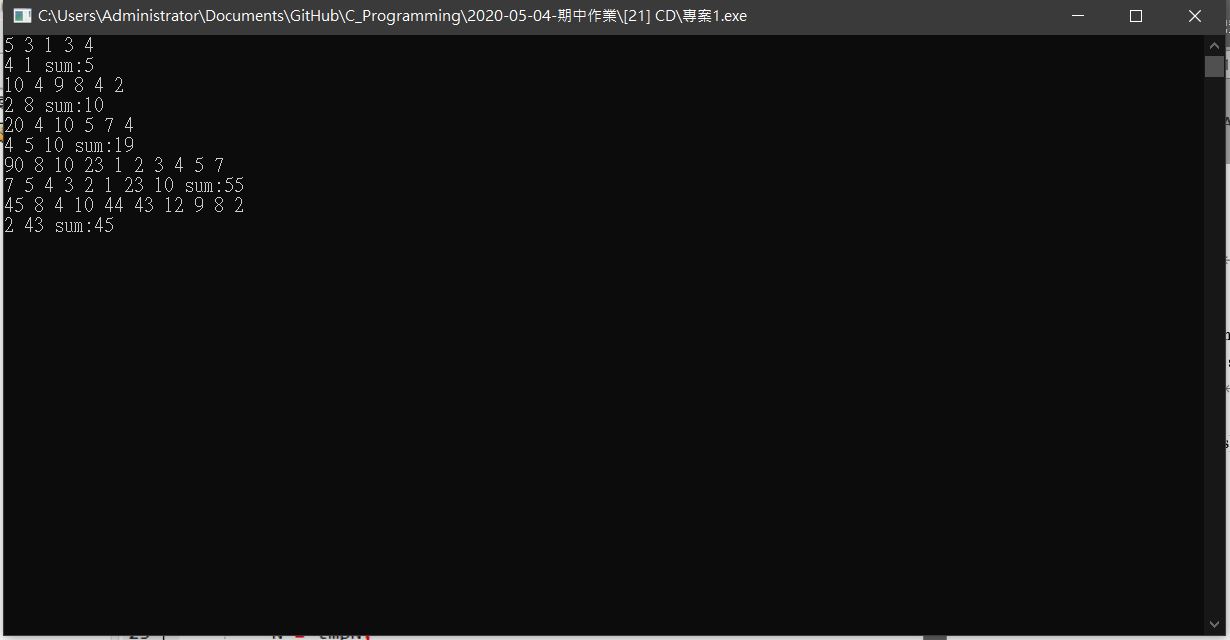
printf("sum:%d\n", N);

}

return 0;

}

**輸出畫面截圖：**

****

**題目14：**

[22] Minesweeper

**題目翻譯說明：**

這個遊戲的目的就是要在M\*N的地雷區格子中找出所有的地雷。在非地雷的格子上有些數字，告訴你這一個格子的鄰居共有多少個地雷。

每組測試資料的第一列有2個整數 n,m（ 0 < n,m <= 100）。分別代表地雷區的寬和長。接下來的n列，每列有m個字元代表地雷區。地雷以 \* 表示，非地雷以 . 表示。當n=m=0代表輸入結束。

**程式碼片段：**

#include <stdio.h>

#include <string.h>

int table[201][201];

void add(int x, int y, int n, int m) {

int i, dx, dy;

int D[8][2] = {{1,0},{0,1},{-1,0},{0,-1},

{1,1},{1,-1},{-1,1},{-1,-1}};

for(i = 0; i < 8; i++) {

dx = D[i][0], dy = D[i][1];

if(x + dx >= 0 && x + dx < n && y + dy >= 0 && y + dy < m) {

table[x+dx][y+dy]++;

}

}

}

int main() {

int n, m, Case = 0, i, j;

char map[201][201];

while(scanf("%d %d", &n, &m) == 2) {

if(n == 0 && m == 0)

break;

if(++Case != 1) puts("");

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

scanf("%s", &map[i]);

memset(table, 0, sizeof(table));

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

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

if(map[i][j] == '\*') {

add(i, j, n, m);

}

printf("Field #%d:\n", Case);

for(i = 0; i < n; i++, puts("")) {

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

if(map[i][j] == '\*')

printf("\*");

else

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

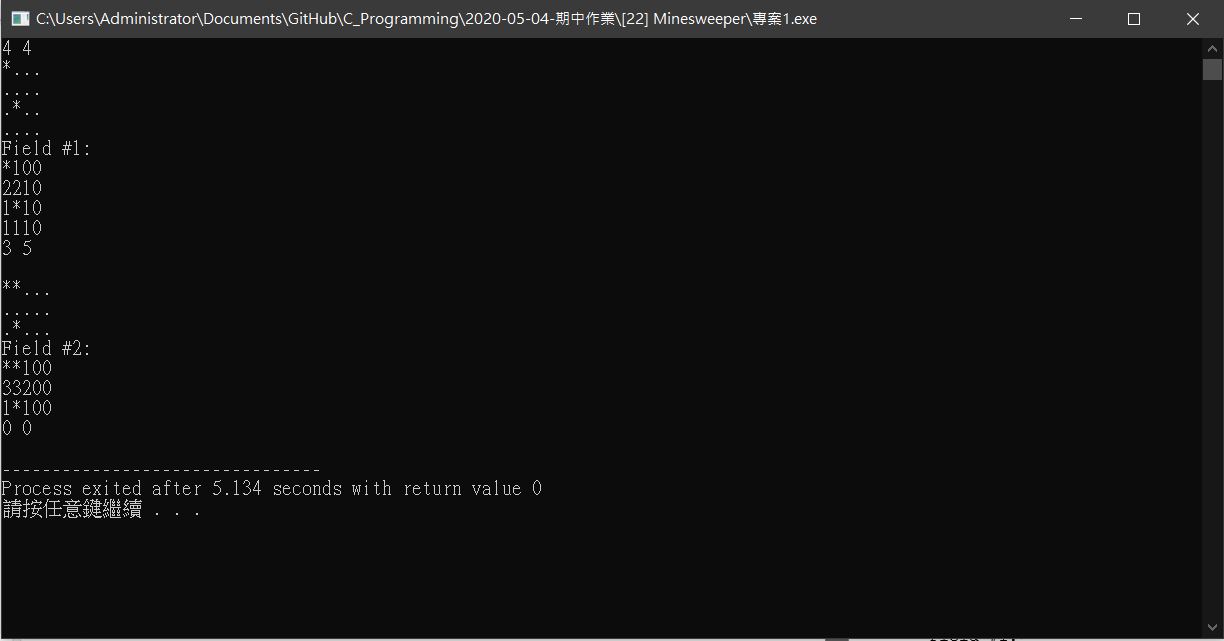
}

}

return 0;

}

**輸出畫面截圖：**



**題目15：**

[23] What's Cryptanalysis

**題目翻譯說明：**

密碼翻譯（cryptanalysis）是指把某個人寫的密文（cryptographic writing）加以分解。這個程序通常會對密文訊息做統計分析。你的任務就是寫一個程式來對密文作簡單的分析。輸入的第1列有一個正整數n，代表以下有多少列需要作分析的密文。接下來的n列，每列含有0或多個字元（可能包含空白字元）  
  
每列包含一個大寫字元（A~Z）和一個正整數。這個正整數代表該字元在輸入中出現的次數。輸入中大小寫（例如：A及a）視為相同的字元。輸出時請按照字元出現的次數由大到小排列，如果有2個以上的字元出現次數相同的話，則按照字元的大小（例如：A在H之前）由小到大排列。

**程式碼片段：**

#include <stdio.h>

#include <ctype.h>

struct Word

{

char ch;

int count;

};

struct Word w[26], c;

int index = 0, repeatIndex ;

int check(char ch)

{

for (repeatIndex = 0; repeatIndex < index; repeatIndex ++)

{

if (w[repeatIndex].ch == ch)

return 0;

}

return 1;

}

void insert(char ch)

{

int i;

if (check(ch))

{

w[index].count = 1, w[index].ch = ch;

index ++;

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

if (w[i].ch < w[i - 1].ch && w[i].count >= w[i - 1].count)

c = w[i], w[i] = w[i - 1], w[i - 1] = c;

else break;

}

else

{

w[repeatIndex].count ++;

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

if (w[i].count > w[i - 1].count ||

w[i].count == w[i - 1].count && w[i].ch < w[i - 1].ch)

c = w[i], w[i] = w[i - 1], w[i - 1] = c;

else break;

}

}

void print()

{

int i;

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

printf("%c %d\n", w[i].ch, w[i].count);

}

int main()

{

int n, i, lineCount = 0;

char ch;

scanf("%d", &n);

getchar();

while (1)

{

ch = getchar();

if (isalpha(ch))

{

ch = toupper(ch);

insert(ch);

}

/\* printf("%c", ch); \*/

if (ch == '\n')

{

lineCount ++;

if (lineCount == n)

{

print();

break;

}

}

}

return 0;

}

**輸出畫面截圖：**

