装订线内

不要答题

北京大学信息科学技术学院考试试卷

考试科目： 数据结构与算法 A **姓名：**_____ **学号：**_____

考试时间： 2014 年 1 月 1 日 **任课教师:**_____

题号	一	二	三	四	总分
分数					
阅卷人					

北京大学考场纪律

1、考生进入考场后，按照监考人员安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，试题共 5 页。

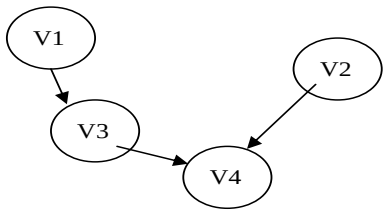
得分	

一、填空（30 分，选择题都是单选）

1. （2 分）当各边上的权值_____时，BFS 算法可用来解决单源最短路径问题。

   A．均相等　　B．均互不相等　　C．不一定相等

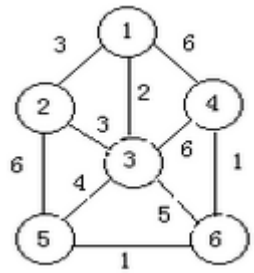
2. （4 分）有向图 G 如下图所示：



   （1）写出所有可能的拓扑序列：_____。

   （2）添加一条弧_____之后，则仅有惟一的拓扑序列。

3. （3 分）设有如下所示无向图 G，用 Prim 算法构造最小生成树所走过的边的集合_____。



4. （2 分）如果只想得到 1000 个元素组成的序列中第 5 个最小元素之前的部分排序的序列，用 _____方法最快。

   A．起泡排序　B．快速排列　C．堆排序　D．简单选择排序

5. （2 分）若要求尽可能快地对序列进行稳定的排序，则应选_____

   A．快速排序　　B．归并排序　　C．冒泡排序　　D.归并排序

6. （2 分）在文件局部有序或文件长度较小的情况下，最佳的排序方法

是_____。

A.直接插入排序　　B. 冒泡排序　　C. 直接选择排序　　D.归并排序

7. （2 分）设输入的关键字满足 $K_1>K_2>\ldots>K_n$ ，缓冲区大小为 m ，用置换选择排序方法可产生_____个初始归并段。

8. （4 分）设有序顺序表中的元素依次为 017, 094, 154, 170, 275, 503, 509, 512, 553, 612, 677, 765, 897, 908，则对其进行折半查找时，等概率下搜索成功的平均查找长度和不成功的平均查找长度分别为_____和_____。

9. （3 分）设散列表的长度为 14，哈希函数为 H(key)=key%11，表中已有数据的关键字为 15,38,61,84 共四个，现要将关键字为 49 的结点将加入表中，用二次探查法解决冲突，则放入的位置是_____。

A．8　　B．3　　C．5　　D．9

10. （2 分）在一棵阶为 k 的红黑树中，内部结点最少有_____个；从根到叶的简单路径长度的范围为_____。

11. （2 分）在一棵含有 1023 个关键字的 4 阶 B 树中进行查找（不读取数据主文件），至多读盘_____次。

A. 7　　B. 8　　C. 10　　D. 9

12. （2 分）下列关于红黑树的说法中错误的是 _____。

A. 红黑树上插入操作的最差情况下的时间复杂度为 O（log n)

B. 红黑树上任意节点的左右子树高度差绝对不大于 1

C. 红黑树上删除操作最差情况的时间复杂度为 O(log n)

D. 红黑树上查找操作最差情况下的时间复杂度为 O(log n)

得分

二、 辨析与简答（30 分）

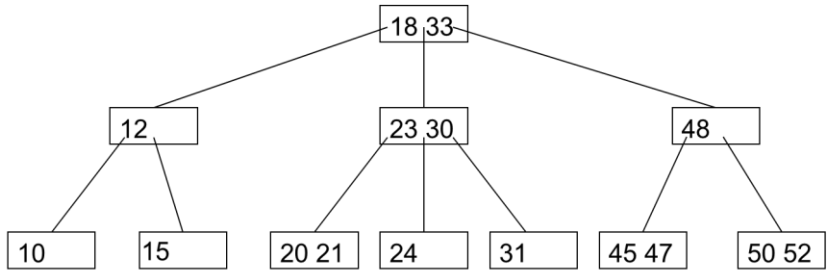
1. （8分）将关键字序列（7、8、30、11、18、9、14）散列存储到散列表中。

散列表是一个下标从0开始的一维数组，散列函数为：H(key) = (key * 3) MOD 7，处理冲突采用线性探测法，要求装填（载）因子为0.7。

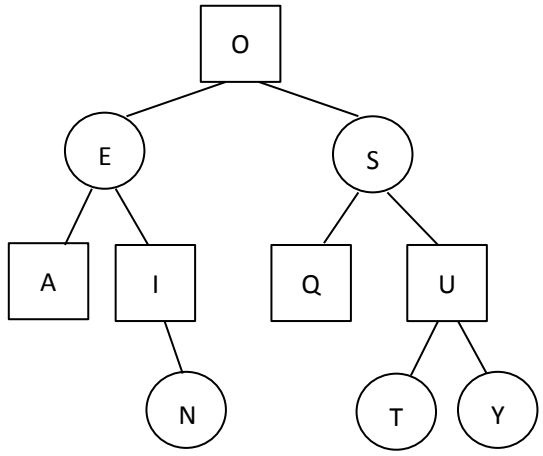
(1) 请画出所构造的散列表。

(2) 分别计算等概率情况下查找成功和查找不成功的平均查找长度。

2. （6分）设有如下一棵3阶B树，请画出在其中插入关键码19后的B树，并给出插入过程中的访外（读写）次数；在此基础上再删除关键码48，请画出删除后的B树及删除过程中的访外（读写）次数。



3. （6分）将下列字符序列 E A S Y Q U E S T I O N 依次插入到初始为空的红黑树（RB-tree）中，请画出最终得到的红黑树。（用圆圈表示红色结点，方框表示黑色结点，外部空叶结点可省略不画）。



4. （4分）利用广义表的Head和Tail运算，把原子d分别从下列广义表中分离出来，L1＝(((((a),b),d),e))；L2＝（a,(b,((d)),e））。

5. （6分）将关键码1，2，3，...，（2k-1）依次插入到一棵初始为空的AVL树中，试证明结果是一棵高度为k的完全满二叉树。

得分

三、算法填空 （30 分）

阅读和完成下面的代码（每空可不限一条语句）。

1. （8 分）简单路径条数。

给出图的 ADT 如下：

```
Class Graph {
public:
    int VerticesNum();
    int EdgesNum();
    Edge FirstEdge(int oneVertex);
    Edge NextEdge(Edge preEdge);
    bool IsEdge(Edge onEdge);
    int FromVertex(Edge oneEdge);
    int ToVertex(Edge oneEdge);
};
```

设计算法过出图 G 中从顶点 i 到顶点 j 之间长度为 len 的简单路径条数:

int GetPathNum_Len(Graph& G, int i, int j, int len);并给出算法代价分析

答案：深度优先，回溯。

```
int visited[MAXSIZE];                    // 初始化为 0
int GetPathNum_Len(Graph& G, int i, int j, int len)  {
    if (填空 1) return 1;     // 找到了一条路径,且长度符合要求
    sum = 0;                             // sum 表示通过本结点的路径数
    visited[i] = 1;
    for (Edge e = G.FirstEdge(i); G.IsEdge(e); e = G.NextEdge(e)) {
        int v = G.ToVertex(e);
        if (!visited[v])
            填空 2 // 剩余路径长减 1
    }// for
    visited[i] = 0;          //本题允许曾经被访问过的结点出现在另一条路径中
    return sum;
} // GetPathNum_Len
```

2. （10 分）以单向链表为存储结构，实现选择排序算法（得到非递减序列）。

```
struct node {
      int key;
      node *next;
};

node* sort_linked_list(node* head) {
   i = head;
   while (i!=NULL) {
        填空 1          ;
      while (j!=NULL) {
        if (  填空 2   )   填空 3      ;
          填空 4          ;
      }
           填空 5          ;
   }
   return head;
}
```

3. （12 分）下面是败者树在内部结点从右分支向上比赛的成员函数实现，请将空缺部分补充完整。

```
template<class T>
void LoserTree<T>::Play(int p, int lc, int rc, int(*winner)(T A[], int b, int c),
int(*loser)(T A[], int b, int c)) {
      B[p] = loser(L, lc, rc);
      int temp1, temp2;
      填空 1;
     while (p>1 &&  填空 2) {
           temp2 = winner(L, temp1, B[p/2]);
           填空 3;
           temp1 = temp2;
           p/=2;
```

```
        }
    填空 4
}
```

得分

四、算法设计与分析 （10 分）

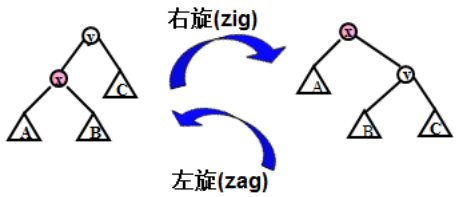
请尽量按照试题中的要求来写高效率的可读算法。应该申明算法思想，在代码种加以恰当的注释。

1. 伸展树是一种自平衡的BST，其结点的数据结构如下：

```
struct TreeNode
{
    int   data;
    TreeNode   * father,* left,* right;
};
```

旋转分为左旋(zag)和右旋(zig)，这两个是对称的。下图就是函数调用示意

```
void zag(TreeNode* y);
void zig(TreeNode* x);
```



另外，能用到的操作是删除以x结点为根的子树 Delete zig(TreeNode* x);

函数Splay(x, f) ，其功能是将 x 旋转为 f 的子结点（前题是 f 一定在 x 的祖先路径上）。例如把 x 旋转到根结点即 Splay(x, NULL)。

```
void Splay (TreeNode *x, TreeNode *f) {                    // 将 x 旋转为 f 的子结点
    while (x->parent != f) {
        TreeNode *y = x-> parent, *z = y-> parent;
        if (z != NULL)   {                                 // 祖父不空
            if (z->lchild == y) {
                if (y->lchild == x)
```

```
                { Zig(y);   Zig(x); }                    // 一字型双右旋
              else  { Zag(x);   Zig(x); }                // x 左旋上来，接着右旋
            } else {
              if (y->lchild == x)
                {   Zig(x);   Zag(x); }                  // x 右旋上来，接着左旋
              else  { Zag(y);   Zag(x); }                // 一字型双左旋
            }
        }
        else {
            if (y->lchild == x)
                Zig(x);                                  // 右单旋
              else Zag(x);                               // 左单旋
          }
    }
    if (x->parent == NULL)
        Root = x;
}
```

请运用上述给定的函数，删除树 rt 中所有大于 u 小于 v 的结点（假设u, v 是Splay树中的结点）。函数原型如下：

```
void DeleteUV(TreeNode* rt, TreeNode* u, TreeNode* v )
```