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：2015 年 11 月 13 日　　**任课教师：**_____

题号	一	二	三	四	五	**总分**
分数						
阅卷人						

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，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、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机等）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，并关闭手机等一切电子设备。

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。考试结束监考人员宣布收卷时，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。不准旁窥、交头接耳、打暗号，不准携带与考试内容相关的材料参加考试，不准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。凡有严重违纪或作弊者，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习纪律管理规定》及其他相关规定严肃处理。

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。

注意事项：

1. 全部题目都在空白答题纸上解答。
2. 本试卷对算法设计都有质量要求，请尽量按照试题中的要求来写算法。否则将酌情扣分。
3. 请申明所写算法的基本思想，并在算法段加以恰当的注释。

以下为试题和答题纸，共 5 页，请把答案写在答题纸。

一、 选择(每题 2 分，共 20 分)

1. 若要一个运行时间为 $100n^2$ 的算法在相同机器上快于运行时间为 2n 的另一个算法，则最小的 n 值应为_____。

2. 某算法仅含顺序执行的语句 1 和语句 2，语句 1 执行次数为 $20n^2+3n\log n$，而语句 2 执行次数为 $2n^3$，则该算法的时间复杂度为 _____。

3. 下面术语中， _____与数据结构的存储结构无关。

   A. 顺序表　　B. 链表　　C. 队列　　D. 循环链表　　E. 堆

4. 在一个具有 n 个结点的有序单链表中插入一个新结点并仍保持其有序的时间复杂度为 _____。

   A. $O(n\log_2 n)$　B. O(1)　C. O(n)　D. $O(n^2)$

5. 元素 a、b、c、d、e、f 依次进栈，若允许进栈、退栈操作交替进行，但不允许连续三次进行退栈操作，则不可能得到的出栈序列是 _____。

   A. dcebfa　　B. cbdaef　　C. bcaefd　　D. afedcb

6. 表达式 3* 2^(4+2*2-6*3)-5 求值过程中，当扫描到 6 时，操作数栈和运算符栈为 _____ ， 其中 ^ 为乘幂。

   A. 3,2,4,1,1；(*^(+*-　　B. 3,2,8；(*^-

   C. 3,2,4,2,2；(*^(-　　D. 3,2,8；(*^(-

7. 有 4 棵结点存储整数关键码的二叉树，它们的中序遍历序列分别为：

   A. 5, 3, 1, 2, 4　　B. 1, 2, 3, 4, 5　　C. 5, 4, 3, 2, 1　　D. 1, 3, 5, 4, 2。

   请问其中可能是二叉搜索树的包括_____。

8. 一棵 Huffman 树的带权外部路径长度定义为所有叶结点权值与其深度的乘积之和。设根结点的深度为 0，对集合 { 2, 3, 5, 7, 8 } 构造二叉 Huffman 树，则其最小带权外部路径长度为 _____。

9. 两个字符串相等的充分必要条件是 _____。

10. 2-3 树是一种满足以下两个条件的特殊的树：（1）每个内部结点有两个或三个子结点；（2）所有叶结点到根的路径长度相同。如果一棵 2-3 树有 9 个叶结点，那么它可能有_____个非叶结点。

二、 辨析与简答(每题 6 分，共 30 分)

1. 什么是循环队列？循环队列的优点是什么？如何判别它的空和满？

2. 在包含 n 个结点的单链表、双链表和单循环链表中，若仅知道指针 p 指向某结点，不知道表的头指针，能否将结点 p 从相应的链表中删去？若可以，其时间复杂度各为多少?

3. 请分别计算模式 p = “aabaac” 优化前和优化后的 next 数组，并按照优化后的 next 数组针对目标 t = "aabaabaabaac" 进行 KMP 快速模式匹配，请画出匹配过程的示意图并计算匹配过程中的比较次数。

4. 根据树的双标记层次遍历序列，求其带度数的后根遍历序列 。譬如，已知一棵树的双标记层次遍历序列如下：

   A : ltag: 0 , rtag 1
   B : ltag: 0 , rtag 0
   C : ltag: 0 , rtag 1
   D : ltag: 1 , rtag 0
   G : ltag: 0 , rtag 1
   E : ltag: 0 , rtag 1
   H : ltag: 1 , rtag 1
   F : ltag: 1 , rtag 0
   I : ltag: 1 , rtag 1

   则其带度数的后根遍历序列为：D0 H0 G1 B2 F0 I0 E2 C1 A2 （注：各个结点按照“结点名 度数”的方式给出，结点之间用空格分隔）。现给出树的双标记层次遍历序列如下：

   A : ltag: 0 , rtag 1
   B : ltag: 0 , rtag 0
   G : ltag: 1 , rtag 1
   C : ltag: 0 , rtag 0
   F : ltag: 0 , rtag 1
   D : ltag: 1 , rtag 0
   E : ltag: 1 , rtag 1
   H : ltag: 1 , rtag 0
   I : ltag: 1 , rtag 1

   请给出其带度数的后根遍历序列。

5. 使用重量权衡合并规则（权重合并规则）与路径压缩，对下列从 0 到 15 之间的数的等价对进行归并。在初始情况下，集合中的每个元素分别在独立的等价类中。当两棵树规模同样大时，使结点数值较大的根结点作为值较小的根结点的子结点。

   (0,2) (1,2) (3,4) (3,1) (3,5) (9,11) (12,14) (3,9) (4,14) (6,7) (8,10) (8,7) (7,0) (10,15) (10,13)

   请填写下面表格的空白部分树的父指针表示法的数组表示。也就是所有等价对都被处理之后，所得父结点的下标值（没有父结点则填“–1”）。

父结点下标																
结点值	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
结点的下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

三、 算法填空(共 22 分)

1. （每空 3 分，共 12 分）下面的代码利用一个栈对一个给定的序列进行从小到大排序。长度为 n 的序列由 1, 2, …, n 这 n 个连续的正整数组成。请补全下面的代码段，使其可以判断给定的序列是否可以利用一个栈进行排序，如果可以，输出相应的操作。

例如：序列 4 3 1 2，此序列可以排序，输出：push push push pop push pop pop pop。

```
template <class T>                                  // 栈的元素类型为 T
class Stack{
    public:
        void clear();                               // 变为空栈
        void push(const T item);                    //  item 入栈
        T pop();                                    // 返回栈顶内容并弹出
        T top();                                    // 返回栈顶内容但不弹出
        bool isEmpty();                             // 若栈已空返回真
        bool isFull();                              // 若栈已满返回真
};
Stack<int> s;                                       // s 为栈

int sequence[maxLen], len;          // sequence 为待排序序列， len 为序列长度

bool islegal()  {                                   // 判断序列是否可以排序
    int i, j, k;
    for (i = 0; i < len; i++) {
        for (j = i+1; j < len; j++) {
            for (k = j+1; k < len; k++) {
                if (_____________)
                    return false;
            }
        }
    }
    return true;
}
void transform() {                                  // 输出转换操作
    int cur = 1, i = 0;
    while (_____________) {
        while (s.isEmpty() || _____________) {
            _____________
            cout << "push ";
        }
        s.pop();
        cout << "pop ";
        ++i;
    }
}
```

2. （每空 3 分，分析 1 分，共 10 分）以下算法用来判断两棵树是否相同，试补全下面的代码段，并分析算法的运行时间代价。

```
template<class T>
bool IsEqual(TreeNode<T> *root1, TeeNode<T> *root2) {
    while (root1 != NULL && root2 != NULL) {
        if (root1->value() != root2->value())
            return    ______;
        if (_____________________)
            return false;
        root1 = root1->RightSibling();
        root2 = root2->RightSibling();
    }
    if   (_____________________)
        return true;
    return false;
}
```

四、 算法设计与实现（12 分）

给定一个正整数序列 S 和一个整数 p，请设计一个算法判断是否存在一个连续子序列 S’，使得 S’的所有元素之和恰好为 p。尽量优化你的算法并分析其时间复杂度。

五、 分析证明题（共 16 分）

1. （4 分） 请证明非空满 K 叉树的叶结点数目为(K-1)*n+1,其中 n 为分支结点数目。

2. （12 分）有 n 个数 $K_1, K_2, \ldots, K_n$ 顺序存储在数组中，形成一棵 n 个结点的完全二叉树。现在要将它按如下方式建成一个最小值堆：从左至右扫描整个数组，对第 i 个元素 K_i (i = 1, 2, …, n)，将它与其父结点 $K_{[i/2]}$比较，如果 $K_i >= K_{[i/2]}$，则不再进行操作；若 $K_i < K_{[i/2]}$，将 2 个结点对换，再将 $K_{[i/2]}$与 $K_{[i/4]}$比较，以此类推，直到比较到根结点 K_1。

   (1) 按照这种建堆方式，最坏情况下建堆的时间复杂度是多少？

   (2) 教材上使用的“筛选法”建堆(Siftdown)，最坏情况下建堆的时间复杂度是多少？

   (3) 上述 2 种建堆法有类似之处，然而是什么导致它们的时间复杂度不一样？