西南交通大学 2008 年硕士研究生入学考试试题解析

试题名称：程序设计与数据结构

一、填空题

1. 2.0

解析：以为 x,y 都是 int 型，所以 x/y 的值为 0

2. \0 0

解析：\0 是字符串的结束符，其 ASCⅡ值为 0

3. -3

4. 0

解析：从左到右，首先 a>b 为真，所以值为 1，再判断 1>c 显然为假，所以值为 0.

5. doubel a[3][2]={1,0,3,8,2,6,3,3,5,0,9,8}

1.0 3.8 2.6

3.3 5.0 9.8

解析：这是一个三行二列二维数组，因此数组定义为 doubel a[3][2]={1,0,3,8,2,6,3,3,5,0,9,8}

6. 6 个 \0

解析：char 类型是一个字符占用一个字节，由于字符串还有一个结束符\0，故占用的内存为 6 个字节，数组下表从零开始，S[5]为\0

7. a[3]的值 a[0]的值+3

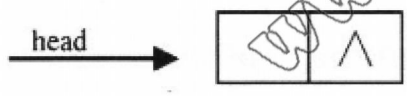
解析：*p=a 表示将数组的首地址给了指针 p，p+3 表示指针顺着移动了 3，即 a[3]的地址，所以*(p+3)表示 a[3]的值，p 表示 a[0]的地址，则*p 表示数值 a[0]，所以*p+3 表示 a[0]的值加三。

8. n-1

解析：若队列全部装入元素最多为 n 个元素，这里浪费一个单元格莱表示队满，则队满共有 n-1 个元素

9. head→next==NULL

解析：



故 head→next==NULL

10. 326

解析：locA[b][12]=A[0][0]+(6+12*10)*1=200+126=326（因为以列序为主序）

11. 2^{k-1} 2^k-1

解析：深度为 k 的二叉树至少有 2^{k-1} 个结点，最多为满二叉树，2^k-1 个结点

12. $n_0=n_2+1$

解析： $\begin{cases} n = n_0 + n_2 + n_1 \\ n = 2n_2 + n_1 + 1 \end{cases} \Rightarrow n_0 = n_2 + 1$

13. 1

解析：无向图的邻接矩阵是对称的，A[i][j]=A[j][i]

14. n-1

解析：当元素有序排列时，只需进行一趟排序，在排序过程中进行 n-1 次关键字的比较

15. 顺序 有序

解析：折半查找的条件，课本 P221

二、单项选择题

1. 选 D

解析：自动向精度大的那个类型转换，double 的精度最大，char,int,float,double 一次增大

2. 选 C

解析：if 语句的嵌套，明白 x>y?x:y 表达式的含义，如果 x>y 为真，则表达式值为 x，否则为 y

3. 选 A

解析：参数名可以缺省，但参数类型不能缺省

4. 选 D

5. 选 A

解析：注意计算字符串的长度时不包括\0

6. 选 D

解析：*p 为一级指针，**q 为二级指针，其中 p，*q 为指针地址，只能给变量或指针变量赋值，不能给地址赋值，所谓地址，就是由 ASCII构成，目的要区分地址中存储的值

7. 选 D

解析：a 为这个数组的首地址，a[3]指向第四个元素，所以 p 指向第四个元素，p[5]是从 p 后的五个，也就是第九个元素，所以 b=9

8. 选 B

解析：int main( )是标准形式，表示程序将会返回一个值，操作系统可以根据这个返回值判断程序的执行状态，void main( )表示程序不返回值，这样操作系统无法通过程序的返回值判断其运行状态。在 c 语言中，为说明返回值的函数类型默认为 int

9. 选 C

解析：这里需要传一个地址参数，ABD 都表示地址

10. 选 C

解析：（！x）和 x！=0 的意思是一样的，因为 c 语言里非零即真，为零即假

11. 选 B

12. 选 D

13. 选 C

解析：i++是循环+1，在接受完字符时+1，scanf 参数必须是地址

14. 选 D

解析：见课本 P336

15. 选 C

16. 选 B

解析：A 顺序存储于链式存储各有优缺点，不能单纯的说谁比谁好，CD 说反了

17. 选 C

解析：C 中 a 不可能比 b 先出栈

18. 选 C

解析：因为 P1=n,故该输出序列是原序列的逆序，故 Pi 为 n-i+1

19. 选 A

解析：队列中的元素个数为（rear-front+m）%m

20. 选 D

解析：subs(s1,2,len(s2)):BCDEF　　subs(s1,len(s2),2):ef

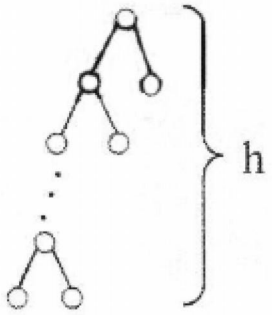
con(subs(s1,2,len(s2)),subs(s1,len(s2),2)):BCDEFEF

21. 选 A

22. 选 B

解析：树的形状如下：

故所含的结点数至少为 2h-1

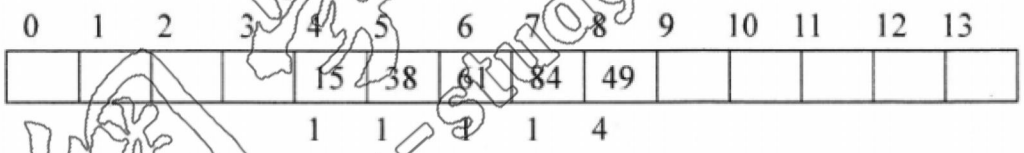


23. 选 C

解析：要连接 n 各顶点的无向图，至少需要 n-1 条边

24. 选 D

解析：



25. 选 C

26. 选 A

解析：邻接矩阵表示的有向图，每一列表示顶点的入度，每一行表示该行顶点的出度。

27. 选 D

解析：第二趟之前剩余 n-1 个元素，第三趟之前剩余 n-2 个元素……第 i 趟之前剩余 n-(i-1)个元素

28. 选 D

解析：无论有序还是无序，堆排列的时间复杂度均为 $o(n\log_2 n)$

29. 选 B

解析：画出折半查找判定树，见课本 P220

30. 选 B

解析：当数据有序时，快速排序就蜕化为冒泡排序，时间复杂度为$O(n^2)$

三、阅读程序，按提示给出结果

1. 求数组 a 中元素的平均数

2. 2 1

解析：case1 语句后，未跟 break，所以继续执行 case2 语句。

3. 7

4. 24

5. -893

四、程序填空

1. &a[i] a[i]

2. s[i]>='0'&&s[i]<='9' '\0'

3. （low+high）/2 high=mid-1 mid+1

解析：见 P220

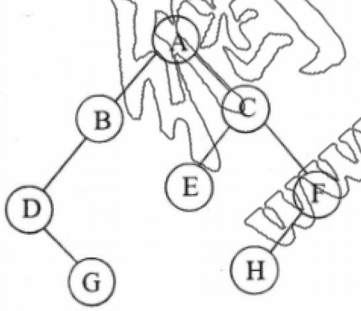
4. L.r[0]=L.r[i] L.r[j+1]=L.r[j] L.r[j+1]=L.r[0]

解析：见 P265

五、简要回答题

1.**解析：**后序遍历：g d b e h f c a

由前序和中序遍历画出对应的二叉树为：

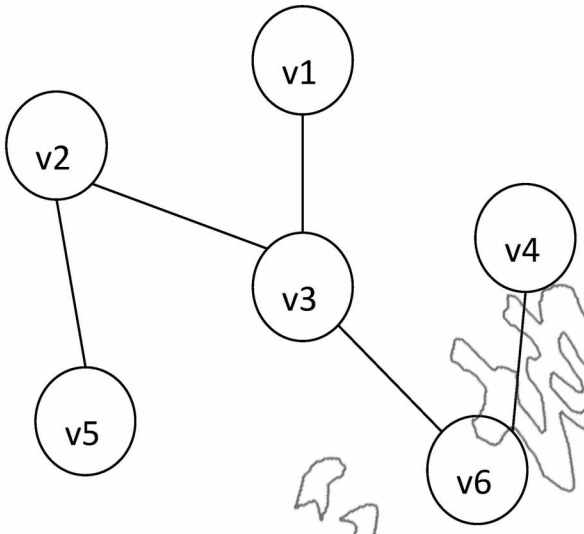


2.**解析：**随着队头元素的不断出队，数组会产生一些空单元，入队只能在队尾进行，使得这些空单元无法使用，造成“假溢出”，采用循环队列的形式可以避免这种现象。

3.**解析：**在有向图的邻接矩阵中，某一行全为 0 的顶点是出度为 0 的顶点，某一列全为 0 的顶点是入度为 0 的顶点，在有向图的邻接表中，若顶点对应的链表只有表头结点，说明该顶点的出度为 0.

4.解析;在顺序存储结构中，逻辑上相邻的数据元素物理位置上也相邻，只要确定了起始地址，就能确定线性表中任一数据元素的存储位置。

5.**解析**：最小生成树：



六、程序设计

1.**解析**：

```
#include"stdio.h"
void Func(int*a,int*n)
{
int sum=0;
for(int i=1;i<=1000;i++)
{
    if((i%7==0||i%11==0)&&i%77)
      {
          a[sum++]=i;
      }
    }
     *n=sum;
}
int main(int argc,char*argv[])
{
   int a[300];
   int sum=300;
   Func(a,&sum);
   for(int i=0;i<sum;i++)
   {
       printf("%d    ",a[i]);
   }
   printf("数组的个数=%d\n",sum);
```

}指的数组当中，通过 n 返回这些数的个数

2.**解析：**

```
flost myfunction(int n,int x)
{if(0==n)return 1;
else if(1==n)return 1;
else return(2*n-1)*x-myfunction(n-1, x)-(n-1)*myfunction(n-2,x)/n);}
int main(int argc,char*argy[])
{int n,x;float renult;;
printf("please input n,x:");
scanf("%d,%d",&n,&x);
result=myfunction(n,x);
printf("the result is:%f\n",result);
return 0;
}
```

3.**解析：**

```
void Del-Same(Linklist*L)
 {LNode*p=L→next,*q;
 while(p→next!=NULL)
        {q=p→next;
        if(p→data==q→data)
          {p→next=q→next;free(q);}
       else p=p→next;
        }
    }
```

4.**解析：**

```
int Depth(BiTree T)
{if(T==NULL)return 0;
else
    n=Depth(T→lchild);
    m=Depth(T→rchild);
return(m>n?m:n)+1;
}
```