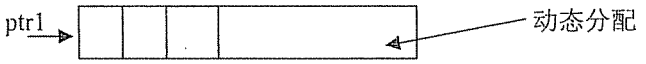
2000 年研究生入学考试《C 语言》试题与解答：

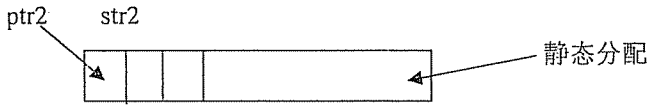
一、基础部分：

1、解：*ptr1=(char *)malloc(10);

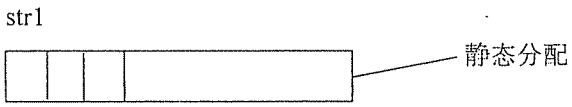


10 个单元

ptr2=str2;



scanf("%s",str1);

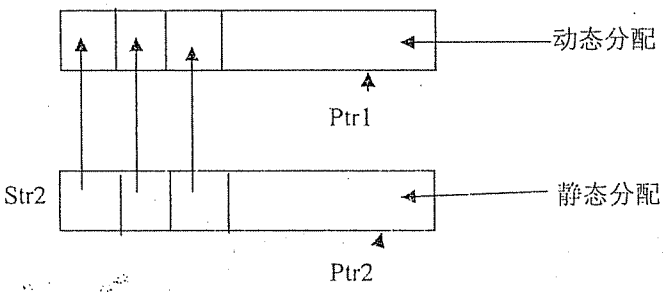


strcpy(ptr,str1);

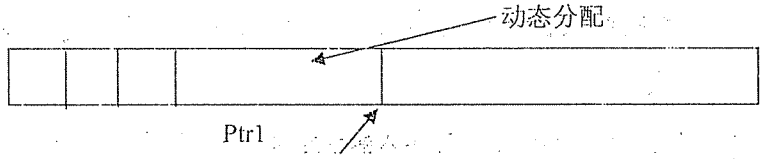
ptr 未声明，为语法错误。

while(*ptr1++ = *ptr2++);

这个循环将 str2 数组中的元素值依次赋给由指针 ptr1 所指向的空间，ptr1 和 ptr2 同步移动，结果如下：



free(ptr1);



free 的是这 10 个单元，这是非常危险的，错！！

2、使用 typedef 来将 FUNPTR 定义为指向函数的指针类型，该函数返回整型值。

```
typedef  int (*FUNPTR ());
```

二、略

四、解：int turn(char *str)

```
{
    char *p;
    p=str+strlen(str)-1;
      if(*str!=*p)
            return 0;
      else
      {                                    //当前是回文
        if(p->str<=2)
```

```
            return 1;
          else
         {
            *p='\0';
            return (turn(str+1));
         }
       }
}
```

五、

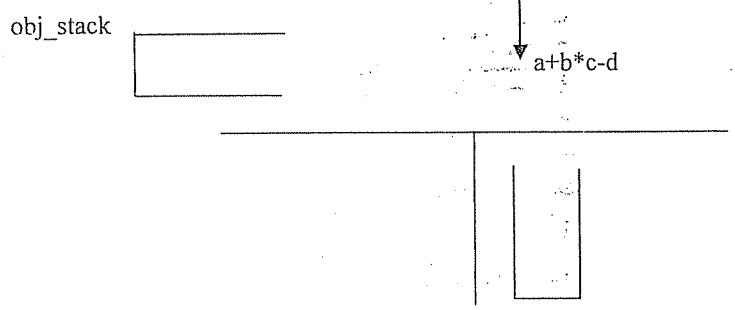
```
void store(char *filename)
{
      FILE *fp;
      int i,k;
      char x,str[30],str1[30],*p,*q;
      if((fp=fopen(filename,"w"))!=NULL)
     {
           scanf("%s",str);
           p=str;q=str1;
           *q=*p;q++;
           x=*(p+2);                                    //x 为参数
           while(*p!='=')   p++;
           while(*(p+2)!='\0')
          {
              for(i=0,p++;i<3;i++,p++);
              k=0;
              while(*p>='2'&&*p<'9')                    //计算指数 k
                    k=k*10+(*p++)-'0';
              for(i=0;i<k;i++)
                    *q++=x;
              *q++=*p;
          }
          *q++=x;                                       //添加运算符
          *q='\0';
          fputs(str1,fp);fputc('\n',fp);
      }
      fclose(fp);
}
void restore(char *filename)
{
   FILE *fp;
   int i,k;
   char x,str[30],str1[30],*p,*q,num[5];
   if((fp=fopen(filename,"r+"))!=NULL)
  {
```

```
        fgets(str,30,fp);
        p=str;q=str1;
            *q++=*p++;
        x=*p;                                                    //取变量
        *q++='(';    *q++=x;    *q++=')';    *q++='=';
        while(*p!='\0')
       {
         k=0;                                                    //计算指数
         while(*p==x)
        {
          k++;   p++;
        }
         if(k==1)                                                //最后一项
        {
          *q++=x;p++;
        }
         else
        {
          i=0;                                                   //生成指数项
          while(k/10!=0)
         {
           num[i++]=k%10+'0';
           k/=10;
         }
          num[i]=k+'0';
         *q++=x;    *q++='*';    *q++='*';
         for(;i>=0;i--)                                          //拼装指数
              *q++=num[i];
         if(*p!='\0')
              *q++=*p++;
         else
              break;
        }
       }
          *q='\0';
         printf("%s\n",str1);
 }
      fclose(fp);
}
int main()
{
    restore("dxs.txt");
    return 0;
}
```

六、解：struct node

```
{
    char n;
    struct node *left,*right;
};
char op_stack[20];                    //运算符号栈
struct node *obj_stack[20];           //运算对象栈
char str[50]="a+b*c-d";
```

假定运算优先级：*、/、>、+、—



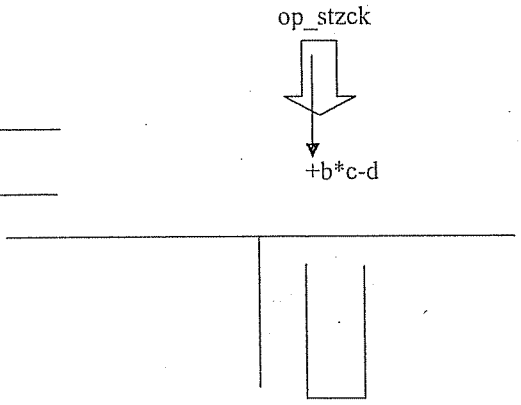
原理

1、看 str 中的符号串，

如果当前符号是运算对象，

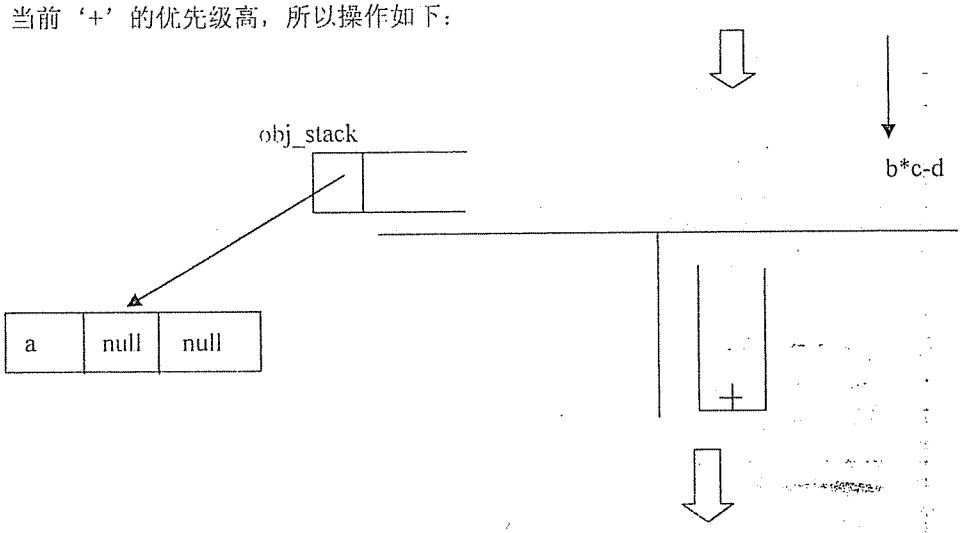
例如 a，则为其建立一个节点，

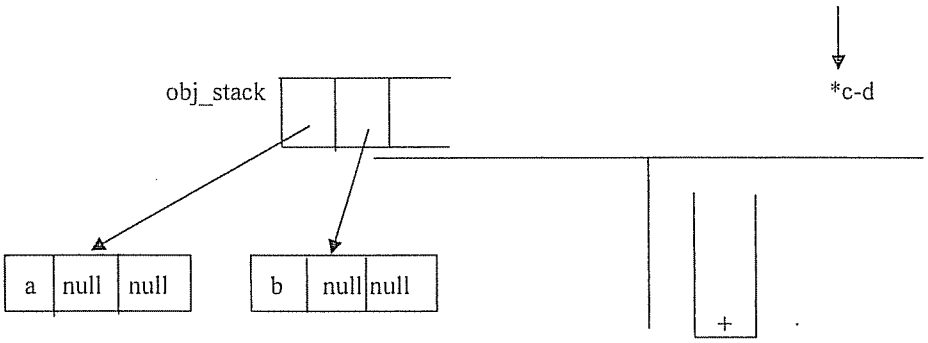
并将节点指针存入 obj_stack 栈。

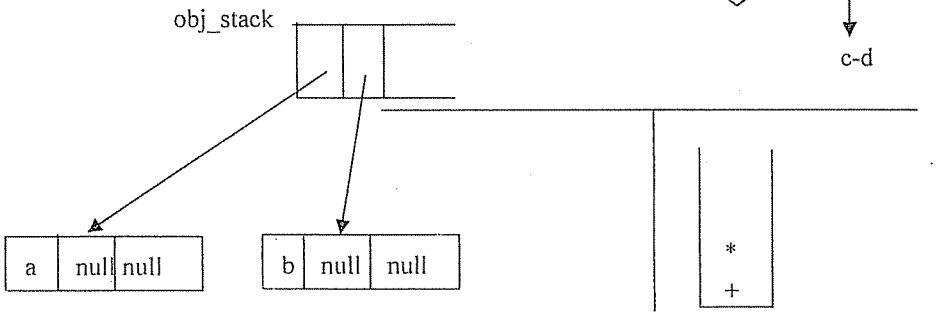


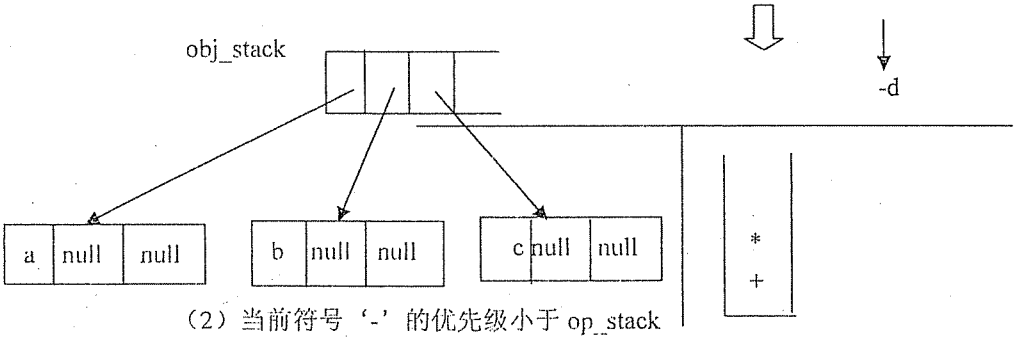
2、如果 str 当前符号是运算符号，则可能有如下几种操作：

（1）当前符号的优先级大于 op_stack 的栈顶符号的优先级，则当前符号如 op_stack 栈；例如，当前‘+’的优先级高，所以操作如下：

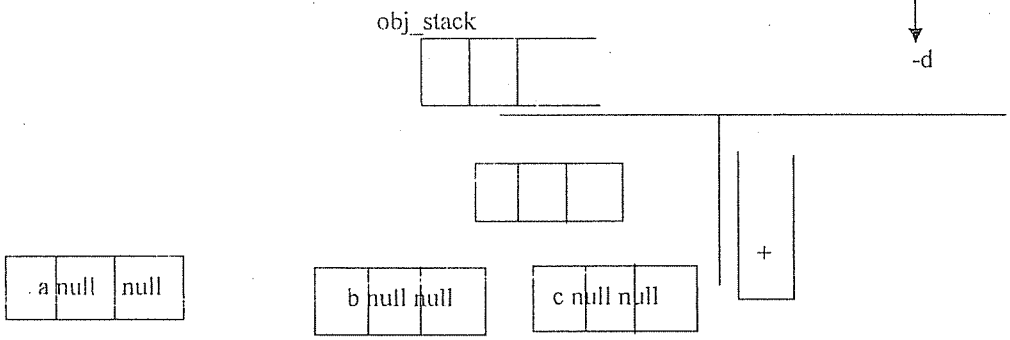




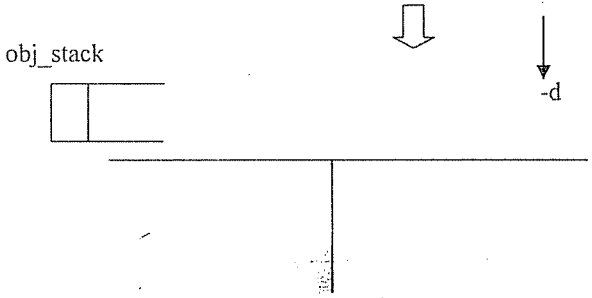


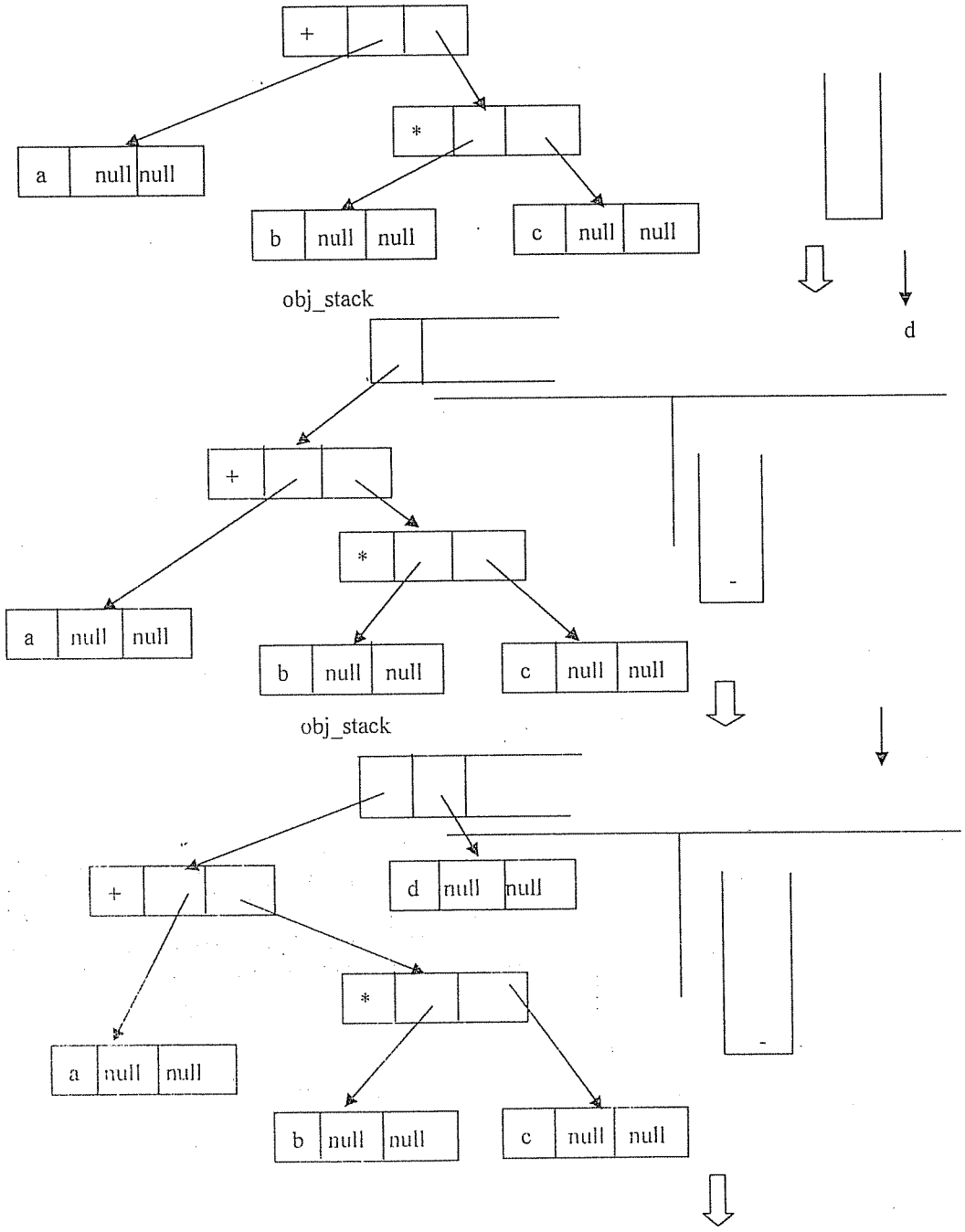


（2）当前符号‘-’的优先级小于 op_stack 的栈顶符号‘*’的优先级，则生成一个子树，并调整栈如下：



（3）当前符号‘-’的优先级等于 op_stack 的栈顶符号‘+’的优先级，则生成一个子树，并调整栈如下：





对这棵树进行后序遍历，则得到逆波兰式。

程序略。

七、例如：str="a nice string","pattern="%nice%str_ng"时函数返回值为真。

```
int like(char *substr,char *str)
{
   int i,j,k;
   for(i=0;str[i];i++)
   {
       for(j=i,k=0;str[j]==substr[k]||substr[k]=='?'||sustr[k]=='*'||substr[k]=='%')
```

```
}
{
    if(str[j]==substr[k])
    {j++; k++;}
    else
    {
        if(k>0&&substr[k-1]=='\\')
                    break;
        else
        {
            if(substr[k]=='?'||substr[k]=='_')
            {j++;   k++;}
            else
            {
                k++;
                while(substr[k]!=str[j]&&str[j]!='\0')
                        j++;
                if(substr[k]==str[j]&&str[j]!='\0')
                {j++;   k++;}
                else
                        return i;
            }
        }
     }
       if(!substr[k])
          return i;
   }
 }
  return -1;
}
void main()
{
    char str1[]="%nice%str*g",str2[]="a nice string";
    printf("%d",like(str1,str2));
    return 0;
}
```

2002 年研究生入学考试《C 语言》试题与解答：

一、1、解：输出#*#*#$

要点：continue 语句使得执行立即转向循环条件语句。

2、解：是死循环，一直输出#。

要点：1<=x<=2，1<=x 的值非即 0，永远小于 2，所以整个表达式的值永远为 1.

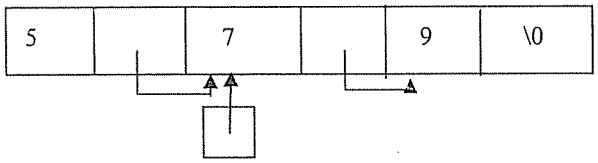
3、D

4、B

解：注意"\t\v\b\\\0abc\n"中的'\0'，是字符串结束符号。

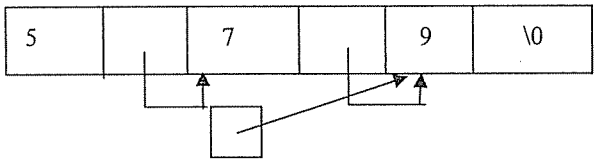
5、D

6、解：



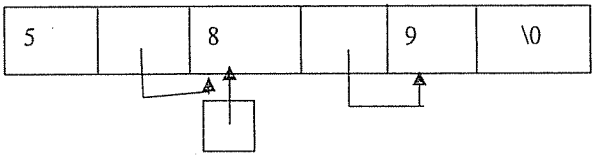
（A）p++->n

输出 7，然后 p++



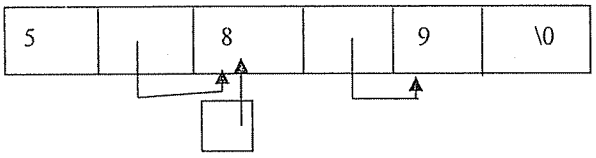
（B）p->n++

输出 7，然后 n++



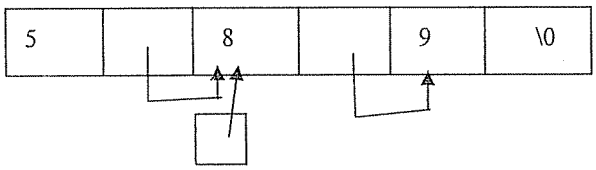
（C）（*p）.n++

输出 7，然后 n++



（D）++p->n

p->n 先++，然后输出 8.



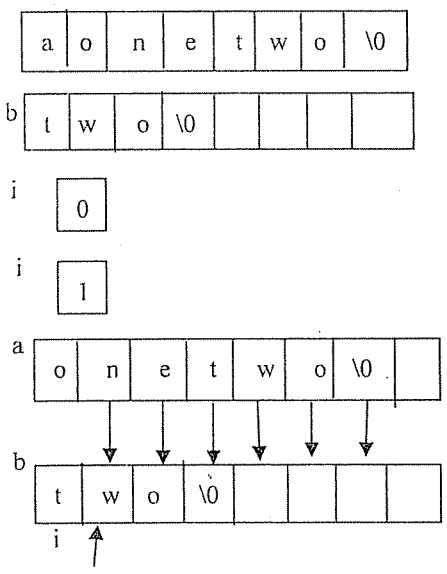
6、解：输出 1 2 3 4 5 6

7、解：输出 3.

8、输出 tnetwo

```
char a[80]="one",b[80]="two";
    int i=0;
    strcat(a,b);

while(a[i++]!='\0')
b[i]=a[i];
```



9、解：输出 in。

10、解：abc 文件的内容为 firstd。

二、略。

四、1、解：　　输入：　　输入：

数组元素　　鞍点的值；

鞍点的位置；

鞍点的个数 n。

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 3 & 6 \\ 3 & 5 & 6 & 5 \end{pmatrix}$$

变量设计：

```
int i,i0,j;                    //行和列的指针
int max;                       //一行上的最大值，有可能就是鞍点的值
int j0;                        //一行上的最大值所在的列
```

算法设计：

1、输入数组的值，并打印出来；

2、for(i=0;i<N;i++)

{

3、　寻找第 i 行上的鞍点，并输出鞍点；

}

4、if（n>0）

　　输出鞍点的个数 n；

else

　　输出“不存在鞍点”；

```
void SEARCHAD(int a[][10],int row,int col)
{
    int i,j;
    int i0,j0,max,n=0;
    for(i=0;i<row;i++)
    if(a[i][j]>max)
    {
      max=a[i][j];
      j0=j;
    }
    for(i0=0;i0<row&&max<=a[i0][j0];i0++);
    if(i>=row)
    {
      n++;
      printf("a saddle point--%d in the line%d,columu%d!\n",max,i,j0);
    }
  }
    if(n)
        printf("have %d saddle-points!\n",n);
    else
        printf("no saddle-points!\n");
}
```

4、解：

```
#define N 9
```

```
int Qsort(int *a,int l,int h)
{
    int key,low=1,high=h;
    if(low==high)
          return 1;
    else
   {
     key=a[low];
     while(low<high)
     {
         while(low<high&&a[high]>=key) high--;
         a[low]=a[high];
         while(low<high&&a[low]>=key) low--;
         a[high]=a[low];
     }
     a[low]=key;
     if((low=low-1)<1) low=l;
       if((high=low+1)>h) high=h;
     Qsort(a,l,low);        Qsort(a,high,h);
     return 0;
    }
}
```

2003 年研究生入学考试《C 语言》试题与解答：

一、解：void (*f(int no))();是错误的。因为这是一个指向函数的指针的定义，一般形式是：数据类型 （*指针变量名）（）；即 f（int no）部分应该是一个指针变量名。

```
LRESULT   (*ptr)();
LRESULT   a;
ptr=MyProc();
a=(*ptr)();
```

二、1、解：输出：writeherehere

```
char strlist[3][5]={'\0'};
```

strlist

\0				

```
strcpy(strlist[1],"write--");
```

strlist

\0				
w	r	i	t	e
-	-	\0		

```
srcpy(strlist[2],"here");
```

strlist

\0				
w	r	i	t	e
h	e	r	e	\0

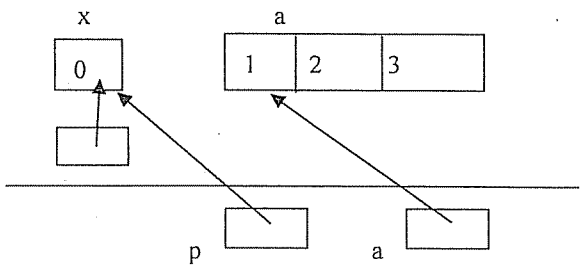
2、解：7

要点：case 子句若无 break，将继续执行下面的 case 子句。

continue 使得运行直接转到条件判断语句。

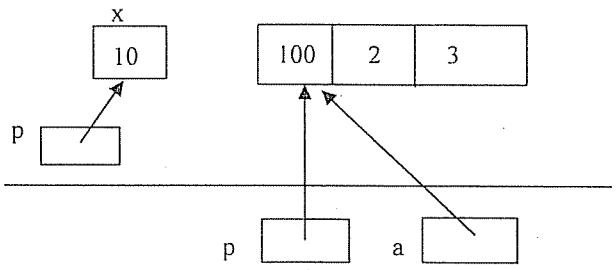
3、解：10-10-100-2

```
int x=0,*p,a[3]={1,2,3};
p=&x;
```




```
f(p,a);

*p=10;
p=a;
*p=100;
```




```
printf("%d-%d-%d-%d",x,*p,a[0],a[1]);
```

4、解：No.0 scores:　2　47.0　4 26.0

No.1 scores:　2　59.0

No.0 scores:　1　43.0　4 56.0

要点：用指针标识多维数组时，注意指针是行指针还是列指针。行指针按行移动。列指针按元素（按列）移动。赋值时。左值和右值指针类型要匹配。

三、要点：闰年的计算公式。

四、1、要点：这是一个涉及文件操作的问题。原则上文件指针每次移 10kb，但是这个位置上不一定恰好是一个词条的起始点，这时需要一次移动文件指针，直到找出最近出现的"#"的位置，这肯定是一个词条的起点，就建立一个索引值：词条 词条在文件中的位置。

2、程序：

```
#define INDEX_LEN 500
#define BLOCK 20480
#define THRESHOLD   1024
typedef struct indtag{
        char entry[50];
        long posi;
}INDEXSTRUCT;
void CreateIndex(FILE *in,FILE *out);
void LoadIndex(FILE *fp,INDEXSTRUCT *ind,int *len);
char *LookUp(char *buf,char word[],FILE *fo);
INDEXSTRUCT inditem[INDEX_LEN];
int len;
char bbuf[BLOCK],word[50];
int main()
{
```

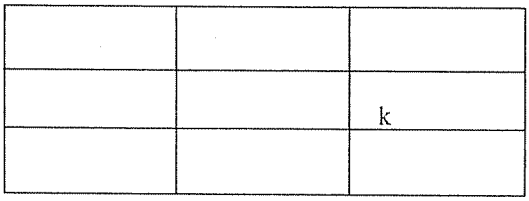
```
FILE *in,*out;
int i;
char *p;
if((in=fopen("die","r+b"))==NULL)
{
    printf("No dic file.Exit");
    exit(1);
}
out=fopen("ind.txt","wb");
CreatIndex(in,out);
fclose(out);
out=fopen("ind.txt","rb");
LookIndex(out,inditem,&len);
fclose(out);
printf("\nInput word to look up");
scanf("%s",word);
p=(char *)lookup(bbuf,word,in);
for(i=0;i<50;i++) printf("%c",*(p+i));
}
void CreatIndex(FILE *in,FILE *out)
{
    char buf[BLOCK],item[50];
    int i,j,l;
    long prevposi,oldposi,newposi;
    char *p;
    int endflag=0;
    oldposi=newposi=prevposi=0L;
    j=0;
    while(!endglag&&!feof(in))
   {
     l=fread(buf+j,l,BLOCK-j,in);
     if(l<BLOCK-j) endflag=1;
     j=0;
     while(BLOCK-j>THRESHOLD&&j<=l)
    {
        newposi++;
        if(*(buf+j)=='#')
        {
            if(newposi-oldposi>10240)
            {
                fprintf(out,"%s %ld\n",item,prevposi);
                oldposi=newposi;
            }
            preposi=newposi;
```

```
            for(i=0;l<50&&isalpha(*buf+i+j+l);i++)
                    item[i]=*(buf+i+j+l);
            item[i]='\0';
          }
          j++;
      }
       memmov(buf,buf+j,BLOCK-j);
    }
}
void LoadIndex(FILE *fp,INDEXSTRUCT *ind,int *len)
{
        int i=0;
        while(!feof(fp))
       {
            fscanf(fp,"%s %ld\n",ind[i].entry,&ind[i].posi);
            i++;
       }
       *len=i;
}
char *LookUp(char *buf,char word[],FILE *fp)
{
    int i,j,m,re,l,k;
    char item[50];
    for(i=0,j=len-1;i<j)
    {
        m=(i+j)/2;
        if((re=strcmp(inditem[m].entry,word))==0)
        {
            i=m;
            break;
        }
        else if(re>0){i=m;}
        else {j=m;}
    }
fseek(fp,inditem[i].posi,0);
l=fread(buf,0,BLOCK,fp);
buf[l]='\0';
k=0;
while(k<=l)
{
    if(*(buf+k)=='#')
    {
        for(i=0;i<50&&isalpha(*(buf+i+k+l));i++)
            item[i]=*(buf+i+k+ll);
```

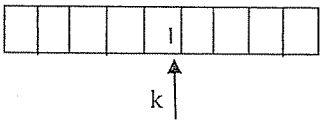
```
        item[i]='\0';
        if((m=strcmp(item,word))==0)    return buf+k+l;
        if(m>0)     return NULL;
    }
    k++;
}
return buf+k;
}
```

五、fill(num,n)

设： a[n][n]



b[num]



b 数组作为占用标志，例如：b[k]=1，表示数字 k 已经在 a 数组中，即已经被占用。

a）算法：

1、根据 b 数组信息，用未被占用的数（如 b[j]=0，则 j 是未被占用数）填满 a 数组；

2、判断 a 数组行、列元素和是否都相同，如果都相同，则 ok=1，否则 ok=0；

3、if(ok)　输出 a 数组；

4、回溯，找到适当的回溯点；

b）程序：

```
int a[10][10],b[50];
int isequal(int n)
{
    int ok,k,i,j,ki,kj;                                    //k 为第一行的和
    ok=1;k=0;
    for(i=0;i<n;i++)    k=k+a[0][i];
    i=1;
    while(ok&&i<n)                                         //判断行元素和是否相同
   {
       ki=0;
       for(j=0;j<n;j++)    ki=ki+a[i][j];
       if(k!=ki)    ok=0;
       i++;
   }
    j=0;
    while(ok&&j<n)                                         //判断列元素和是否相同
   {
      kj=0;
      for(i=0;i<n;i++)    kj=kj+a[i][j];
```

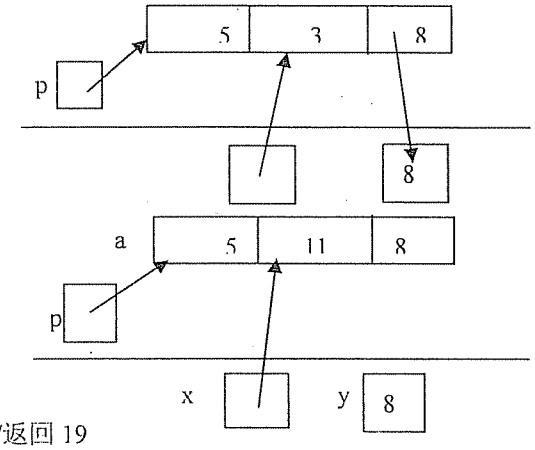
```
    if(k!=kj)       ok=0;
    j++;
  }
   return ok;
}
void fill(int num,int N)
{
    int i,j,k,c=0,flag=1,m,n,i1,j1;
    for(i=0;i<num;i++)     b[i]=0;                    //初始化，1~num 都未被占用
    m=0;n=0;                                          //行和列
    while(flag)
   {
     j=0;
     while(m<N)                                       //找未被占用数，填满 a 数组
     {
        while(j<num&&b[j]) j++;                       //找到一个
        b[j]=1;                                       //j 被占用
        a[m][n]=j;                                    //j 填入 a 数组
        j++;
        if(n<N-1)                                     //a 数组下一个元素
             n++;
        else
             n=0,m++;
     }
      n--;
      if(n<0)
        n=N-1,m--;
      if(isequal(N))                                  //如果行、列元素和相等
      {
        for(j1=0;j1<N;j1++)                           //输出这组数
              printf("%d",a[i1][j1]);
        printf("\n");
      }
       scanf("%c",&c);
   }
    do                                                //回溯
    {
      b[a[m][n]]=0;
      n--;
      if(n<0)
          n=N-1,m--;
      if(m<0) flag=0;
      for(j=a[m][n]+1;j<num&&b[j];j++);
      if(j<num) break;
```

```
    }while(flag);
    if(flag)                                          //定位到回溯点
   {
      b[a[m][n]]=0;b[j]=1;
      a[m][n]=j;
      if(n<N-1)
           n++;
      else
           n=0,m++;
   }
 }
}
```

2004 年研究生入学考试《C 语言》试题与解答：

一、1、解：49 11 8

```
int a[3]={5,3,8},*p=a;
```



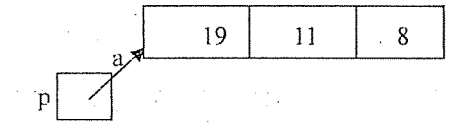

```
*p=f(&a[1],a[2]);
```



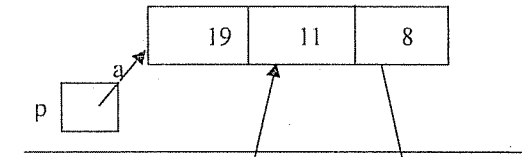
```
 if(*x<y) *x+=y;
    else y+=*x;
```

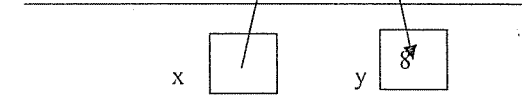


```
 return (*x+y);                    //返回 19
```

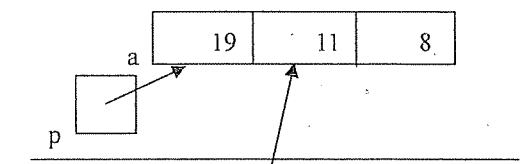



```
*p+=f(&a[1],a[2]);
```

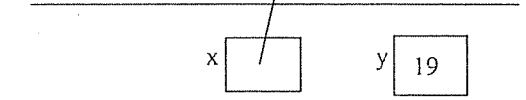


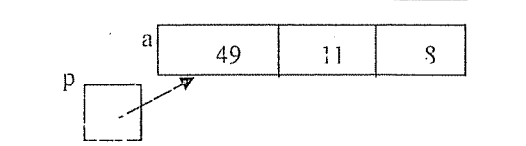



```
if(*x<y) *x+=y;
   else y+=*x;
```




```
return (*x+y);          //返回 30
```





2003 年研究生入学考试数据结构试题与解答

一、1、解：（1）对于有向图，图中的边数等于邻接矩阵中非零元的个数，对于无向图，等于邻接矩阵中非零元个数的一半。

（2）任意两个顶点 i 和 j 是否有边相连，对于无向图，若 e=（1/2）n（n-1），则为完全图，对于有向图，e=n（n-1）则为完全图（n 为定点数目）：若判断两个顶点是否相连，则看邻接矩阵中元素值为 1 的情况。

（3）任一顶点的度等于顶点所在行或列的非零元的个数，对于有向图，则为行和列的非零元的个数之和。

2、解：（1）大顶堆。

（2）不是，调整为小顶堆（12,24,33,65,24,56,48,92,86,70）。

（3）小顶堆。

3、解：（1）二维数组 C[n][n+1]的存储地址为：

Loc[i,j]=Loc[0,0]+[i*(n+1)+j]*L

（2）矩阵 A 中元素在 C 中的存放位置公式：

$$K=\begin{cases} n=(n+1)/2 & \text{当 } i<j \text{ 时} \\ i=(i-1)/2+j-1 & \text{当 } i>=j \text{ 时} \end{cases}$$

（3）矩阵 B 中元素转量后在 C 中存放公式：

$$K=\begin{cases} j(j-1)/2+i-1+n(n+1)/2 & \text{当 } i<=j \text{ 时} \\ n(n+1)-1 & \text{当 } i>=j \text{ 时} \end{cases}$$

二、void InsertSort_L(LinkList &La)

```
{  //用直接插入排序使链表递增有序
   if(La->next)       //链表不空
   {
     p=La->next->next;
     La->next->next=NULL;
     while(p!=NULL)
       { r=p->next;    //暂存 p 的后继
        q=La;
        while(q->next&&q->next->data<p->data)
            q=q->next;  //查找插入位置
            p->next=q->next;  //插入
            q->next=p;
            p=r;
       }//while
    }//if
}//InsertSort_L
```

三、解：结点结构：

child	Ltag	data	Rtag	rchild

void InTraverseThr(BithrTree thrt)

```
{//遍历中序线索二叉树
  p=thrt->lchild;//p 指二叉树根结点
  while(p!=thrt)
  {
    while(p->Ltag==0)
      p=p->lchild;
    printf(p->data);
    while(p->rtag==1&&p->rchild!=thrt)
    {
      p=p->rchild;
      printf(p->data);
    }//while
    p=p->rchild;
    }//while
}//InTraverseThr
```

四、

```
（1）void create_Hs(RedType &H,key Type key)
{
   i=Hash(key);    //创建哈希表
   if(H(i)==NULL) //插入
   H(i)=key;
   else              //解决冲突，再插入
   {
      j=(i+1)%m; //m 为表长
      while(j!=i)
      {
         if(H[j]==NULL)
            H[j]=key;
         else j=(j+1)%m;
      }//while
   }
}//create_Hs
（2）void print_Hs(RedType H)
{
   //
   for(i=0;i<26;i++)
    { j=1;
      while(H[j]!=Null)
      {
         if(f(H[j])==i)
            printf(H[j]);
         j=(j+1)%m;
      }//while
    }//for
```

}//print_Hs

2004 年研究生入学考试数据结构试题与解答：

一、解：第一趟　{20,15,21,10,25,26}30{36,40}

第二趟　{10.15}20{21,25,26}

第三趟　10{15}

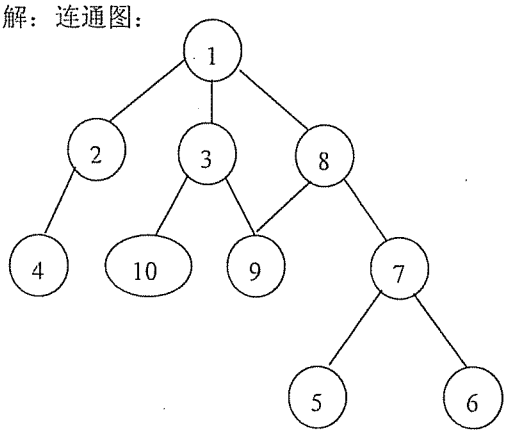
第四趟　21{25,26}

第五趟　25{26}

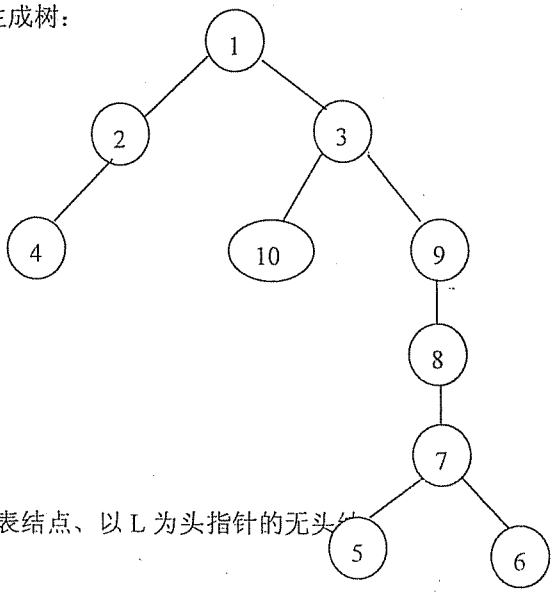
第六趟　36{40}

最终结果：{10,15,20,21,25,26,30,36,40}

2、解：连通图：

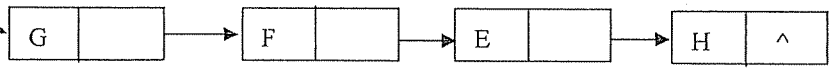


深度优先生成树：



3、解：（1）按中序遍历二叉树，逆序建立以叶子结点为链表结点、以 L 为头指针的无头结点的单链表。

（2）L



二、解：定义此单链表的结点结构：

```
    typedef struct Lnode
{
    int number;//数据域
    int value;//价格域
    struct Lnode *next;//指针域
}Lnode,*Linklist;
void insert_link(Linklist &L,int n)
{//将新到 n 台价格不同的手机插到有序链表中
    for(i=1;i<=n;i++)
    {
        scanf(&price);    //读入每台手机价格
        s=(LinkList )malloc(sizeof(Lnode));
        s->value=price;s->number=1;
        p=l->next;
        if(p->value<s->value)   //插入首部
        {
```

```
            s->next=L->next;L->next=s;
        }//if
        else
        {
            while(p->next&&(p->next->value>s->value))
                p=p->next;
            if(!p->next)//插入表尾
            {
                s->next=p->next;p->next=s;
            }//if
            else if(p->next->value=s->value)
                    p->next->number++;
            else
            {
                s->next=p->next; p->next=s;
            }//else
        }//else
    }//for
}//insert_link
```

三、解：（1）

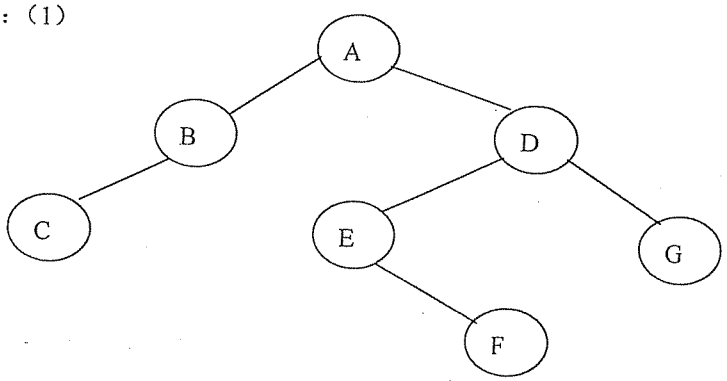
```
void insert_HS(HashTable &H,keytype Key)
{//哈希表插入，用链地址法解决冲突
    i=H(Key);//获取哈希地址
    if(H[i]==Null)
    {
        s=(Linklist)malloc(sizeof(Lnode));
        s->data=key; s->next=H[i];   H[i]=s;
    }//if
    else  //在链表中查找 key
    {
        p=H[i];
        while(p&&p->data!=key) p=p->next;
        if(p->data==key)     exit(1);
        else
        {
          s=(Linklist)malloc(sizeof(Lnode));
          s->next=H[i]; H[i]=s;
        }//else
    }//else
}//insert_HS
```

（2）

```
void Delete_HS(HashTable &H,KeyType key)
{//哈希表删除，用链地址法解决冲突
    i=H(key);        //获得哈希地址
    if(H[i]==Null)   exit(1);
    p=H[i];
```

```
    q=p;        //p 为工作指针，q 为 p 的前趋
    while(p&&p->data!=key)  //查找
   {
      q=p;
      p=p->next;
   }//while
    if(!p)  exit(1);
    if(q==H[i])    //key 为第一个结点
    {
      H[i]=p->next;free(p);
    }//if
    else
    {
      q->next=p->next;
      free(p);
    }//else
}//Delete_HS
```

四、解：（1）



（2）

```
void CreatTree(BiTree &T,char *str)
{//根据广义表的嵌套括号表示法，生成二叉链表树
   BiTree stack[maxsize],p;
   int k,j=0,top=-1;//j 为 str 指针，top 为栈顶指针
   T=null;        //初始化栈顶指针
   char ch=str[j];
   while(ch!='$')
   {
    switch(ch)
    {
      case '(':top++;stack[top]=p;//入栈
                k=1;break;  //k=1，为左孩子
      case ')':top--;break;//出栈
      case ',':k=2;break;//k=2.为右孩子
      default: p=(BiTree)malloc(sizeof(BTNode));
               p->data=ch;p->lchild=p->rchild=Null;
               if(T==Null)  T=p;//创建根节点
```

```
                else
                   switch(k)
                    {
                        case'1':stack[top]->lchild=p;
                                break;
                        case'2':stack[top]->rchild=p;
                                break;
                    }//switch
        }//switch
        j++;
        ch=str[j];
      }//while
}//creatTree
```