

```

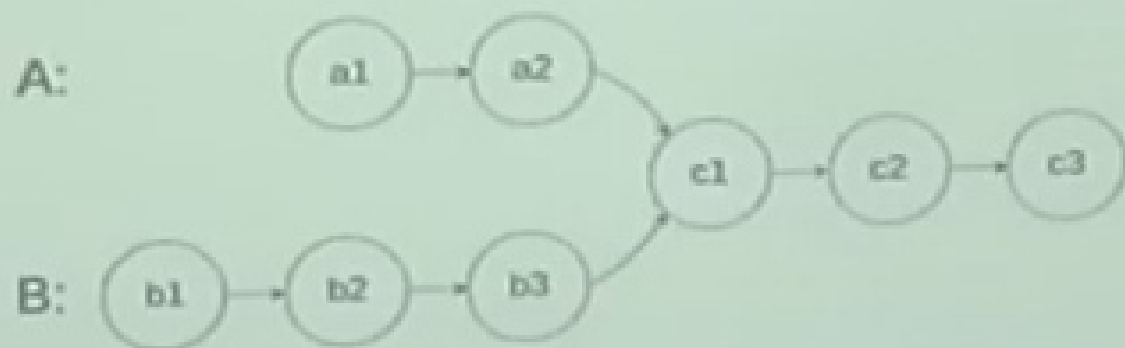
while (index[stk[dep_pos]] != 0)
    ( _____ // 填空 4 )
    dep_pos++;
}
for (int j=1; j<=max; j++)
    for (int k=1; k<=mc; k++) {
        stk[jk] = q[k];
        th++;
    }
    top = th-1;
}
else {
    top++;
    stk[top] = s[i];
}
for (int i=1; i<=top; i++)
    printf("%c", stk[i]);
printf("\n");
return 0;

```

#### 四、 算法设计与实现

1. 假设有两个单链表 A 和 B 在某个结点相交, 如下图所示 (且整个链式结构中不存在环), 单链表 A 和 B 的头结点地址分别为 `headA` 和 `headB`, 请你设计算法, 找出并返回这两个单链表相交的起始结点。请给出算法描述及其伪代码。(算法应尽量满足  $O(\text{len}(A) + \text{len}(B))$  的时间复杂度和  $O(1)$  的空间复杂度, 若不满足复杂度要求, 会酌情扣分。)

参考答案:



### 三、 算法填空

1. 对于一个全由小写字母组成的字符串，我们可以考虑采用以下符号来压缩其长度：用  $x(s)$  表示将字符串  $s$  重复  $x$  次，其中  $s$  为一个任意长度的非空字符串， $x$  为一正整数 ( $0 < x < 300$ )。现给一经过压缩的字符串，请完善下面的程序以使其可解码出原来的字符串。

注意：保证输入合法。保证最后解码得到的字符串长度  $\leq 100000$

提示：程序通过栈这个数据结构来生成答案

✓ 样例输入: 3(a2(c))

✓ 样例输出: aacacacacac

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
char s[N], stk[N], t[N];
// stk 即为模拟栈的数组
// stk[0] 表示栈底元素, stk[top] 表示栈顶元素, top=0 表示栈为空
```

## 二、辨析与简答

1. 对于  $t = \text{"aabaabaabaac"}$  的待匹配文本及  $p = \text{"aabaac"}$  的模式，请求出  $p$  的优化后的  $\text{next}$  数组并进行 KMP 快速模式匹配，画出匹配过程的示意图。

参考答案：

下标	0	1	2	3	4	5
P	a	a	b	a	a	c
N	-1	-1	1	-1	-1	2

O	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	a	b	a	a	b	a	a	b	a	a	c				
a	a	b	a	a	c										
					$\times$	$i = 5, j = 5, i \neq n[j] = 2$									
			a	a	b	a	a	c							
								$\times$	$i = 5, j = 8, i \neq n[j] = 2$						
						a	a	b	a	a	c	匹配			

2. 二叉搜索树的数据结构定义如下。请补全以下代码，实现利用非递归方法找到二叉搜索树中第  $k$  大的数的功能。

```
class Node {  
public:  
    int val;  
    Node* left;  
    Node* right;  
    Node* parent;  
    bool visited;  
  
    Node(int v) {  
        val = v;  
        parent = left = right = NULL;  
        visited = false;  
    }  
    Node(int v, Node* l, Node* r, Node *p) {  
        val = v;  
        left = l;  
        right = r;  
        parent = p;  
        visited = false;  
    }  
};
```

5. 使用重量权衡合并规则与路径压缩优化, 对下列 15 个等价对进行合并。初始情况下, 集合中的每个元素分别在独立的等价类中。使用重量权衡合并规则, 合并时子树结点少的并入结点数多的那棵 (多的那个作为新树根, 少的那个根作为新根的直接子结点); 若两棵树规模同样大, 则把根值较大的并入根值较小 (新树根取值小的)。

(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)

请填写下面表格中的空白部分树的父指针表示法的数组表示, 也即所有等价对都被处理之后, 所得父结点的下标值。

父节点的下标																
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

4. 双端队列是指插入和删除操作限制在两端进行的线性表。若将  $n$  个互不相同的元素依次插入到双端队列中（仅插入不删除），则可得到 \_\_\_\_\_ 种不同的排列？

A. 2      B.  $2n$       C.  $2^{n-1}$       D.  $2^n$

答案：C

解析：第一个元素从左/右入队没有区别，之后每个元素都有左、右两种入队方式

2. 在\_\_\_\_\_中, 即使丢失了头结点, 只要指出表中任何一个结点的指针, 也可以访问到该结点的前驱结点。

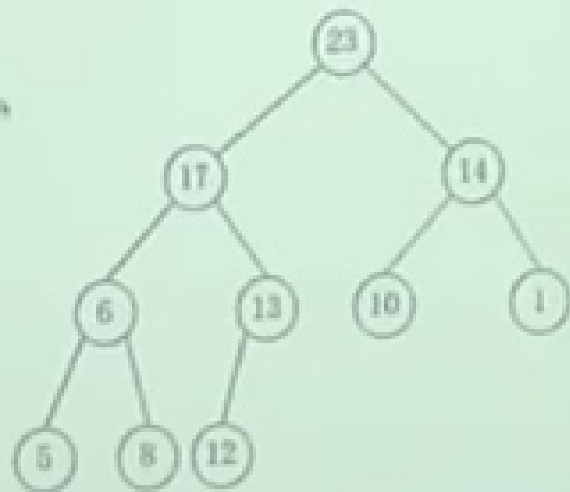
- A. 线性单链表      B. 双向链表      C. 线性链表      D. 循环链表

答案: BD



2. 序列 23, 17, 14, 6, 13, 10, 1, 5, 8, 12 是否为一个最大值堆？若是，请说明理由，否则请严格按照筛选法建堆的过程将其调整成为最大值堆，并画出调整建堆的逐步过程。

参考答案：否，结点 6 比右儿子 8 要小，所在子树不满足最大值堆的性质。



2. 对于一个长度为  $n$  的字符串  $S$ ，我们称  $S[0..i-1]$  为字符串  $S$  长度为  $i (1 \leq i \leq n)$  的前缀。举例来说，字符串 abcd 的所有前缀是：a, ab, abc, abcd。现给出一个字符串  $S$ ，请你设计出一个复杂度尽量低的算法来计算出  $S$  的所有前缀中，前缀长度与前缀出现次数的乘积的最大值。要求写出算法的思想，并分析时空复杂度。若  $S = \text{"cdcdcdc"}$ ，则有前缀：

"c"，长度为 1，出现 4 次，乘积  $1 \cdot 4 = 4$ ，

"cd"，长度为 2，出现 3 次，乘积  $2 \cdot 3 = 6$ ，

"cdc"，长度为 3，出现 3 次，乘积  $3 \cdot 3 = 9$ ，

"cdcd"，长度为 4，出现 2 次，乘积  $4 \cdot 2 = 8$ ，

"cdcdc"，长度为 5，出现 2 次，乘积  $5 \cdot 2 = 10$ ，

"cdcdcd"，长度为 6，出现 1 次，乘积  $6 \cdot 1 = 6$ ，

"cdcdcdc"，长度为 7，出现 1 次，乘积  $7 \cdot 1 = 7$ 。

其中前缀 "cdcdc" 长度为 5，出现 2 次，乘积为 10，大于其他任何一个前缀对应的乘积。故对于此字符串  $S$ ，答案为 10。

参考答案:

```
return c>='0' && c<='9';
```

```
stk[j+1] = '\0'
```

```
t[j-1] = stk[j];
```

```
num = num*10 + stk[digit_pos] - '0' ;
```

```

class BinaryTree {
public :
    Node* root;
    stack<Node *> st;
    BinaryTree(Node *r) {
        root = r;
    }
    BinaryTree() {}
    ~BinaryTree() {
        //...
    }
    //...
    Node * kth_number_non_recursive(int k, Node *root) {
        while (root != NULL) {
            while ( _____ // 填空 5 ) {
                _____ // 填空 6
            }
        }
    }
}

```

7. 将森林 F 转换为对应的二叉树 T, F 中的叶结点的个数为\_\_\_\_\_。

A. T 中叶结点的个数

B. T 中度为 1 的结点个数

C. T 中左子指针为空的结点个数

D. T 中右子指针为空的结点个数

答案: C.



3. 采用教材上将中缀表达式转换为后缀表达式的过程中，栈中不可能出现的状态是\_\_\_\_\_？（左侧为栈底）

- A.  $+/+$     B.  $+(+$     C.  $(+)$     D.  $+*(+ +$

答案：ACD

解析：栈中不可能出现相互匹配的一对括号；优先级较低的运算符不可能紧接着优先级较高的运算符入栈。



参考答案:


root->right != NULL && !root->right->visited (条件的前后顺序不能换)

root = root->right;

k -= 1 (相同意思的语句均可)

root->left != NULL && !root->left->visited (条件的前后顺序不能换)

root = root->left;



8. 设一棵完全二叉树具有 1000 个结点，则此完全二叉树有 \_\_\_\_\_ 个叶子结点，有 \_\_\_\_\_ 个度为 2 的结点，有 \_\_\_\_\_ 个结点只有非空左子树，有 \_\_\_\_\_ 个结点只有非空右子树。

11



5. 设顺序存储的循环队列  $Q(1:40)$ ，数组下标为 1 到 40，其中  $front$  指向队列头元素的位置， $rear$  指向队列尾元素的下一位置，其初始状态为  $front=rear=40$ ，经过一系列入队与退队运算后， $front=20$ ， $rear=15$ ，现要在该循环队列中寻找最小值的元素，最坏情况下需要比较的次数为 \_\_\_\_\_。

答案：34。

解析：Front 在 rear 之后，说明队列中间换向了，有效数据个数是  $15 + (40 - 20) = 35$  个，寻找最小值需要遍历所有元素，即  $35 - 1 = 34$  次比较。

6. 若一棵二叉树的后序遍历序列是{ 1, 3, 2, 6, 5, 7, 4 }, 中序遍历序列是{ 1, 2, 3, 4, 5, 6, 7 }, 则下列哪句是错的 \_\_\_\_\_?

A. 2 是 1 和 3 的父结点

B. 7 是 5 的父结点

C. 这是一棵二叉搜索树

D. 这是一棵完全二叉树

## 一、 选择填空

1. 以下算法的时间复杂度是 \_\_\_\_\_.

```
int m=1, i, j;  
for (i = 1; i <= n; i++)  
    for (j = 1; j <= i; j *= 2)  
        m *= 2;
```

A.  $\Theta(\log_2 n)$

B.  $\Theta(n)$

C.  $\Theta(n \log_2 n)$

D.  $\Theta(n^2)$

答案: C

11

```
if (!root->visited) {  
    _____ // 填空 7  
    if (k == 0) break;  
    root->visited = true;  
}  
if (_____ // 填空 8)  
    _____ // 填空 9  
}  
else {  
    root = root->parent;  
}  
}  
return root;  
}  
};
```