数据结构参考答案

author：熊、琨、曾、颐

仅供参考，敬请指正

选择题

1. B：画个表操作一下即可
2. B：链表用地址指针指向下一个确定，地址可以不连续。顺序表地址要连续。
3. B：如果空栈无法出栈、无法求栈顶、个数为0
4. A：队列先进先出，什么顺序入就什么顺序出
5. B：显然
6. D：后进先出，1不可能比2先出
7. A：见4
8. B：画个树即可
9. B：有头节点的单链表，头结点下一个是第一个节点。
10. C：线性结构头结点前驱为0，后续节点前驱为1。树形结构根节点前驱为0，叶子节点前驱为1。
11. D：AC选项相同，无语了。把q下一个给p的下一个，q的下一个是p。
12. C：p是q下一个，就是要删掉的那个。q的下一个是p的下一个。
13. C：总共匹配10次

    ```
    BDBABDABDAB
    BDA
     B
      BD
       B
        BDA
    ```

14. A：上课讲的
15. A：先序遍历：根左右，中序遍历：左根右，后序遍历：左右根。

填空题

1. 数组下标、指针地址
2. 首地址为144，地址区间为144-147

   第一个元素：100-103

   第二个元素：104-107

   ...
3. 空栈
4. 删除后续节点的意思如果是删掉后面所有节点的话，答案就是`p->link = NULL`

   如果就是删掉一个节点的话，答案是`p->link = p->link->link`
5. `ab+c*d+` 遇到数字就输出，遇到符号判断栈顶符号优先级出入栈

6. `n+1`，前驱，后继

   线索化二叉树的指针存储

7. 碰撞，同义词

   key不同但hash相同，称为碰撞。

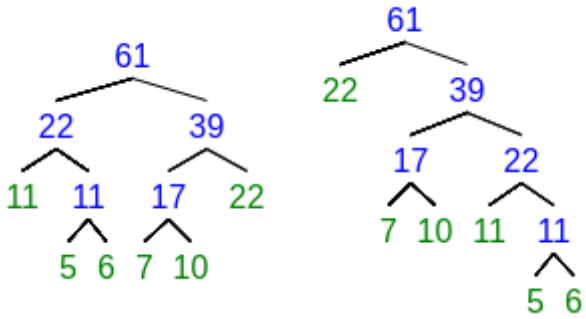
8. `v1->v2->v3->v6->v5->v4` ； `v1->v2->v5->v4->v3->v6`

   根据邻接表，把有向图画出来。v1指向v2、v5、v4；v2指向v3、v5...

   答案不唯一

应用题

1.
   1. `[61[22[11][11[5][6]]][39[17[7][10]][22]]]`

      所有绿色字外面是方框，蓝色字外面是圆框。

      每次取两个最小的数，较小的数在左侧。

   2. `长度 = 2 * 11 + 3 * 5 + 3 * 6 + 3 * 7 + 3 * 10 + 2 * 22 = 150`

      距离 * 节点 求和

2.

key	h(key)
7	0
8	3
30	6
11	5
18	5
9	6
14	0

   冲突线性检测法：如果坑里有数了，往后找，直到有一个没数的。

h(key)	0	1	2	3	4	5	6
key	7	18	9	8	14	11	30

   找7、8、30、11: 1次

   找18: 4次

找9: 4次

找14: 5次

`(1 * 4 + 4 + 4 + 5) / 7`

3.

1. `ABCDEFGHK`
2.

i	info	parent
0	A	-1
1	B	0
2	C	1
3	D	1
4	E	3
5	F	0
6	G	0
7	H	6
8	K	7

阅读填空

1. `pdic->element[mid].key == key`
2. `mid - 1`
3. `mid + 1`

编程题

1.
```
int deleteNode(LinkList llist, Node *p) {
    Node *t;
    for (t = llist->link; t->link != NULL; t = t->link) {
        if (t->link == p) {
            t->link = p->link;
            return 1;
        }
    }
    return 0;
}
```

2.
```
BinTreeNode searchNode(BinTreeNode t, DataType x) {
    BinTreeNode q;
    if (t == NULL) return NULL;
    if (t->info == x) return t;
    q = searchNode(t->llink, x);
    if (q != NULL) return q;
    q = searchNode(t->rlink, x);
    if (q != NULL) return q;
    return NULL;
}
```

3.
```
int Fib(int n) {
    int result=0, x;
    PLinkStack s = createEmptyStack();
    push_stack(s, n);
    while(!isEmptyStack(s)) {
        x = top_stack(s);
        pop_stack(s);
        if (x == 1 || x == 0) result += x;
        else {
            push_stack(s, x - 1);
            push_stack(s, x - 2);
        }
    }
    return result;
}
```