**例题 1.1**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

    int n=0;

    int i,j=0;

    int temp=0;

    int height[100]={0};

    int weight[100]={0};

    int num[100]={0};

    while(scanf("%d",&n)!=EOF)

    {

       for(int i=0;i<n;i++)

        {

            scanf("%d",&height[i]);

            num[i]=i+1;

           // printf("%d ",num[i]);

        }

        printf("\n");

        for(int i=0;i<n;i++)

        {

            scanf("%d",&weight[i]);

           // num[i]=i+1;

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

        }

        for(i=0;i<n-1;i++)

        {

            for(j=i+1;j<n;j++)

            {

                if(height[i]>height[j])

                {

                    temp=num[i];

                    num[i]=num[j];

                    num[j]=temp;

                }

                else if(height[i]==height[j]&&weight[i]>weight[j])

                {

                    temp=num[i];

                    num[i]=num[j];

                    num[j]=temp;

                }

            }

        }

       // printf("%d ",num[i]);

        for(i=0;i<n;i++)

        {

            printf("%d ",num[i]);

        }

        printf("\n");

    }

    return 0;

}

**例题 1.2**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

int main () {

    int num =0;

    char ch;

    int clip\_num=0;

    bool choose\_all=false;

    while((ch=getchar())!='\n')

    {

        if(ch-'0'==1)  //输出a

        {

            if (choose\_all==true)

            {

                num =1;

                choose\_all=false;

            }

            else

             {

                 num++;

                choose\_all=false;

             }

        }

        else if(ch-'0'==2)//ctrl+c

        {

            // 若 Ctrl+c 在 Ctrl+a后面 ，要不要去全选

           if (choose\_all==true)

           {

               clip\_num=num;

               //choose\_all=false;

           }

           else

           {

               clip\_num=num;

              // choose\_all=false;

           }

        }

        else if(ch-'0'==3)//ctrl+x

        {

            if (choose\_all==false)

            {}

            else

            {

               clip\_num=num;

                num =0;

               choose\_all=false;

            }

        }

        else if(ch-'0'==4)//ctrl+v

        {

          if (choose\_all==true)

           {

              num=0;

              num = clip\_num;

              choose\_all=false;

           }

            else

            {

            num=num+clip\_num;

                choose\_all=false;

            }

        }

        else if(ch-'0'==5)//ctrl+a

        {

               choose\_all=true;

        }

    }

    printf("%d\n",num);

    return 0;

}

**例题 1.3**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

int main ()

{

    int a[4]={0};

    int temp=0;

    int dif=0;

    int b[9]={0};

    scanf("%d,%d,%d,%d",&a[0],&a[1],&a[2],&a[3]);

    for(int i=0;i<4;i++)

        for(int j=i+1;j<4;j++)

        {

            if(a[i]>a[j])

            {

                temp = a[i];

                a[i]=a[j];

                a[j]=temp;

            }

        }

    b[0]=a[0];

    b[1]=a[1];

    b[2]=a[2];

    b[3]=a[3];

    b[4]=a[0]\*10+a[1];

    b[5]=a[0]\*10+a[2];

    b[6]=a[0]\*10+a[3];

    b[7]=a[1]\*10+a[0];

    b[8]=a[1]\*10+a[1];

    printf("%d\n",b[a[3]-1]);

    return 0;

}

**例题 2.1**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

//#include <string.h>

struct num  //结构体的使用

    {

         int height;

         int diff;

    };

int main ()

{

    int a,b=0;

    struct num tt[50];

    int diff1=0;

    int diff2=0;

    int temp=0;

     struct num temp2;

    memset(tt,0,sizeof(tt));

    memset(&temp2,0,sizeof(struct num)); //memset的用法

    scanf("%d %d",&a,&b);

    //printf("%d  %d\n",a,b);

    for(int i=0;i<b;i++)

    {

        scanf("%d",&temp);

        tt[i].height=temp;

         if(temp>a)

        {

            tt[i].diff=temp-a;

        }

        else

        {

            tt[i].diff=a-temp;

        }

    }

    // printf("\n ");

    for(int i=0;i<b;i++)

    {

        for(int j=i+1;j<b;j++)

        {

            if(tt[i].diff>tt[j].diff)

            {

             memcpy(&temp2,&tt[i],sizeof(struct num));//memcpy的用法

            memcpy(&tt[i],&tt[j],sizeof(struct num));

            memcpy(&tt[j],&temp2,sizeof(struct num));

           }

        }

    }

    for(int i=0;i<b;i++)

    {

        printf("%d ",tt[i].height);

    }

    return 0；

}

**例题2.2**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

int method\_num(int N)

{

    if(N==1)

    { return 1;

    }

    else if(N==2)

    {

        return 1;

    }

    else if(N==3)

    {

        return 2;

    }

    else if(N>3)

    {

        return method\_num(N-1)+method\_num(N-3);

    }

    else

        return 0;

}

int main()

{

    int n=0;

    scanf("%d",&n);

    printf("%d\n",method\_num(n));

    return 0;

}

**例题2.3**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int is\_yuanyin(char c)

{

    if(c=='a'||c=='e'||c=='i'||c=='o'||c=='u'||

      c=='A'||c=='E'||c=='I'||c=='O'||c=='U')

    {

        return 1;

    }

    else

        return 0;

}

int main()

{

    int num=0;

    char alp[65535]={0};

    int len=0;

    int count=0;

    int flag=0;

    int max=0;

    scanf("%d",&num);

    scanf("%s",alp);

    len=strlen(alp);

      for(int i=0;i<len;i++)

    {

        count=0;

        if(is\_yuanyin(alp[i]))

        {

             for(int j=i+1;j<len;j++)

            {

                if(0==is\_yuanyin(alp[j]))

                {

                    count++;

                }

                if(count==num+1)

                {

                    flag=j;

                    break;

                }

            }

            if(flag-i>count)

            {

               if(is\_yuanyin(alp[flag-1]))

                    {

                          if(flag-i>max)

                           {

                              max=flag-i;

                            }

                    }

            }

        }

    }

    printf("%d\n",max);

    return 0;

}

**例题3.1**  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct

{

    char s[10];

    int size;

    int cnt;

}DataInfo;

Void strSqit(DataInfo memInfo[1024],int num)

{

    char tmp[2][10];

    for(int k=0;k<num;k++)

    {

        int i=0;

        char \*p1=NULL;

        p1=strtok(memInfo[k].s,":");

        while(p1!=NULL)

        {

            strcpy(tmp[i],p1);

            i++;

            p1=strtok(NULL,",");

        }

        memInfo[k].size= atoi(tmp[0]);

        memInfo[k].cnt = atoi(tmp[1]);

    }

}

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

{

    DataInfo\*x = (DataInfo\*)a;

    DataInfo \*y=(DataInfo\*)b;

    return x->size-y->size;

}

int main()

{

    DataInfo memInfo[1024];

    int i=0;

    int j=0;

    char str1[100000]={0};

    char str2[100000]={0};

    int list[1024];

    gets(str1);

    gets(str2);

    char \*p1=NULL;

    char \*p2=NULL;

    p1= strtok(str1,",");

    while(p1!=NULL)

    {

        strcpy(memInfo[i].s,p1);

        i++;

        p1=strtok(NULL,",");

    }

    strSqit(memInfo,i);

    p2=strtok(str2,",");

    while(p2!=NULL)

    {

        list[j]=atoi(p2);

        j++;

        p2=strtok(NULL,",");

    }

    qsort(memInfo,i,sizeof(DataInfo),cmp);

    for(int m=0;m<j;m++)

    {

        int flag=0;

        for(int n=0;n<i;n++)

        {

            if(memInfo[n].cnt<1)

            {

                continue;

            }

            if(memInfo[n].size>list[m])

            {

                flag=1;

                memInfo[n].cnt--;

                break;

            }

        }

        if(flag==1)

        {

            printf("true");

        }

        else

        {

            printf("false");

        }

        if(m<j-1)

        {

            printf(",");

        }

        else

        {

            printf("\n");

        }

    }

    return 0;

}

**例题 3.2**  
#include<stdio.h>

#include<stdlib.h>

#include <string.h>

int main()

{

    char str[100]={0};

    int alp[26]={0};

    int i=0;

    int j=0;

    int min=26;

    while(scanf("%s",&str)!=EOF)

    {

         int min=26;

        int len =strlen(str);

        for(i=0;i<len;i++)

        {

            alp[str[i]-'a']++;

        }

        for(i=0;i<26;i++)

        {

            if(alp[i]&&alp[i]<min)

            {

                min=alp[i];

            }

        }

        for(i=0;i<len;i++)

        {

            if(alp[str[i]-'a']>min)

            {

                printf("%c",str[i]);

            }

        }

        printf("\n");

    }

    return 0;

}

**例题 3.3**  
#include<stdio.h>

#include<stdlib.h>

#include <string.h>

void path(int room[1000][1000],int nextX,int nextY,int x,int y)

{

    if(room[nextX][nextY]==1)

    {

        return;

    }

    if(room[nextX][nextY]!=0)

    {

        return;

    }

    if(nextX==x&&nextY==y)

    {

        room[nextX][nextY]=2;

        return;

    }

    if(nextX<x)

    {

        path(room,nextX+1,nextY,x,y);

    }

    if(nextY<y)

    {

        path(room,nextX,nextY+1,x,y);

    }

    //该点向上/向前均为不可达/陷阱方格则为陷阱方格

        if(nextX == x || nextY == y)

        {

            if(nextX == x && nextY < y && room[nextX][nextY+1] != 2)

            {

                room[nextX][nextY] = 9;

            }else if(nextY == y && nextX < x && room[nextX+1][nextY] != 2)

            {

                room[nextX][nextY] = 9;

            }

            else

            {

                room[nextX][nextY] = 2;

            }

        }

        else if(room[nextX+1][nextY] !=2 && room[nextX][nextY+1] !=2)

        {

            room[nextX][nextY] = 9;

        }

        else

        {

            room[nextX][nextY] = 2;

        }

        return;

}

int main()

{

    int width=0;

    int height=0;

    int a=0;

    int b=0;

    int n=0;

    int room[1000][1000];

    int i=0;

    while(scanf("%d %d",&width,&height)!=EOF)

    {

        scanf("%d",&n);

        for(i=0;i<n;i++)

        {

            scanf("%d %d",&a,&b);

            room[a][b]=1;

        }

        path(room,0,0,width-1,height-1);

        int badPath=0;

        int noWay=0;

        for(int i =0; i< width; i++)

        {

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

            {

                if(room[i][j] == 9)

                {

                    badPath += 1;

                }else if(room[i][j] == 0)

                {

                    noWay += 1;

                }

            }

        }

        printf("%d %d",badPath,noWay);

    }

    return 0;

}

**例题 4.1**

#include"stdio.h"

#include"math.h"

int main()

{

    int n,i,j,a,m,w,k;

    scanf("%d",&n);

    printf("%d=%d\n",n,n);

    for(i=n/2;i>=1;i--)

    {

        a=i;

        w=(2\*a-1)\*(2\*a-1)+8\*n;

        k=(int)sqrt(w);

        m=k-2\*a+1;

        if(k\*k!=w)

            continue;

        else if(m%2!=0)

            continue;

        else

        {

            printf("%d=",n);

            for(j=1;j<=m/2;j++)

            {

                if(1!=j)

                {

                    printf("+");

                }

                printf("%d",i+j-1);

            }

            printf("\n");

        }

    }

    return 0;

}

**例题4.2**

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int a=0;

    int num=0;

    int b[1000];

    int i=0;

    int res=0;

    int count=0;

    int a1=0;

    scanf("%d",&a);

    scanf("%d",&num);

    printf("%d num%d\n",a,num);

    for(i=0;i<num;i++)

    {

        scanf("%d",&b[i]);

    }

    i=0;

    res=0;

    for(i=0;i<num;i++)

    {

        //printf("i:%d\n",i);

        if(b[i]+res<=a)

         {

            count=count+1;

        }

        else

        {

            count=count+(b[i]+res)/a;

            res=(b[i]+res)%a;

        }

       }

    if(res!=0)

        count++;

    printf("%d\n",count);

    return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void str\_split(char \*str,char \*a)

{

    char \*p1=NULL;

    int i=0;

    if(NULL==str)

    {}

    p1=strtok(str," ");

    while(p1!=NULL)

    {

        strcpy(a[i],p1);

        printf("%c %c\n",p1,a[i]);

        i++;

        p1=strtok(NULL," ");

    }

}

int main()

{

   char str[100]={0};

    char temp\_str[1000]={0};

    int step[100];

    int a[100][100];

    int m[2];

    int i,j=0;

    int loc\_x,loc\_y=0;

    char \*p1=NULL;

    char \*p2=NULL;

    memset(a,0,sizeof(a));

    //scanf("%s",&str);

    gets(str);

    printf("%s\n",str);

    p1=strtok(temp\_str," ");

    while(p1!=NULL)

    {

       if(0==strcmp(p1,"U"))

       { step[j]=1;}

        else if(0==strcmp(p1,"D"))

        { step[j]=2;}

        else if(0==strcmp(p1,"L"))

        {  step[j]=3;}

        else if(0==strcmp(p1,"R"))

        {  step[j]=4;}

        else if(0==strcmp(p1,"G"))

        {  step[j]=5;}

        j++;

        p2=strtok(NULL," ");

    }

   // scanf("%d %d",&m,&n);

    gets(temp\_str);

    p1=strtok(temp\_str," ");

    while(p1!=NULL)

    {

        m[i]=atoi(p1);

        i++;

        p1=strtok(NULL," ");

    }

        printf("%d %d\n",m[0],m[1]);

    //printf("%d %d\n",m,n);

   // printf("ttt %s",a[0][0]);

    for(i=0;i<m[0];i++)

    {

        gets(temp\_str);

        printf("%s\n",temp\_str);

        p2=strtok(temp\_str," ");

        j=0;

        while(p2!=NULL)

        {

            if(0==strcmp(p2,"F"))

            { a[i][j]=1;}

            else if(0==strcmp(p2,"E"))

            { a[i][j]=0;}

            else if(0==strcmp(p2,"H"))

            {  a[i][j]=2;}

            j++;

            p2=strtok(NULL," ");

        }

    }

    for(i=0;i<m[0];i++)

        for(j=0;j<m[1];j++)

        {

            if(a[i][j]==2)

            {

                loc\_x=i;

                loc\_y=j;

            }

       }

    return 0;

}

**例题5.1**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

    char str[2000];

    int a[1000];

    char \*p1=NULL;

    int i=0;

    int num=0;

    int j=0;

    int min=65535;

    gets(str);

    p1=strtok(str," ");

    while(p1!=NULL)

    {

        a[i]=atoi(p1);

        i++;

        p1=strtok(NULL," ");

    }

    num =i;

    for(i=0;i<num-1;i++)

    {

        for(j=i+1;j<num;j++)

        {

            if(abs (a[i]+a[j])<min)

            {

                min=abs(a[i]+a[j]);

            }

        }

    }

    printf("%d\n",min);

    return 0;

}

**例题 5.2**

#include<stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

    int num=0;

    int i=0;

    char str[1000];

    char \*p1=NULL;

    int a[1000];

    int j=0;

    int ab\_count=0;

    int loc=0;

    int flag\_late\_ear=0;

    int k=0;

    int rand\_count=0;

    gets(str);

    num=atoi(str);

    for(i=0;i<num;i++)

    {

        ab\_count=0;

        loc=0;

        j=0;

        rand\_count=0;

        memset(a,0,sizeof(a));

        gets(str);

        p1=strtok(str," ");        while(p1!=NULL)

        {

       if(0==strcmp(p1,"absent"))

            {

                a[j]=1;

                ab\_count++;

            }

            else if(0==strcmp(p1,"late"))

            {

                a[j]=2;

            }

            else if(0==strcmp(p1,"leaveearly"))

            {

                a[j]=3;

            }else if(0==strcmp(p1,"present"))

            {

                a[j]=4;

            }

            j++;

            p1=strtok(NULL," ");

        }

        loc=j;

        for(j=0;j<loc-1;j++)

        {

            if(a[j]==2&&a[j+2]==2||a[j]==2&&a[j+1]==3 || a[j]==3&&a[j+2]==2||a[j]==3&&a[j+1]==3)

            {

                flag\_late\_ear=1;

                break;

            }

        }

        for(j=0;j<loc-7;j++)

        {

            for(k=1;k<7;k++)

            {

                if(a[j+k]==1||a[j+k]==2||a[j+k]==3)

                {

                    rand\_count++;

                }

                if(rand\_count>3)

                {

                    break;

                }

            }

        }

        if(ab\_count<=1&&flag\_late\_ear<1&&rand\_count<=3)

        {

            printf("true ");

        }

        else

        {

            printf("false ");

        }

    }

    return 0;

}

**例题5.3**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

    char str[1000000];

    int a[1000000];

    int b[1000000];

    int i=0;

    int j=0;

    int num=0;

    char \*p1=NULL;

    int row=0;//行

    int col=0;//列

    int full\_flag=0;

    int day=0;

    int count=0;

    fgets(str,1000000,stdin);

    printf("%s",str);

    p1=strtok(str,",");

    while(p1!=NULL)

    {

        a[i]=atoi(p1);

        i++;

        p1=strtok(NULL,",");

    }

    num=i;

    row=sqrt(num);

    if(row\*row!=num)

    {

        ////返回输入错误

    }

    col=row;

    for(i=0;i<num;i++)

    {

        if(a[i]==1)

        {

            count++;

        }

    }

    if(count==num)

    {

        printf("all%d\n",-1);

        return 0;

    }

    else if(count==0)

    {

        printf("null%d\n",-1);

        return 0;

    }

    while(!full\_flag)

    {

        memset(b,0,sizeof(b));

        for(i=0;i<row;i++)

        {

            for(j=0;j<col;j++)

            {                                    //特别注意：

      if(a[col\*i+j]==1)   //如果是在原数组中修改，那么下一行的数据就是已经被修改的了，这样下去肯定是只有1天

                {

      b[col\*i+j]=1;//因此，需要将一个备份用的数组用来备份每次修改的数据，然后在循环结束时，将数据拷贝回原来的数组中

       if(j+1<col&&a[col\*i+j+1]!=1)

                    {

                         b[col\*i+j+1]=1;

                    }

          if(j-1>0&&a[col\*i+j-1]!=1)

                    {

                         b[col\*i+j-1]=1;

                    }

       if(i-1>0&& a[col\*(i-1)+j]!=1)

                    {

                         b[col\*(i-1)+j]=1;

                    }

     if(i+1<row&& a[col\*(i+11)+j]!=1)

                    {

                         b[col\*(i+1)+j]=1;

                    }

                }

            }

        }

        day++;

        count=0;

        for(i=0;i<num;i++)

        {

            if(b[i]==1)

            {

                count++;

            }

        }

        if(count==num)

        {

            full\_flag=1;

        }

        memcpy(a,b,sizeof(a));

    }

    printf("%d\n",day);

    return 0;

**例题 6.1**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

int main()

{

    char str[9];

    int len=0;

    int i=0;

    int sum=0;

    gets(str);

    len=strlen(str);

    printf("len=%d\n",len);

    for(i=0;i<len;i++)

    {

        if(str[i]-'0'>4)

        {

            str[i]=str[i]-1;

        }

     printf("str[%d]=%d\n",i,str[i]-'0');

        sum = sum\*9 + str[i]-'0';

    }

    printf("%d",sum);

    return 0;

}

**例题6.2**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

int max(int a,int b)

{

return a>b?a:b;

}

int isValid(char \*str)

{

    int len=0;

    int i=0;

    int j=0;

    char temp\_open[50000];

    char \*combin="(),{},[]";

    char temp\_str[3];

    len =strlen(str);

    for(i=0;i<len;i++)

    {

        memset(temp\_str,0,sizeof(temp\_str));

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

        {

            temp\_open[j]=str[i];

            j++;

        }

        else

        {

            if(j==0)//OPEn 数组为空

            {

                printf("jian 0\n");

                return 0;

            }

            else

            {

      temp\_str[0]=temp\_open[j-1];

      temp\_str[1]=str[i];

        printf("test temp\_str=%s\n",temp\_str);

                if(NULL==strstr(combin,temp\_str))

                {

                    printf("jian 1\n");

                    return 0;

                }

            }

        }

    }

    return 1;

}

int cal\_maxDepth(char \* str,char left,char right)

{

    int depth=0;

    int maxdepth=0;

    int len=0;

    int i=0;

    len=strlen(str);

    for(i=0;i<len;i++)

    {

        printf("str[%d]=%c\n",i,str[i]);

        printf("left=%c,right=%c\n",left,right);

        if(str[i]==left)

        {

            depth++;

            maxdepth=max(depth,maxdepth);

        }

        else if(str[i]==right)

        {

            depth--;

        }

    }

    printf("test maxdepth=%d\n",maxdepth);

    return maxdepth;

}

int main()

{

    char str[1000000];

    int len=0;

    int aa=0;

    int bb=0;

    int cc=0;

    int i=0;

    int maxi=0;

    gets(str);

    len=strlen(str);

    if(len%2)

    {

        printf("1 0\n");

        return 0;

    }

    for(i=0;i<len;i++)

    {

        if(str[i]=='(')

        {

            aa++;

        }

        else if(str[i]==')')

        {

            aa--;

        }

        else if(str[i]=='{')

        {

            bb++;

        }

        else if(str[i]=='}')

        {

            bb--;

        }

        else if(str[i]=='[')

        {

            cc++;

        }

        else if(str[i]==']')

        {

            cc--;

        }

    }

    if(aa!=0||bb!=0||cc!=0)

    {

        printf("2 0\n");

        return 0;

    }

    if(0==isValid(str))

    {

        printf("3 0\n");

        return 0;

    }

    maxi=max(cal\_maxDepth(str, '(', ')'),cal\_maxDepth(str, '{', '}'));

    maxi=max(maxi,cal\_maxDepth(str, '[', ']'));

    printf("%d\n",maxi);

    return 0;

}

**例题 6.3**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

typedef struct

{

    int size;

    int cnt;

}resourceInfo;

int main()

{

    int r=0;

    resourceInfo R[20];

    int i=0;

    int user=0;

    int res=0;

    int user\_cnt=0;

    scanf("%d",&r);

    printf("r=%d\n",r);

    for(i=0;i<r+1;i++)

    {

        scanf("%d",&R[i].cnt);

        R[i].size=pow(2,i);

    }

    scanf("%d",&user);

       i=0;

    while(i<=r)//循环的条件，需要特别注意

    {

        if(R[i].size==0||R[i].cnt==0)

        {

            i++;

            continue;

        }

        if(res!=0)  //上次有预留的，处理

        {

           if(R[i].size\*R[i].cnt>res)//先处理预留的数据，同时记得将res清空

           {

                if(res%R[i].size==0)

               {

                   R[i].cnt=R[i].cnt-res/R[i].size;

               }

               else

               {

                   R[i].cnt=R[i].cnt-res/R[i].size-1;

               }

               //本次阶中可以分配一次用户

               user\_cnt++;

               res=0;//清空

           }

           else

           {

               res=res-R[i].size \* R[i].cnt;

               i++;

           }

        }

        else   //上次没有预留的，

        {

           if(R[i].size\*R[i].cnt>user)  //直接判断，在本次阶中能不能分配足够的数据，就仅分配一次

           {

               if(user%R[i].size==0)

               {

                   R[i].cnt=R[i].cnt-user/R[i].size;   //分配完一次后，会修改 cnt，这样下次循环还能用

               }

               else

               {

                   R[i].cnt=R[i].cnt-user/R[i].size-1;

               }

               //本次阶中可以分配一次用户

               user\_cnt++;

           }

            else

            {

                res = user -R[i].size\*R[i].cnt;

                i++;

                printf("test i[%d],res[%d]\n",i,res);

            }

        }

    }

    printf("%d\n",user\_cnt);

    return 0;

}

**例题 8-1**

#include<stdio.h>

#include <stdlib.h>

#include <string.h>

void alp\_sprint(char\* str,int k)

{

    int j=0;

    int alp=0;

    for(j=0;j<k;j++)

    {

        if(str[j]>='a'&&str[j]<='z')

        {

            alp--;

        }

        else if(str[j]>='A'&&str[j]<='Z')

        {

            alp++;

        }

    }

    for(j=0;j<k;j++)

    {

        if(alp>0)

        {

            if(str[j]>='a'&&str[j]<='z')

            {

                printf("%c",str[j]-32);

            }

            else

            {

                printf("%c",str[j]);

            }

        }

        else if(alp<0)

        {

            if(str[j]>='A'&&str[j]<='Z')

            {

                printf("%c",str[j]+32);

            }

            else

            {

                printf("%c",str[j]);

            }

        }

        else

        {

            printf("%c",str[j]);

        }

    }

}

int main()

{

    int k=0;

    char str[1000];

    char \*p1=NULL;

    char temp\_str[1000];

    char temp\_str2[10];

    int i=0;

    int j=0;

    int first=0;

    int len=0;

    int alp=0;

    int count=0;

    scanf("%d",&k);

    printf("k=%d\n",k);

    scanf("%s",&str);

    printf("str=%s\n",str);

    memset(temp\_str,0,sizeof(temp\_str));

    p1=strtok(str,"-");

    while(p1!=NULL)

    {

        if(i==0)

            first=strlen(p1);

        strcat(temp\_str, p1);

        p1=strtok(NULL,"-");

        i++;

    }

    len=strlen(temp\_str);

    for(i=0;i<first;i++)

    {

         printf("%c",temp\_str[i]);

    }

    count=(len-first)/k;

    for(i=0;i<count;i++)

    {

        printf("-");

        for(j=0;j<k;j++)

        {

            temp\_str2[j] = temp\_str[first+k\*i+j];

        }

        alp\_sprint(temp\_str2,k);

    }

    if((len-first)%k!=0)

    {

        printf("-");

        for(j=0;j<len-first-k\*count;j++)

        {

            temp\_str2[j] = temp\_str[first+k\*count];

        }

        alp\_sprint(temp\_str2,len-first-k\*count);

    }

    return 0;

}

**例题8-2**

#include<stdio.h>

int main(void){

    int i,N,E,X,Y,area,a,b;

    scanf("%d %d",&N,&E);

    a=0;

    b=0;

    area=0;

    for(i=N;i>0;i--){

        scanf("%d %d",&X,&Y);

        area+=(X-a)\*b;

        a=X;

        b+=Y;

    }

    area+=(E-a)\*b;

    printf("%d\n",area);

    return 0;

}

**例题9-2**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

// 思路：计算任意两个之间的未照明距离，分别计算左A和右B所有的照明叠加，

//即计算A，A-1，A-2... 在此段距离的照明半径叠加

int getshadowLength(int lightNum,int\*lightRadium)

{

    int length=0;

    int index=0;

    while(index<lightNum-1)

    {

        bool jump=false;

        if(lightRadium[index]+lightRadium[index+1]>=100)

        {

            //若当前间隔下的光半径加起来超过100，则不需要判断了，直接跳过

            index++;

            continue;

        }

        //本次间隔下半径不够，分别向 左边、右边的路灯查找

        int leftLightlen=lightRadium[index];

        int rightLightlen=lightRadium[index+1];

        int index\_l=index-1;

        while(index\_l>=0)

        {

            if(lightRadium[index\_l]<=100)

            {

                //左边的路灯半径都小于1个间隔，肯定不够，跳过继续找

                index\_l--;

                continue;

            }

            int n=lightRadium[index\_l]/100;//计算左边找到的半径有多少个间隔

            if(n>=index-index\_l)//若 找到的半径 都不够差值个间隔，就照耀不到当前的间隔了

            {

                if(n>=index-index+1)//如果找到的半径间隔 比 差值个数+1 还大，说明可以照耀到index+1 的间隔中

                {

                    index +=n-(index-index\_l);//index 可以跳过中间的这些间隔了

                    jump=true;

                    break;

                }

                //如果 index-index\_l=< n <=index-index\_l+1,说明找到的左边的灯仅能照耀到当前的间隔

                int l=lightRadium[index\_l]-100\*(index-index\_l);//计算出当前剩余的光长度

                if(l>leftLightlen)//如果剩余的光长比原来的index半径还大，则替换掉

                {

                    leftLightlen=l;

                }

            }

        }

        if(jump)

        {

            continue;

        }

        //经过上面寻找左边灯的逻辑后，在index 没有发生跳转时候，重新判断是否超过100

        if(leftLightlen+lightRadium[index+1]>=100)

        {

            index++;

            continue;

        }

        //灯还是不够，需要从右边继续找

        int index\_r=index+2;

        while(index\_r<lightNum)

        {

            if(lightRadium[index\_r]<=100)//半径小于100 的直接跳过

            {

                index\_r++;

                continue;

            }

            int n=lightRadium[index\_r]/100;//算出半径能够够多少个间隔

            if(n>=index\_r-(index+1))//若找到的光半径够差值个间隔

            {

                if(n>=index\_r-(index+1)+1)//若右边找到的光半径，比index\_r-(index+1)+1=index\_r-index还大，说明 能够照到index的左边

                {

                    index =index+2\*n;//index可以做到一下子 连跳2\*n个灯柱，index+n是到了index\_r,但是index\_r半径为n,还可以跳n个，所以是+2n

                    jump = true;

                    break;

                }

                //差值在 index\_r-(index+1)<= n <index\_r-(index+1)+1直接，其实就是 index\_r-(index+1)，

                //index\_r刚好可以照到index+1的左边，和index在一个间隔中

                int r=lightRadium[index\_r]-100\*(index\_r-(index+1));

                if(r>rightLightlen)

                {

                    rightLightlen=r;

                }

            }

        }

        if(jump)

        {

            continue;

        }

        if(leftLightlen+rightLightlen>=100)

        {

            index++;

            continue;

        }

        length +=100-leftLightlen-rightLightlen;

        index++;

    }

    return length;

}

int main(void){

    int n=0;

    int r[100000]={0};

    int i=0;

    int dis=0;

    scanf("%d",&n);

    printf("%d\n",n);

    for(i=0;i<n;i++)

    {

        scanf("%d",&r[i]);

        printf("%d\n",r[i]);

    }

    dis=getshadowLength(n, r);

    printf("%d\n",dis);

    return 0;

}