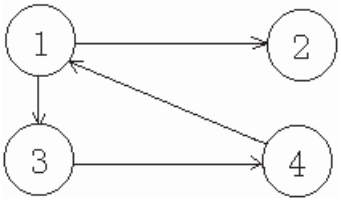
中国科学院软件研究所

一九九五年招收硕士学位研究生入学考试试题

试题名称：软件基础

一.（6 分）请给出下图的邻接矩阵、邻接表、逆邻接表和十字链表。



二.（12 分）编一个程序，按递增次序生成集合 M 的最小的 100 个数。M 的定义如下：

（a）数 1 属于 M；

（b）如果 x 属于 M，则 y=2*x+1 和 z=3*x+1 也属于 M；

（c）再没有别的数属于 M。　（M={1，3，4，7，9，10……}）

三.（8 分）使用对半查找程序的限制条件是什么？下列的三种对半查找程序（Pascal 语言），哪些是正确的，哪个效率高一些？假定 N>0，以及下列变量已经定义。

```
var i，j，k ： integer；
        a ： array [1..N] of T;
        x ： T;
```

程序 A:

```
    i:=1;    j:=N;
    repeat
      k:=(i+j) div 2
      if a[k]<x then i:=k else j:=k;
    until (a[k]=x) ∨(i≧j)
```

程序 B:

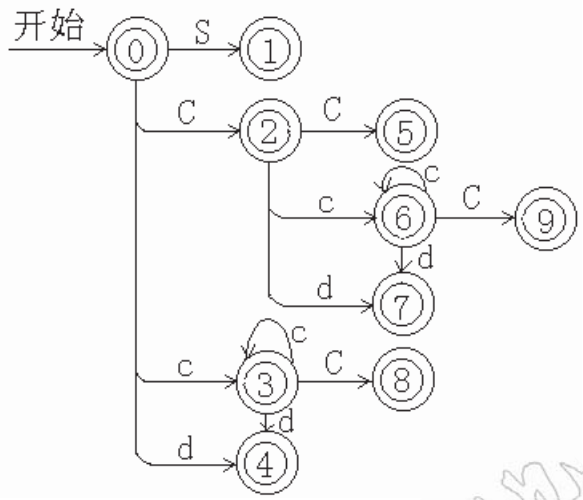
```
    i:=1;    j:=N;
    repeat
      k:=(i+j) div 2;
      if x≦a[k] then j:=k-1;
      if a[k]≦x then i:=k+1;
    until i>j;
```

程序 C:

```
    i:=1;    j:=N;
    repeat
      k:=(i+j) div 2;
      if x<a[k] then j:=k else i:=k+1;
    until i≧j
```

四．（14 分）用高级语言（C 或 Pascal）设计一个非递归的快速排序程序。

五．（6 分）将下面的 DFA 化成最简 DFA（注意，大写 C 和小写 c 是不同符号）



六．（7 分）文法 G 的产生式如下：

S ⟶ I|R　　I ⟶ **d**|I**d**　　R ⟶ W**p**F

W ⟶ ε |W**d**　　F ⟶ **d**|F**d**

1．令 **d** 表示任意数字，**p** 表示十进制小数点，那么非终结符 S，I，R，W 和 F 在程序设计语言中分别表示什么？

2．该文法是 LR（1）文法吗？为什么？

七．（10 分）有文法：

S ⟶ L.L|L

L ⟶ LB|B

B ⟶ **0**|**1**

给此文法配上语义动作子程序（或者说为此文法写一个语法制导定义），它输出 S 产生的二进制数的值。例如，输入 101.101 时，输出 5.625。

八．（7 分）

1．下面 C 语言程序中，函数 printf 的调用仅含一个参数。该程序输出三个整数。试从存储分配和 printf 的实现来分析，为什么此程序仍有三个整数输出。

```
main（）{   printf(“%d,%d,%d\n”)   }
```

2．考虑下面的 C 程序

```
main（）
  {
    char *cp1，*cp2；
    cp1:= “12345”;
    cp2=“abcdefghij”;
    strcpy(cp1,cp2);
```

```
    printf("cp1=%s\ncp2=%s\n",cp1,cp2);
}
```

该程序运行的结果是：

```
cp1=abcdefghij
cp2=ghij
```

试分析，为什么 cp2 所指向的串被修改了？

九．简答题：（2 分×5）

1．采用多道程序设计的主要优点是什么？

2．什么是 SPOOLing 技术？

3．叙述“打开（OPEN）”和“关闭（CLOSE）”文件操作的意义。

4．有哪些对空闲盘块的管理方案？UNIX 系统采用的是什么？

5．什么是死锁？对死锁问题有哪些对策？

十．（5 分）在 UNIX 系统中，将进程控制块（PCB）和文件控制块（FCB）各分解为哪二个部分？为什么？

十一．（6 分）分页存储管理有效的解决了什么问题？试叙述其实现原理。

十二．（9 分）多个进程共享一个文件，其中只读文件的称之为读者，其余只写的称之为写者，读者可以同时读，但是写者只能独立地写。请

1．说明进程间得相互制约关系；应设置哪些信号量；

2．用 P、V 操作写出其同步算法；

3．修改上述的同步算法，使得它对写者优先，即一旦有写者到达，后续的读者都必须等待，而无论是否有读者在读文件。

中国科学院软件研究所

一九九五年招收硕士学位研究生入学考试试题答案

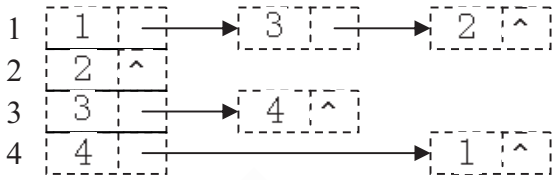
试题名称：软件基础

一．

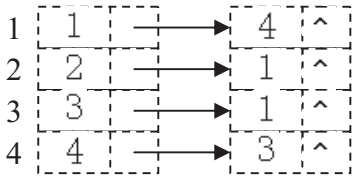
1．邻接矩阵：

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

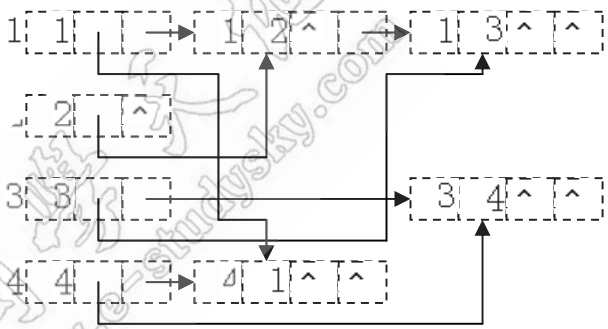
2．邻接表



3．逆邻接表



4.十字链表



二．解决该问题的一个 C 语言程序如下：

```
main（）
 {   int M[100];
     int yi，zi，j，y，z;
     M[0]=1;   j=1;
     y=3;     yi=0;     z=4;     zi=0;
     for(j=1；j<100;j++）
      {
       if (y<z)   { M[j]=y;   yi++;   y=2*M[yi]+1;   }
       else     if   (y==z)
                    { M[j]=y;   yi++;   zi++;
                      y=2*M[yi]+1;     z=3*M[zi]+1;}
                else {                                    /* (y>z) */
                       M[j]=z;   zi++;   z=3*M[zi]+1;}
      }
     for (j=0;j<100;j++)
       printf(“%d ”,M[j]);
 }
```

三．1．对半查找程序只适用于以向量（数组）做存储结构的有序表。

2．程序 A 和 B 正确，C 错误。

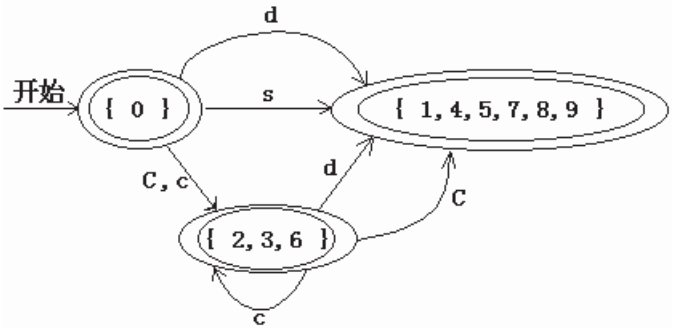
程序 B 的效率更高一些。

四．快速排序的非递归程序用 PASCAL 语言描述如下，要排序的元素 n 个。

```
type index   =   0..n;
type item    =   record
                   key   :   integer;
                   其它项;
                 end

//----------------------procedure START----------------------------------
    procedure quicksort
      const m = 12;          /*  栈大小  */
      var   i，j，l，r ：index;
                x，w ：item;
                    s ：0..m;
                stock ：array [1..m] of   record
                                            l , r : index
                                          end;
                    a : array [1..n] of item;
      begin
        s:=1;   stack[2].l:=1;    stack[1].r:=n;
        repeat
          l:=stack[s].l;          r:=stack[s].r;          s:=s-1;
          repeat
            i:=1;          j:=r;          x:=a[(l+r) div 2];
            repeat
              while a[i].key < x.key do i:=i+1;
              while x.key < a[j].key do j:=j-1;
              if i<=j then
                        begin
                          w:=a[i];     a[i]:=a[j];     a[j]:=w;
                          i:=i+1;      j:=j-1;
                        end
            until   i>j;
            if i<r then
                      begin
                        s:=s+1;          stack[s].l:=i;          stack[s].r:=r
                      end
            r:=j
          until   l>=r
        until   s=0
      end {quicksort}
```

五．



六．1．S：无符号数，　　I：无符号整数，　　R：无符号实数，
W：无符号实数的小数点前面部分，F：无符号实数的小数点后面部分。
2．当第一个终结符是 **d** 时，不知应把 **d** 归约成 I 呢，还是空归约成 W。

七．

S’ ⟶ S		print(S.val)
S ⟶ L_1.L_2		S.val:=L_1.val+L_2.val/2
S ⟶ L		S.val:=L.val
L ⟶ L_1B		L.val:=L_1.val*2 + B.val;
		L.length:=L_1.length + 1
L ⟶ B		L.val:=B.val　　L.length:=2
B ⟶ **1**		B.val:=1
B ⟶ **0**		B.val:=0

八．1．（简单回答）
根据 C 的存储分配特点，被调用过程总能知道第一个实参的位置，printf 的实现是从第一个参数分析还有几个参数。

2．执行前常数区是

```
cp1     cp2
↓       ↓
 12345\0abcdefghij\0
```

strcpy 执行后，该区域变成了：

```
 abcdefghij\0fghij\0
↑       ↑
cp1     cp2
```

九．1．多道程序设计通过将用户的 CPU 请求和 I/O 请求重叠起来的办法，提高了 CPU 的使用效率。
2．它使用直接存取的大容量磁盘作为缓冲，将一个可共享的磁盘空间，改造成若干台输入设备和输出设备，并使得 I/O 设备与 CPU 并行操作。
3．OPEN：告诉系统所指定的文件将变成活跃，其文件说明（或文件目录）复制到内存中；
CLOSE：告诉系统，用户不再使用指定的文件，其文件说明不需保存在内存中

4. 空闲文件目录，空闲盘块链，位示图，成组链接法。

5. 死锁是指在系统中，有二个后多个进程无限期的等待永远不会发生的条件的一种系统状态。

对待死锁的对策有：死锁的预防、避免、死锁的检测和解除。

十. PCB➔ prec 结构，含常用信息，常驻内存；
user 结构，含进程运行时所用信息，可调入/调出内存。

目的：节省内存空间。

FCB➔ 名号 ： 含文件名及编号 i；
小结点： 对应 i 的其它文件控制信息。

目的： 加快检索速度，便于文件的共享。

十一. 1. 解决了存储器零头（碎片）问题。

2. 实现要点：

a. 存储器等分为块，称之为存储块（页架），是分配的单位，其大小为 2^n。

b. 作业地址空间分页，页与块的大小相等。

c. 页表、作业在物理空间上不要求连续存放，存放页表表目，建立页与块的对应关系。

d. 地址变换机构，动态地实现逻辑地址到物理地址的映射。

十二. a. 读者之间同时读；读者与写者之间互斥进行；写者之间互斥进行。

设置 mutex1：各读者进程使用计数器 rc 时的互斥信号量；
mutex2：写者之间、写者与读者之间互斥使用文件所用的互斥信号量。

b. 用类 Pascal 语言描述算法如下：

```
begin
  mutex1：=1;   mutex2：=1;     rc：=0;
  cobegin
     ┇
    reader: begin
              P(mutex1);
              rc:=rc+1;
              if rc=1 then P(mutex2);
              V(mutex1);
                Reading the file:
              P(mutex1);
                rc:=rc-1;
                if rc=0 then V(mutex2);
              V(mutex1);
            end;
              ┇
```

```
    writer: begin
              P(mutex2);
                Writing the file;
              V(mutex2);
            end;
              ┆
  coend;
end.
```

c. 增加一个信号量 WW，用以写进程到时封锁后续的读者。相应的算法改为：

```
begin
  mutex1：=1;     mutex2：=1;     WW：=1;     rc：=0
  cobegin
    ┆
    reader：  begin
              P (WW);
              P (mutex1);
              rc：=rc+1;
              if rc=1 then P (mutex2);
              V (mutex1);
              V (WW);
                Reading the file;
              P (mutex1);
              rc：=rc-1;
              if rc=0 then V (mutex2);
              V (mutex1);
            end;
    writer：  begin
              P (WW);
              P (mutex2);
                 Writing the file;
              V (mutex2);
              V (WW);
            end;
  coend;
end。
```