一、单项选择题：

1.B 2.D 3.A 4.C 5.D 6.B 7.B 8.C 9.C 10.D 11.C 12.B 13.C 14.D 15.C

二、填充题：

1. 23 12 3 \* 2 - 4 / 34 5 \* 7 / + + 108 9 / +

2. 69

3. 2, M/2

4. k t e r s p

5. 5

6. 0, 2

7. 前序：A, B, F, J；中序：E, D, H；后序：C, K, G

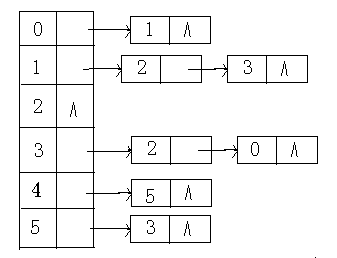
8. (49, 13, 27, 50, 76, 38, 65, 97)

9. 1, 3, 2, 4, 5

10. (n-1)n/2

三、简答题：

1. 答：

（2分）

共有4个强连通分量（3分）。

2. 答：（结论得2分）树的高度一定增加。（说明得3分）因为“查找路径上的任一结点的平衡因子皆为零”，从根结点开始查找，根结点的平衡因子为零，说明根的左右子树高度相等。沿左（或右）子树向下查找时，查找路径上所有结点的平衡因子皆为零，说明任一结点的左右子树高度向度，查找失败是在叶子结点，插入也是在叶子结点，树的高度自然增加。

四、分析题：

1. 答：

1. （2分）生成优先级队列的for循环将所有的边入队。需要的时间是O（|E|log|E|）
2. （2分）在最坏的情况下，归并的循环可能需要检查所有的边。对于每条边，最多需要执行两次Find操作和一次Union操作。因此，归并循环的最坏情况的时间复杂度是O（|E|log|V|）。
3. （2分）在一个连通图中，一般边数总比结点数大，所以，Kruskal算法的时间复杂度是O（|E|log|E|）

2. 答：（4分）将无向图的邻接矩阵转为对应邻接表的算法。

五、设计题：

1. 设计一个求结点x在二叉树中的双亲结点算法。

判断根的关键字等于x的情况，得1分（一定要考虑）；

判断找不到关键字等于x的情况，得1分（一定要考虑）；

其他根据算法正确性可得3分。此题满分5分。

2. 试设计一个算法（伪代码），求以二叉链表表示的二叉树中所有叶子结点数。

判断根是叶子的情况，得1分（一定要考虑）；

其他根据算法正确性可得4分。此题满分5分。

六、程序填充题：

1.

if (array[pos].state == 0) return false;

if (array[pos].state == 1 && array[pos].data== x)

} while (pos != initPos);

2．

ptr->data = \*ppos ;

for(pos=ipos; pos < ipos+n ; pos++) if(\*pos==\*ppos) break;

k = pos-ipos ;

ptr->right = buildBinTree( ppos+1+k, pos+1,n-1-k);

S.push(new TOHobj(num-1, tmp, goal, start));

S.push(new TOHobj(start, goal));

S.push(new TOHobj(num-1, start, tmp, goal));

七、附加题：

(1).

template<class T>

T GetKth(T a[], int N, int K)

{

priorityQueue<T, greater<T>> hp(a, N);

T ret;

for(int I = 0; i < K; i++)

ret = hp.deQueue();

return ret;

}

评分标准：

1. 在算法中体现出最大化堆思想，得1分
2. 算法正确的，得1分
3. 程序正确的，得3分，程序不再细分得分情况。

(2).

template<class T>

T GetKth(T a[], int N, int K)

{

priorityQueue<T> H(a, K);

for (int i = K; i < N; i++)

{

if (H.getHead() < a[i])

{

H.deQueue();

H.enQueue(a[i]);

}

}

return H.getHead();

}

评分标准：

1. 在算法中体现出最小化堆思想，得1分
2. 算法正确的，得1分
3. 程序正确的，得3分，程序不再细分得分情况。