#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*struct Node //创建成绩链表并删除某个成绩

{

int data;

struct Node \*next;

};

struct Node \*createlist()//创建链表

{

struct Node \*head =(struct Node\*)malloc(sizeof(struct Node));

head->next=NULL;

return head;

}

struct Node \* createnode (int data)//创建新节点

{

struct Node\* newnode=(struct Node\*)malloc(sizeof(struct Node));

newnode->data=data;

newnode->next=NULL;

return newnode;

}

void printlist(struct Node \*head)//遍历链表

{

struct Node \*move=head->next;

while (move)

{

printf("%d\t",move->data);

move=move->next;

}

printf("\n");

}

void insert (struct Node \*head,int data)//尾插法

{

struct Node \*newnode=createnode(data),\*end=head;

while (end->next !=NULL)

{

end= end->next;

}

newnode->next=NULL;

end->next=newnode;

}

struct Node \*deletnode (struct Node \*head,int y)

{

struct Node \*dnode = head->next;

struct Node \*prednode=head;

if (dnode==NULL)

printf("链表为空无法删除");

else

{

while (dnode->data!= y)

{

prednode=dnode;

dnode=dnode->next;

if (dnode==NULL)

{

printf("未找到该元素");

return head;

}

}

prednode->next=dnode->next;

free(dnode);

}

return head;

}

int main()

{

struct Node \* list=createlist();

int m;

printf("请输入人数");

scanf("%d",&m);

int i,x;

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

{

printf("请输入第%d个人的分数",i+1);

scanf("%d",&x);

insert(list,x);

}

printlist(list);

int y;

printf("请输入要删除的分数");

scanf("%d",&y);

list=deletnode(list,y);

printlist(list);

return 0;

}\*/

/\*int fun(char \*s, char \*t, int k)

{

int i;

k = 0;

for (i = 0; s[i]; i++)

{

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

{

t[k] = s[i];

k+= 1;

}

}

return k;

}

void main()

{

char str[100], tOrigin[20];

long n=0;

int i, k=0;

int k1;

printf("Please enter string s:");

scanf("%s", str);

k1=fun(str, tOrigin, k);

for (i = 0; i < k1; i++)

n += (tOrigin [i] - '0') \* pow(10, k1-i-1);

printf("The result is: %ld",n);

}\*/

/\*typedef struct node //链表实现栈内存

{

int data;

struct node \*next;

}NODE;

NODE \*Create();

void Reverse(NODE \*p,int i);

int main()

{

int i=0;

NODE \*head=NULL;

head=Create();

Reverse(head,i);

return 0;

}

NODE \*Create()

{

NODE \*pr=NULL,\*pre=NULL,\*head=NULL;

int i=0;

while(i<5)

{

printf("请输入要压栈的整数：");

pr=(NODE\*)malloc(sizeof(NODE));

scanf("%d",&pr->data);

(head==NULL)?(head=pr):(pre->next=pr);

pre=pr;

printf("Push %dth Data:%d\n",i+1,pr->data);

i++;

}

pr->next=NULL;

return head;

}

void Reverse(NODE \*p,int i)

{

NODE \*pr=NULL;

i++;

if(p->next!=NULL)

{

Reverse(p->next,i);

}

printf("Pop %dth Data:%d\n",i,p->data);

pr=p;

free(pr);

p=p->next;

}\*/

/\*int main(void)

{

int num, num1, op;

char ch;

num = 0;

num1 = 0;

op = '+';

do

{

ch = getchar();

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

{

num1 = num1 \* 10 + (ch - '0');

}

else

{

if (op == '+')

{

num = num + num1;

}

else if (op == '-')

{

num = num - num1;

}

else if (op == '\*')

{

num = num \* num1;

}

else if (op == '/')

{

num = num / num1;

}

else

{

printf("错误的运算符：%c", op);

exit(0);

}

op = ch;

num1 = 0;

}

}

while (ch != '=');

printf("%d", num);

return 0;

}\*/

/\*int main() //23根火柴游戏

{

int match=23;

int pc,player;

printf("这里是23根火柴游戏！！\n");

printf("注意：最大移动火柴数目为三根\n");

while(match!=0)

{

printf("请输入您移动的火柴数目：\n");

scanf("%d",&player);

printf("您移动的火柴数目为：%d\n",player);

match=match-player;

printf("您移动后剩下的火柴数目为：%d\n",match);

if(match==0)

{

printf("对不起！您输了！");

break;

}

if(match==3)

{

pc=2;

}

else

pc=match%3+1;

if(pc==match)

{

pc=pc-1;

}

else if(match==1)

{

pc=1;

}

match=match-pc;

printf("计算机移动的火柴数目为：%d\n",pc);

printf("计算机移动后剩下的火柴数目为：%d\n",match);

if(match==0)

{

printf("恭喜您！您赢了！");

break;

}

}

return 0;

}\*/

/\*typedef struct node

{

int data;

struct node \*next;

} NODE;

NODE \*Create();//逆序输出链表

void Output(NODE \*head);

void Release(NODE \*head);

void Reverse(NODE \*head);

int main()

{

NODE \*head=NULL;

head=Create();

printf("\n原来表：\n");

Output(head);

printf("\n\n反转表：\n");

Reverse(head);

Release(head);

return 0;

}

void Reverse(NODE \*head)

{

if(head->next!=NULL)

{

Reverse(head->next);

}

printf("%4d",head->data);

}

void Output(NODE \*head)

{

NODE \*p=head;

while(p!=NULL)

{

printf("%4d",p->data);

p=p->next;

}

}

void Release(NODE \*head)

{

NODE \*p=head,\*pr=NULL;

while(p!=NULL)

{

pr=p;

p=p->next;

free(pr);

}

}

NODE \*Create()

{

int temp;

NODE \*head = NULL,\*p1=NULL,\*p2=NULL;

printf("\n请输入链表（非数表示结束）\n结点值：");

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

{

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

(head == NULL) ? (head = p1) : (p2->next = p1);

p1->data = temp;

printf("结点值：");

p2 = p1;

}

p2->next = NULL;

return head;

}\*/

/\*int main()

{

NODE \*head=NULL;

head=Create(head);

Output(head);

Reverse(head);

Release(head);

return 0;

}

NODE \*Create(NODE \*head)

{

int i;

NODE \*p=NULL,\*pre=NULL;

p=(NODE\*)malloc(sizeof(NODE));

if(p!=NULL)

{

printf("Input data:\n");

i=scanf(" %d",&p->data);

head=p;

pre=p;

}

else

{

printf("No enough memory!");

exit(0);

}

while(i)

{

p=(NODE\*)malloc(sizeof(NODE));

if(p==NULL)

{

printf("No enough memory!");

exit(0);

}

else

{

printf("Input data:\n");

i=scanf(" %d",&p->data);

pre->next=p;

pre=p;

}

}

p->next=NULL;

return head;

}

void Output(NODE \*head,int n)

{

NODE \*p=head;

while(p!=NULL)

{

printf("%-8d",p->data);

p=p->next;

}

}

void Release(NODE \*head)

{

NODE \*p=head,\*pr=NULL;

while(p!=NULL)

{

pr=p;

p=p->next;

free(pr);

}

}

void Reverse(NODE \*head)

{

if(head->next!=NULL)

{

Reverse(head->next);

}

printf("%d\t",head->data);

}\*/

/\*#define MAX\_LEN 10

#define N 150

void SortString(char \*ptr[], int n);

main()

{

int i, n;

char \*pStr[N];

char name[N][MAX\_LEN];

printf("How many countries?\n");

scanf("%d",&n);

getchar();

printf("Input their names:\n");

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

{

pStr[i]=name[i];

gets(pStr[i]);

}

SortString(pStr,n);

printf("Sorted results:\n");

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

{

puts(pStr[i]);

}

}

void SortString(char \*ptr[], int n)

{

int i, j;

char \*temp=NULL;

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

{

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

{

if (strcmp(ptr[j],ptr[i])<0)

{

temp = ptr[i];

ptr[i] = ptr[j];

ptr[j] = temp;

}

}

}

}\*/

/\*int Fun(char str[]);

int main()

{

int flag=1;

char str[10];

do

{

printf("请输入一个字符串(长度为[%d..%d]个字符)：\n",3,5);

scanf("%s",str);

flag=Fun(str);

}while(flag==0);

printf("你输入的字符串为：%s\n",str);

return 0;

}

int Fun(char str[])

{

int n=strlen(str);

if(n<3 || n>5)

return 0;

else

return 1;

}\*/

/\*#define SIZE sizeof(struct node) //动态链表存成绩改错

struct node

{

long num;

float score;

struct node \*next;

};

int main()

{

int n = 1;

struct node \*head = NULL, \*p1=NULL, \*p2=NULL;

p1 = p2 = (struct node\*)malloc(SIZE);

printf("Input %d node data:\n", n);

scanf("%ld%f", &p1->num, &p1->score);

while (p1->num != 0)

{

if (n == 1)

{

head = p1;

p2=p1;

n++;

}

else

{

p1 = (struct node\*)malloc(SIZE);

p2->next=p1;

p2=p1;

printf("Input %d node data:\n", n);

scanf("%ld%f", &p1->num, &p1->score);

n++;

}

}

printf("Prit list:\n");

p1->next = NULL;

if (head != NULL)

{

p2=head;

while (p2 != NULL && p2->num!=0)

{

printf("num:%ld\tscore:%.2f\n", p2->num, p2->score);

p2 = p2->next;

}

}

return 0;

}\*/

/\*#define N 6

typedef struct node

{

int data;

struct node \*next;

}NODE;

NODE \*Create(NODE \*head,int a[]);

NODE \*Delete(NODE \*head,int x);

NODE \*Find\_Node(NODE \*head,int x);

void Print(NODE \*head);

void Release(NODE \*head);

int main()

{

int i,x;

int a[N];

NODE \*head=NULL;

printf("输入数组%d个元素的值。\n",N);

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

{

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

}

head=Create(head,a);

printf("此链表各个结点的数据域为：");

Print(head);

printf("\n");

printf("输入要删除的数据x: ");

scanf("%d",&x);

head=Delete(head,x);

printf("删除后链表各个结点的数据域为：");

Print(head);

Release(head);

return 0;

}

NODE \*Create(NODE \*head,int a[])

{

int i=0;

NODE \*pre=NULL;

NODE \*pr;

pr=(NODE\*)malloc(sizeof(NODE));

if(pr==NULL)

{

printf("No enough memory!");

exit(0);

}

else

{

head=pr;

pre=pr;

head->data=a[i];

}

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

{

pr=(NODE\*)malloc(sizeof(NODE));

if(pr==NULL)

{

printf("No enough memory!");

exit(0);

}

else

{

pr->data=a[i];

pre->next=pr;

pre=pr;

}

}

pr->next=NULL;

return head;

}

void Print(NODE \*head)

{

NODE \*p=head;

while(p!=NULL)

{

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

p=p->next;

}

}

void Release(NODE \*head)

{

NODE \*pr=head;

NODE \*p=NULL;

while(pr!=NULL)

{

p=pr;

pr=pr->next;

free(p);

}

}

NODE \*Find\_Node(NODE \*head,int x)

{

int i=1;

NODE \*pr=head;

while(pr!=NULL && i<x)

{

pr=pr->next;

i++;

}

return pr;

}

NODE \*Delete(NODE \*head,int x)

{

int i;

NODE \*p=NULL;

NODE \*pr=NULL;

if(head->data==x)

{

p=head->next;

free(head);

return p;

}

pr=head->next;

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

{

if(pr->data==x)

{

p=Find\_Node(head,i-1);

p->next=pr->next;

free(pr);

}

else

pr=pr->next;

}

return head;

}\*/

/\*int Mystrlen(char str[]);

int main()

{

int n;

char str[85];

printf("Enter a string:");

gets(str);

n=Mystrlen(str);

printf("The length of the string is:%d\n",n);

return 0;

}

int Mystrlen(char str[])

{

int n=0;

int i=0;

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

{

n++;

i++;

}

return n;

}\*/

/\*void Calculate(int a[2][3],int b[3][2],int c[2][2]);

void main()

{

int a[2][3];

int b[3][2];

int c[2][2];

int i,j;

printf("Input 2\*3 matrix a:\n");

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

{

for(j=0;j<3;j++)

{

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

}

}

printf("Input 3\*2 matrix b:\n");

for(i=0;i<3;i++)

{

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

{

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

}

}

Calculate(a,b,c);

printf("Results:\n");

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

{

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

{

printf("%6d",c[i][j]);

}

printf("\n");

}

}

void Calculate(int a[2][3],int b[3][2],int c[2][2])

{

int i,j;

int k;

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

{

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

{

c[i][j] = 0 ;

for (k=0; k<3; k++)

{

c[i][j] = c[i][j]+a[i][k]\*b[k][j] ;

}

}

}

}\*/

/\*#define N 6

typedef struct node //链表赋值与删除某个节点

{

int data;

struct node \*next;

}NODE;

NODE \*Create\_Input(NODE \*head,int a[]);

void Print(NODE \*head);

NODE \*Delete(NODE \*head,int x);

void Release(NODE \*head);

NODE \*FindAnode(NODE \*head,int n);

int main()

{

int a[N],i,x;

NODE \*head=NULL;

printf("输入数组%d个元素的值。\n",N);

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

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

head=Create\_Input(head,a);

printf("此链表各个结点的数据域为：");

Print(head);

printf("\n");

printf("输入要删除的数据x: ");

scanf("%d",&x);

head=Delete(head,x);

printf("删除后链表各个结点的数据域为：");

Print(head);

Release(head);

return 0;

}

NODE \*Create\_Input(NODE \*head,int a[])

{

NODE \*pr=NULL;

NODE \*pre=NULL;

int i=0;

pr=(NODE\*)malloc(sizeof(NODE));

if(pr==NULL)

{

printf("No enough memory!");

exit(0);

}

else

{

head=pr;

pre=pr;

head->data=a[i];

}

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

{

pr=(NODE\*)malloc(sizeof(NODE));

if(pr==NULL)

{

printf("No enough memory!");

exit(0);

}

else

{

pr->data=a[i];

pre->next=pr;

pre=pr;

}

}

pr->next=NULL;

return head;

}

void Print(NODE \*head)

{

NODE \*p=head;

while(p!=NULL)

{

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

p=p->next;

}

}

NODE \*Delete(NODE \*head,int x)

{

NODE \*p=NULL;

NODE \*pr=NULL;

int i;

if(x==head->data)

{

p=head->next;

free(head);

return p;

}

p=head;

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

{

if(x==p->data)

{

pr=FindAnode(head,i-1);

pr->next=p->next;

free(p);

}

else

p=p->next;

}

return head;

}

void Release(NODE \*head)

{

NODE \*p=head;

NODE \*pr=NULL;

while(p!=NULL)

{

pr=p;

p=p->next;

free(pr);

}

}

NODE \*FindAnode(NODE \*head,int n)

{

int i=1;

NODE \*p=head;

while(p!=NULL && i<n)

{

p=p->next;

i++;

}

return p;

}\*/

/\*typedef struct node //用数组给链表赋值

{

int data;

struct node \*next;

} NODE;

NODE \*Create\_bianli(NODE \*head,int a[]);

void Print(NODE \*head);

void Release(NODE \*head);

int main()

{

NODE \*head=NULL;

int a[N],i;

printf("输入数组%d个元素的值。\n",N);

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

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

head=Create\_bianli(head,a);

Print(head);

Release(head);

return 0;

}

NODE \*Create\_bianli(NODE \*head,int a[])

{

int i=0;

NODE \*pr=NULL;

NODE \*pre=NULL;

pr=(NODE\*)malloc(sizeof(NODE));

if(pr==NULL)

{

printf("No enough memory!");

exit(0);

}

else

{

head=pr;

pre=pr;

head->data=a[i];

}

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

{

pr=(NODE\*)malloc(sizeof(NODE));

if(pr==NULL)

{

printf("No enough memory!");

exit(0);

}

else

{

pr->data=a[i];

pre->next=pr;

pre=pr;

}

}

pr->next=NULL;

return head;

}

void Print(NODE \*head)

{

NODE \*p=head;

printf("此链表各个结点的数据域为：");

while(p!=NULL)

{

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

p=p->next;

}

}

void Release(NODE \*head)

{

NODE \*p=head;

NODE \*pr=NULL;

while(p!=NULL)

{

pr=p;

p=p->next;

free(pr);

}

}\*/

/\*int Find(char date[7][20],char str[]);//查找星期

int main()

{

char date[7][20]={"sunday","monday","tuesday","wednesday","thursday","friday","saturday"};

char str[20];

int result;

printf("Enter a string:");

scanf("%s",str);

result=Find(date,str);

if(result<0 || result>6)

{

printf("Error\n");

}

else

printf("%s is %d\n",str,result);

return 0;

}

int Find(char date[7][20],char str[])

{

int i;

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

{

if(strcmp(str,date[i])==0)

return i;

}

return -1;

}\*/

/\*void Rank(char str[8][30]);//字符串排序

int main()

{

char str[8][30];

int i;

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

{

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

}

Rank(str);

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

{

printf("%s\n",str[i]);

}

return 0;

}

void Rank(char a[8][30])

{

int i,j;

char term[30];

for(i=0;i<8-1;i++)

{

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

{

if(strcmp(a[i],a[j])>0)

{

strcpy(term,a[i]);

strcpy(a[i],a[j]);

strcpy(a[j],term);

}

}

}

}\*/

/\*#define N 80

void MyStrcat(char \*dstStr, char \*srcStr);

main()

{

char s[N], t[N];

printf("Input a string:\n");

gets(s);

printf("Input another string:\n");

gets(t);

MyStrcat(s, t);

printf("Concatenate results:%s\n", s);

}

void MyStrcat(char \*dstStr, char \*srcStr)

{

while (\*dstStr != '\0')

{

dstStr++;

}

while (\*srcStr != '\0')

{

\*dstStr = \*srcStr;

srcStr++;

dstStr++;

}

\*dstStr='\0';

}\*/

/\*int is\_within(char \*p, char s);

int main()

{

char a[80],s;

int flag;

gets(a);

s = getchar();

flag = is\_within(a, s);

if (flag)

printf("Read the characters in a string.");

else

printf("Read the characters not in the string.");

return 0;

}

int is\_within(char \*p, char s)

{

while(\*p!='\0')

{

if(\*p==s)

return 1;

else

p++;

}

return 0;

}\*/

/\*void Judge(int a[],char str[],char letter[]);

void Rank(int a[],char letter[],int result\_a[],char result\_le[]);

int main()

{

int i,a[26],result\_a[26];

char str[30],letter[30],result\_le[26];

for(i=0;i<26;i++)

{

a[i]=0;

}

for(i=0;;i++)

{

str[i]=getchar();

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

break;

}

str[i]='\0';

Judge(a,str,letter);

Rank(a,letter,result\_a,result\_le);

for(i=0;i<26;i++)

{

printf("%c(%c):%d\n",result\_le[i],result\_le[i]+32,result\_a[i]);

}

}

void Judge(int a[],char str[],char letter[])

{

int j,n=strlen(str);

int k;

char i;

for(i='A',k=0;i<='Z';i++,k++)

{

letter[k]=i;

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

{

if(str[j]==i||str[j]==i+32)

{

a[k]=a[k]+1;

}

}

}

letter[k]='\0';

}

void Rank(int a[],char letter[],int result\_a[],char result\_le[])

{

int i,j=0,k,l,x=0,ta;

char tle,t[26];

for(i=0;i<26;i++)

{

if(a[i]!=0)

{

result\_a[j]=a[i];

result\_le[j]=letter[i];

j++;

}

}

for(i=0,l=0; i<26; i++)

{

if(result\_le[l]==letter[i])

{

l++;

}

else

{

t[x]=letter[i];

x++;

}

}

for(i=0; i<j-1; i++)

{

for(k=1; k<j-1; k++)

{

if(result\_a[k]>result\_a[k-1])

{

ta=result\_a[k];

result\_a[k]=result\_a[k-1];

result\_a[k-1]=ta;

tle=result\_le[k];

result\_le[k]=result\_le[k-1];

result\_le[k-1]=tle;

}

}

}

x=0;

for(i=j;i<26;i++)

{

result\_a[i]=0;

result\_le[i]=t[x];

x++;

}

}\*/

/\*void Inverse(char \*str1);//逆序存放字符串

int main()

{

char str[40];

printf("Please Enter String1:\n");

gets(str);

Inverse(str);

printf("Result is:\n%s\n",str);

return 0;

}

void Inverse(char \*str1)

{

int len;

char temp;

char \*pStart;

char \*pEnd;

len = strlen(str1);

for (pStart = str1, pEnd = s + len - 1; pStart < pEnd; pStart++, pEnd--)

{

temp = \*pStart;

\*pStart = \*pEnd;

\*pEnd = temp;

}

}\*/

/\*int Insert(char str1[],char str2[],char ch);//插入字符串

int main()

{

char str1[50],str2[50],ch;

int flag;

printf("Input source string 1:\n");

gets(str1);

printf("Input inserted string 2:\n");

gets(str2);

printf("Input a letter to locate the index:\n");

scanf(" %c",&ch);

flag=Insert(str1,str2,ch);

if(flag)

{

printf("The new string is:%s",str1);

}

else

{

printf("Not found!");

}

return 0;

}

int Insert(char str1[],char str2[],char ch)

{

int i=0,j,k,l,m,n;

m=strlen(str1);

n=strlen(str2);

while(str1[i]!='\0')

{

if(str1[i]==ch)

{

str1[m+n]='\0';

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

{

str1[k+n]=str1[k];

}

for(j=0,l=i;j<n;j++)

{

str1[l]=str2[j];

l++;

}

return 1;

}

i++;

}

return 0;

}\*/

/\*void InputScore(int a[5][4]);

int Sum(int a[5][4],int i);

void Print(int a[5][4],char ch[5],int sum[],int aver[]);

int main()

{

int score[5][4],sum[5],aver[5];

int i;

char ch[5];

InputScore(score);

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

{

sum[i]=Sum(score,i);

aver[i]=sum[i]/3;

if(aver[i]>90)

{

ch[i]='Y';

}

else

{

ch[i]='N';

}

}

Print(score,ch,sum,aver);

return 0;

}

void InputScore(int a[5][4])

{

int i,j;

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

{

printf("Enter No. and score as: MT EN PH\n");

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

{

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

}

}

}

int Sum(int a[5][4],int i)

{

int j,sum=0;

for(j=1;j<4;j++)

{

sum+=a[i][j];

}

return sum;

}

void Print(int a[5][4],char ch[],int sum[],int aver[])

{

int i,j;

printf("NO\tMT\tEN\tPH\tSUM\tV\t>90\n"

"------------------------------------------------------\n");

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

{

j=0;

printf("%d\t%d\t%d\t%d\t%d\t%d\t%c\n",a[i][j],a[i][j+1],a[i][j+2],a[i][j+3],sum[i],aver[i],ch[i]);

}

}\*/

/\*void Input(int \*p,int r,int c);//求鞍数

int Find(int \*p,int r,int c);

void Print(int \*p,int r,int c,int pr,int pc,int result);

int main()

{

int r,c,\*p=NULL;

int result,pr,pc,i,j,temp;

printf("\n输入行数：");

scanf("%d",&r);

printf("\n输入列数：");

scanf("%d",&c);

p=(int\*)calloc(r\*c,sizeof(int));

Input(p,r,c);

result=Find(p,r,c);

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

{

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

{

temp=p[i\*c+j];

if(temp==p[result])

{

pr=i;

pc=j;

}

}

}

Print(p,r,c,pr,pc,result);

free(p);

return 0;

}

void Input(int \*p,int r,int c)

{

int i,j;

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

{

printf("第%d行？\n",i);

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

{

scanf("%d",&p[i\*c+j]);

}

}

}

int Find(int \*p,int r,int c)

{

int an,i,j,k,pos;

an=p[0];

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

{

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

{

if(an<p[i\*c+j])

{

an=p[i\*c+j];

pos=i\*c+j;

}

}

for(k=i+1; k<r; k++)

{

if(p[k\*c+j]>an)

return pos;

}

}

return -1;

}

void Print(int \*p,int r,int c,int pr,int pc,int result)

{

int i,j;

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

{

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

{

printf("%5d",p[i\*c+j]);

}

printf("\n");

}

if(result<0)

printf("\n矩阵中无鞍点!\n");

else

printf("\n第%d行，第%d列的%d是鞍点\n",pr,pc,p[pr\*c+pc]);

}\*/