西南交通大学 2012 年全日制硕士研究生入学试题解析
试题名称：数据结构与程序设计

一、单项选择题

1.选 C

解析：见课本 P7

2.选 C

解析：这是一个逗号运算符，最后表达式的值是最右边的一个

3.选 C

解析：课本 P54

4.选 D

解析：见课本 P77，可以用%1x 输出长整型数也可以输出字段的宽度如%12x

5.选 C

解析：见课本 P266

6.选 A

7.选 C

解析：当 i=4 时仍继续执行一次 i++操作，所以 i=5

8.选 D

解析：数据下标越界，应该为 i<3。

9.选 D

解析：字符数组可以用字符串常数初始化，可以去掉花括号，也可用单个字符初始化，D 项数组长度不足。

10.选 A

解析：见课本 P180

11.选 D

解析：在复合语句内定义的变量只在本复合语句内有效，只在本复合语句内才能引用它们

12.选 C

13.选 C

解析：见课本 P339

14.选 C

解析：课本 P5，数据结构的表示

15.选 A

解析：顺序表可以随机存取，故可以存取任一指定序号的元素并且在最后进行插入、删除操作的时间复杂度均为 O（1）

16.选 B

解析：在 n 个结点的有序单链表中插入结点需要查找插入结点的位置，故时间复杂度为 O（n）

17.选 C

解析：因为 p_1=n，该输出序列只能是原序列的逆序，故 p_i 为 n-i+1

18.选 B

解析：见课本 P78

19.选 D

20.选 C

解析：树元素之间的关系是一对多的关系，故适合用来表示具有分支层次关系的数据

21.选 C

解析：具有 n 个结点的二叉树的形态总数为 $\frac{1}{n+1}C_{2n}^{n}$

22.选 D

解析：先根遍历 x 在 y 之前，x 或者是根，或是 y 左侧的点，后根遍历 x 在 y 之后，x 或者是根或者是 y 右侧的结点

23.选 B

解析：一棵有 n 个顶点的生成树有且仅有 n-1 条边，生成树即该图的最小连通分量。

24.选 A

解析：顺序查找对表的结构物任何要求，无论记录是否按关键字有序均可应用。

25.选 D

解析：快速排序，堆排序是不稳定的排序方法，直接插入排序的时间复杂度为 O（n^2）,故为归并排序。

二、填空题

1. 3

解析：\0,1,0 共三个。

2. x=-14

解析：从右向左一次计算，先计算 x+x 等于 14，即 x-=14，所以 x=-7，再计算 x+=-7，所以 x=-14。

3. 1

解析：因为 x，y，z 均为非零，所以 x&&y 值为 1，x||y 值为 1，所以（x&&y）==(x||y) 的值为 1。

4. Sihiu。

解析：读入一个字符，先判断是不是‘？’，如果不是，输出它后面的一个字符（比如‘q’后面是‘u’），是问号就停止了。

5. 0 6

解析：初始化的元素如下：

$$\begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 4 & 6 & 8 & 10 \end{pmatrix}$$

6. 函数的首部 函数体

解析：见课本 P6

7. 2

解析：swap(int x,int y)实现对 x 和 y 值的交换，但从函数定义看，只是对形参进行了数据修改，不影响实参数据，因此结果不会变化，仍然是 a[0]=1,a[1]=2.

8. 9.0

解析：A=4，x=a=4.5，所以 A*x/2=4*4.5/2=9.0

9. （12）

解析：P 声明为一个 int 型指针，但是没有指向具体的内存空间，赋值操作是非法的，运行时要报错。

10. badABCD

解析：strcat(sp“ABCD”)意思是将 ABCD 字符串添加到 sp 的结尾处，而 sp++后，sp 指

针指向第二个字符 b。

11. 二进制 ASCII

解析：见课本 P330。

12. 数据对象 D 上关系的有限集 对 D 的基本操作集

解析：见课本 P5、P8

13. q→next

解析：用 q 所指结点的值将 p 的值覆盖，然后再删除 q 结点，这样就避免了找 p 的前驱结点。

14. （r+n-f）%n

15. 栈

解析：见课本 P52

16. 129

解析：n=n0+n1+n2，其中 n0=50，n1=30，n2=n0-1=49 所以 n=129。

17. 二叉树

解析：孩子兄弟表示法又称二叉树表示法或二叉链表表示法，以二叉链表作为树的存储结构。

18. 邻接矩阵 邻接表

解析：邻接矩阵中边是唯一的，故邻接矩阵也唯一，在邻接表中，依附于某个顶点的顶点在边表中的顺序可不同。

19. 插入 删除

解析：对查找表的操作包括：查找元素，检索元素的属性，插入元素，删除元素，只做前两种操作的查找表为静态查找表，在查找过程中，同时插入删除某个元素的表为动态查找表。

20. 21

解析：顺序查找的分块查找的平均查找长度 $ASL_{bs}=\frac{b+1}{2}+\frac{s+1}{2}=\frac{1}{2}(\frac{n}{s}+s)+1$ 其中 n 为表长，s 为表中的记录个数，b 为块数。

21. 快速排序 归并排序

解析：辅助空间：归并排序>快速排序，其中均为 O(1)；快速排序平均复杂度为 $O(\log_2 n)$，直接插入排序、选择排序、冒泡排序为 $O(n^2)$，归并排序为 $O(n\log_2 n)$，故选择快速排序。

三、简答题

1.
31 8
34 38

解析：do-while 语句先执行再判断条件。

2.
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1

解析：当 i！=j 时，（i/j)*(j/i)为 0，当 i=j 时，a[i][j]=1。

3.
4 5 6
1 5 6

解析：b 和 c 均为指针变量，分别指向 5 和 6 的地址。

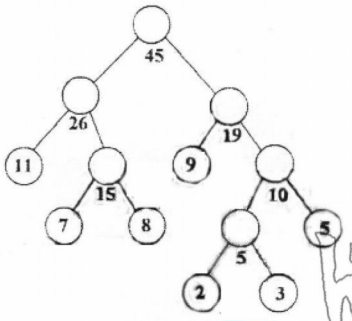
4. 0 1 3 6

解析：t 为静态变量。每次程序执行完保留执行后的结果。

5. ① ②

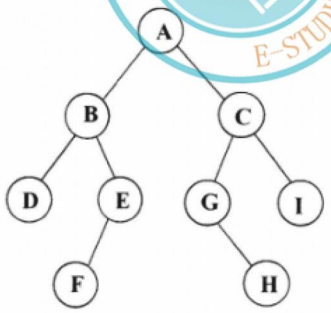
6. **解析：**3；s1,s2 进栈，s2 出栈，s3、s4 分别进栈再出栈，s5、s6 进栈，s6、s5 出栈，s1 出栈，故顺序栈的容量至少应为 3.

7. **解析：**哈夫曼树如下：



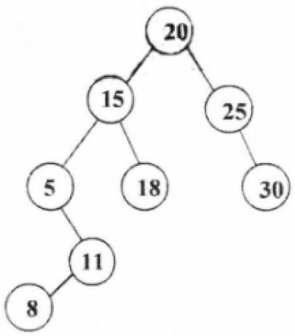
带权路径长度：
WPL=（11+9）*2+（7+8+5）*3+（2+3）*4=120

8. **解析：**



前序遍历：ABDEFCGHI

9. **解析：**二叉排序树如下



对该二叉排序树进行中序遍历即可得到一个有序序列

四、

（1.）解析：

```
#include<stdio.h>
Int main()
{
   int search(int list[],int x);
   int a[10],x,i,y;
   Printf(“输入数组元素”);
    for(i=0;i<10;i++)
       scanf(“%d”,&a[i]);
       printf(“输入 x 的值”);
       scanf(“%d”,&x);
       y=search(a,x);
       if(y==-1)printf(“Not found”);
       else printf(“%d”,y);
       return 0;
}
int search(int list[],int x)
{
   int i,j;
   for(i=0,i<10;i++)
  {if(x==list[i])return(i+1);}
}
  return-1;
}
```

2.解析：

```
#include “stdio.h”
 int main()
{
       int n,m,r;
       printf(“请输入一个四位正整数”);
       scanf(“%d”, &n);
```

```
        m=n;
        printf(“\n 反序输出为：”，n)；
        while(m)
        {
           r=m%10;
           printf(“%d”,r)；
           m=m/10；
        }
```

3.①：!t;
②:bt=(BiTree)malloc(sizeof(BiNode));
③:bt→data=t→data;
④:return bt;

4.解析：

```
#include<stdio.h>
#define INIF_SIZE 20
#define INCR_SIZE 10
unsigned int strlen(char*str)
{
   unsigned int i;
   for(i=0;str[i++]!=‘\0’)；
   return(i-1);
   }
   int IsPalindrome(char*str)
   {
       int len=strlen(str);
       int i=0;
        for(;i<len/2;i++)
        {if(str[i]!=str[len-1-i])return 0;}
        return 1;
}
void   main()
{
    char*str=(char*)malloc(INIT_SIZE*sizeof(char));
    char ch;
    int i=0;
    int len=INIT_SIZE;
    while(ch=getchar())
    {
        if(ch==’@’)
          {
            str[i]=’\0’;
            break;
```

```
        }
        if(i<len-1){str[i]=ch}
        else
        {
str=(char*)realloc(str,(len+INCR_SIZE)*sizuof(char));
str[i]=ch;
Len+=INCR_SIZE;
}
i++;
}
if(IsPalindrome(str)!=0){printf(“YES\n”);}
   else {printf(“NO\n”);}
}
```