一、填空题（每空 1 分，共 20 分）

1. __不包含任何字符（长度为 0）的串__称为空串；____由一个或多个空格（仅由空格符）组成的串____称为空白串。

2. 设 S=“A;/document/Mary.doc”，则 strlen(s)=____20______，“/”的字符定位的位置为_____3_____。

4. 子串的定位运算称为串的模式匹配；_被匹配的主串_____称为目标串，__子串____称为模式。

5. 设目标 T=”abccdcdccbaa”，模式 P=“cdcc”，则第__6____次匹配成功。

6. 若 n 为主串长，m 为子串长，则串的古典（朴素）匹配算法最坏的情况下需要比较字符的总次数为__(n-m+1)*m__。

7. 假设有二维数组 $A_{6\times8}$，每个元素用相邻的 6 个字节存储，存储器按字节编址。已知 A 的起始存储位置（基地址）为 1000，则数组 A 的体积（存储量）为____288 B____；末尾元素 A_{57} 的第一个字节地址为___1282______；若按行存储时，元素 A_{14} 的第一个字节地址为__(8+4)×6+1000=1072_____；若按列存储时，元素 A_{47} 的第一个字节地址为__(6×7＋4)×6＋1000）＝1276______。

(注：数组是从 0 行 0 列还是从 1 行 1 列计算起呢？由末单元为 A_{57} 可知，是从 0 行 0 列开始！)

8. 设数组 a[1…60, 1…70]的基地址为 2048，每个元素占 2 个存储单元，若以列序为主序顺序存储，则元素 a[32,58]的存储地址为___8950______。

答：不考虑 0 行 0 列，利用列优先公式： $LOC(a_{ij})=LOC(a_{c_1,c_2})+[(j-c_2)*(d_1-c_1+1)+i-c_1)]*L$

得：$LOC(a_{32,58})=2048+[(58-1)*(60-1+1)+32-1]]*2 = 8950$

9. 三元素组表中的每个结点对应于稀疏矩阵的一个非零元素，它包含有三个数据项，分别表示该元素的____行下标_____、___列下标______和____元素值______。

10.求下列广义表操作的结果：

（1） GetHead【((a,b),(c,d))】===____(a, b)_______; //头元素不必加括号

（2） GetHead【GetTail【((a,b),(c,d))】】===__(c,d)_____;

（3） GetHead【GetTail【GetHead【((a,b),(c,d))】】】===____b___;

（4） GetTail【GetHead【GetTail【((a,b),(c,d))】】】===_____（d）_______;

二、单选题（每小题 1 分，共 15 分）

（ B ）1. 串是一种特殊的线性表，其特殊性体现在：

A．可以顺序存储　　　B．数据元素是一个字符

C．可以链式存储　　　D．数据元素可以是多个字符

（ B ）2. 设有两个串 p 和 q，求 q 在 p 中首次出现的位置的运算称作：

A．连接　　B．模式匹配　　C．求子串　　D．求串长

（ D ）3. 设串 s1=’ABCDEFG’，s2=’PQRST’，函数 con(x,y)返回 x 和 y 串的连接串，subs(s, i, j)返回串 s 的从序号 i 开始的 j 个字符组成的子串，len(s)返回串 s 的长度，则 con(subs(s1, 2, len(s2)), subs(s1, len(s2), 2))的结果串是：

A．BCDEF　　B．BCDEFG　　C．BCPQRST　　D．BCDEFEF

解：con(x,y)返回 x 和 y 串的连接串，即 con(x,y)＝‘ABCDEFGPQRST’；
subs(s, i, j)返回串 s 的从序号 i 开始的 j 个字符组成的子串，则
subs(s1, 2, len(s2))＝subs(s1, 2, 5)=’BCDEF’;　subs(s1, len(s2), 2)＝subs(s1, 5, 2)=’EF’;
所以 con(subs(s1, 2, len(s2)), subs(s1, len(s2), 2))＝con(’BCDEF’, ’EF’)之连接，即 BCDEFEF

（ A ）4. 假设有 60 行 70 列的二维数组 a[1…60, 1…70]以列序为主序顺序存储，其基地址为 10000，每个元素占 2 个存储单元，那么第 32 行第 58 列的元素 a[32,58]的存储地址为______。（无第 0 行第 0 列元素）

A．16902　B．16904　C．14454　D．答案 A, B, C 均不对

答：此题与填空题第 8 小题相似。（57 列×60 行＋31 行）×2 字节＋10000=16902

(B)5. 设矩阵 A 是一个对称矩阵，为了节省存储，将其下三角部分（如下图所示）按行序存放在一维数组 B[**1, n(n-1)/2**]中，对下三角部分中任一元素 $a_{i,j}$(i≤j), 在一维数组 B 中下标 k 的值是：

$$A=\begin{bmatrix} a_{1,1} & & & \\ a_{2,1} & a_{2,2} & & \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

A．i(i-1)/2+j-1　B．i(i-1)/2+j　C．i(i+1)/2+j-1　D．i(i+1)/2+j

解：注意 B 的下标要求从 1 开始。
先用第一个元素去套用，可能有 B 和 C；
再用第二个元素去套用 B 和 C，B=2 而 C=3（不符）；
所以选 B

6 …的最确切的解答，把相应编号写在答卷的对应栏内 …标的范围是 1 到 5，每个数组元素用相邻的 4 个字节 …的第一个字节的地址是 0。
存储数组 A 的最后一个元素的第一个字节的地址是__A__。若按行存储，则 A[3,5]和 A[5,3]的第一个字节的地址分别是__B__和__C__。若按列存储，则 A[7,1]和 A[2,4]的第一个字节的地址分别是__D__和__E__。
供选择的答案：
A～E：①28　② 44　③ 76　④ 92　⑤ 108
⑥ 116　⑦ 132　⑧ 176　⑨ 184　⑩ 188
答案：ABCDE=8, 3, 5, 1, 6

7. 有一个二维数组 A，行下标的范围是 1 到 6，列下标的范围是 0 到 7，每个数组元素用相邻的 6 个字节存储，存储器按字节编址。那么，这个数组的体积是__A__个字节。假设存储数组元素 A[1,0]的第一个字节的地址是 0，则存储数组 A 的最后一个元素的第一个字节的地址是__B__。若按行存储，则 A[2,4]的第一个字节的地址是__C__。若按列存储，则 A[5,7]的第一个字节的地址是__D__。
供选择的答案
A～D：①12　② 66　③ 72　④ 96　⑤ 114　⑥ 120
⑦ 156　⑧ 234　⑨ 276　⑩ 282　（11）283　（12）288
答案：ABCD=12, 10, 3, 9

三、简答题（每小题 5 分，共 15 分）

1. 已知二维数组 Am,m 采用按行优先顺序存放，每个元素占 K 个存储单元，并且第一个元素的存储地址为 Loc(a11)，请写出求 Loc(aij)的计算公式。如果采用列优先顺序存放呢？
解：公式教材已给出，此处虽是方阵，但行列公式仍不相同；
按行存储的元素地址公式是：　Loc(aij)= Loc(a11) +[(i-1)*m+(j-1)] * K
按列存储的元素地址公式是：　Loc(aij)= Loc(a11) +[(j-1)*m+(i-1)] * K

2.递归算法比非递归算法花费更多的时间，对吗？为什么？

答：不一定。时间复杂度与样本个数 n 有关，是指最深层的执行语句耗费时间，而递归算法与非递归算法在最深层的语句执行上是没有区别的，循环的次数也没有太大差异。仅仅是确定循环是否继续的方式不同，递归用栈隐含循环次数，非递归用循环变量来显示循环次数而已。

四、计算题（每题 5 分，共 20 分）

1. 设 s='I AM A STUDENT', t='GOOD', q='WORKER', 求 Replace(s,'STUDENT',q) 和 Concat(SubString(s,6,2), Concat(t,SubString(s,7,8)))。

解：① Replace(s,'STUDENT',q)＝**'I AM A WORKER'**

② 因为 SubString(s,6,2)＝ 'A '；SubString(s,7,8)＝ ' STUDENT'

Concat(t,SubString(s,7,8))＝'GOOD STUDENT'

所以 Concat(SubString(s,6,2), Concat(t,SubString(s,7,8)))＝ 'A GOOD STUDENT'

2. （P60 4-18）用三元组表表示下列稀疏矩阵：

$$(1)\begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&3&0&0&0&8&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&6&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&5 \\ 2&0&0&0&0&0&0&0 \end{bmatrix} \qquad (2)\begin{bmatrix} 0&0&0&0&0&-2 \\ 0&0&0&0&9&0 \\ 0&0&0&0&0&0 \\ 0&0&5&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&3&0 \end{bmatrix}$$

解：参见填空题 4. 三元素组表中的每个结点对应于稀疏矩阵的一个非零元素，它包含有三个数据项，分别表示该元素的__行下标__、__列下标__和__元素值__。

所以（1）可列表为：

8	8	5
3	2	3
3	6	8
5	4	6
7	8	5
8	1	2

（2）可列表为：

6	6	4
1	6	-2
2	5	9
4	3	5
6	5	3

3. （P60 4-19）下列各三元组表分别表示一个稀疏矩阵，试写出它们的稀疏矩阵。

$$(1)\begin{bmatrix} 6 & 4 & 6 \\ 1 & 2 & 2 \\ 2 & 1 & 12 \\ 3 & 1 & 3 \\ 4 & 4 & 4 \\ 5 & 3 & 6 \\ 6 & 1 & 16 \end{bmatrix} \qquad (2)\begin{bmatrix} 4 & 5 & 5 \\ 1 & 1 & 1 \\ 2 & 4 & 9 \\ 3 & 2 & 8 \\ 3 & 5 & 6 \\ 4 & 3 & 7 \end{bmatrix}$$

解：（1）为 6×4 矩阵，非零元素有 6 个。（2）为 4×5 矩阵，非零元素有 5 个

0	2	0	0
12	0	0	0
3	0	0	0
0	0	0	4
0	0	6	0
16	0	0	0

1	0	0	0	0
0	0	0	9	0
0	8	0	0	6
0	0	7	0	0

五、算法设计题（每题 10 分，共 30 分）

1. 编写一个实现串的置换操作 Replace(&S, T, V)的算法。

解：

```
int Replace(Stringtype &S,Stringtype T,Stringtype V);//将串 S 中所有子串 T 替换为
V,并返回置换次数
{
  for(n=0,i=1;i<=Strlen(S)-Strlen(T)+1;i++) //注意 i 的取值范围
    if(!StrCompare(SubString(S,i,Strlen(T)),T)) //找到了与 T 匹配的子串
    { //分别把 T 的前面和后面部分保存为 head 和 tail
      StrAssign(head,SubString(S,1,i-1));
      StrAssign(tail,SubString(S,i+Strlen(T),Strlen(S)-i-Strlen(T)+1));
      StrAssign(S,Concat(head,V));
      StrAssign(S,Concat(S,tail)); //把 head,V,tail 连接为新串
      i+=Strlen(V); //当前指针跳到插入串以后
      n++;
      n++;
    }//if
  return n;
}//Replace
```

分析:i+=Strlen(V);这一句是必需的,也是容易忽略的.如省掉这一句,则在某些情况下，会引起不希望的后果,虽然在大多数情况下没有影响.请思考:设 S='place'，T='ace'，V='face',则省掉 i+=Strlen(V);运行时会出现什么结果?

2. **试设计一个算法，将数组 An 中的元素 A[0]至 A[n-1]循环右移 k 位，并要求只用一个元素大小的附加**

存储，元素移动或交换次数为 O(n)

解：

分析:要把 A 的元素循环右移 k 位,则 A[0]移至 A[k],A[k]移至 A[2k]......直到最终回到 A[0].然而这并没有全部解决问题,因为有可能有的元素在此过程中始终没有被访问过,而是被跳了过去.分析可知,当 n 和 k 的最大公约数为 p 时,只要分别以 A[0],A[1],...A[p-1]为起点执行上述算法,就可以保证每一个元素都被且仅被右移一次,从而满足题目要求.也就是说,A 的所有元素分别处在 p 个"循环链"上面.举例如下:

n=15,k=6,则 p=3.

第一条链:A[0]->A[6],A[6]->A[12],A[12]->A[3],A[3]->A[9],A[9]->A[0].　/已“顺便”移动了 A[6]、A[12]…

第二条链:A[1]->A[7],A[7]->A[13],A[13]->A[4],A[4]->A[10],A[10]->A[1].

第三条链:A[2]->A[8],A[8]->A[14],A[14]->A[5],A[5]->A[11],A[11]->A[2].

恰好使所有元素都右移一次.

虽然未经数学证明,但作者相信上述规律应该是正确的. 程序如下：

```
void RSh(int A[n],int k)//把数组 A 的元素循环右移 k 位,只用一个辅助存储空间
{
  for(i=1;i<=k;i++)
    if(n%i==0&&k%i==0) p=i;//求 n 和 k 的最大公约数 p
  for(i=0;i<p;i++)
  {
    j=i;l=(i+k)%n;temp=A[i];
    while(l!=i)
    {
      A[j]=temp;
      temp=A[l];
      A[l]=A[j];
      j=l;l=(j+k)%n;
    }// 循环右移一步
    A[i]=temp;
  }//for
}//RSh
```