

## 目录

1 简答题	1
2 下面 C 语言的输出是什么, 并给出解释	1
3 给出下面 C 语言程序的输出, 并解释为什么	1
4 下面 C 语言会输出什么, 并给出解释	2
5 二叉树如下, 使用先序遍历的结果是:	3
6 二叉搜索树如下, 请问以何种顺序输入无法构造这样的二叉树	3
7 使用直线划分空间	4
8 使用折线划分空间	5
9 打印三角形	6
10 实现 atof 函数	6
11 使用栈的数据结构实现队列的功能	7

## 1 简答题

一共 1000 个苹果, 有任意多个箱子用来装苹果, 要求一个或多个箱子中的苹果数量之和可以得到 1 到 1000 中的任意数目的苹果。

请问最少需要多少个箱子才能满足上述条件?

## 2 下面 C 语言的输出是什么, 并给出解释

```
char p[20];  
char *s = "string";  
int length = strlen(s);
```

```
int i;
for (i = 0; i < length; i++)
    p[i] = s[length - i];
printf("%s",p);
```

- a) gnirts
- b) gnirt
- c) string
- d) 没有输出

### 3 给出下面 C 语言程序的输出, 并解释为什么

```
#include <stdio.h>

int main() {
    if (sizeof(int) > -1)
        printf("True");
    else
        printf("False");
}
```

### 4 下面 C 语言会输出什么, 并给出解释

```
#include <stdio.h>

main()
{
    int n = 0, m = 0;
    if (n > 0)
        printf("True");
    if (m > 0)
```

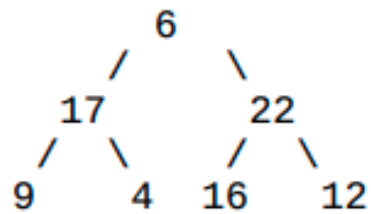
```

    else
printf("False");
}

```

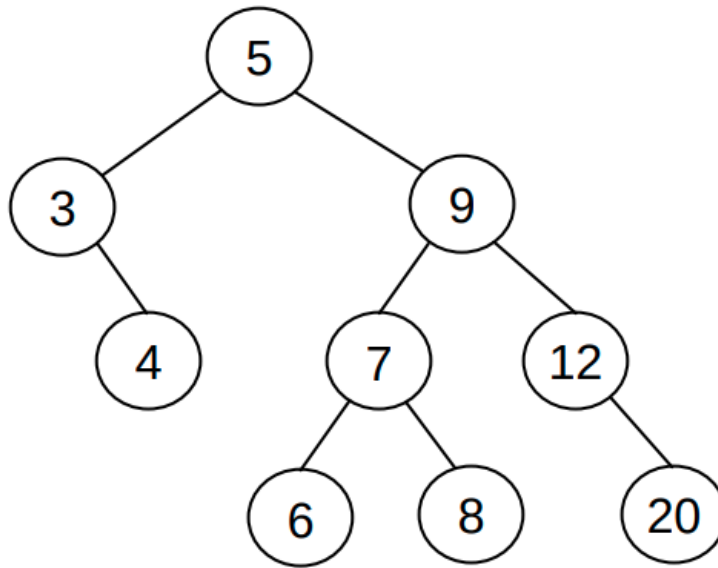
- a) True
- b) False
- c) 没有输出
- d) 运行错误

5 二叉树如下, 使用先序遍历的结果是:



- A. 9 4 17 16 12 11 6
- B. 9 17 6 4 16 22 12
- C. 6 9 17 4 16 22 12
- D. 6 17 22 9 4 16 12
- E. 6 17 9 4 22 16 12

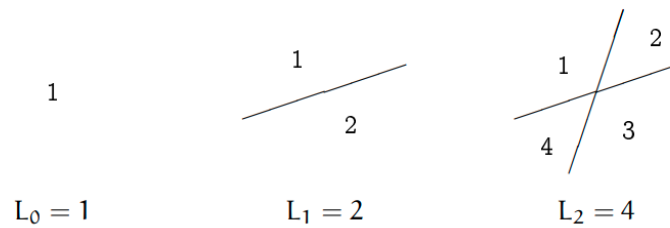
- 6 二叉搜索树如下，请问以何种顺序输入无法构造这样的二叉树



- A. 5 3 4 9 12 7 8 6 20  
B. 5 9 3 7 6 8 4 12 20  
C. 5 9 7 8 6 12 20 3 4  
D. 5 9 7 3 8 12 6 4 20  
E. 5 9 3 6 7 8 4 12 20

## 7 使用直线划分空间

如下图所示:



- 0 根直线可以划分出 1 个空间
- 1 根直线可以划分出 2 个空间
- 2 根直线可以划分出 4 个空间

**问题:**

1. 写出公式  $L(n)$ ;  $n$  表示折线数量,  $L(n)$  表示通过  $n$  根折线可以划分出的最多的空间数量
2. 使用 C 语言实现计算  $L(n)$  的函数

```
int calc_line_spaces(int n); // n >= 0
```

## 8 使用折线划分空间

如下图所示:

- 0 根折线可以划分出 1 个空间
- 1 根折线线可以划分出 2 个空间
- 2 根折线最多可以划分出 7 个空间



问题:

1. 写出公式  $Z(n)$ ;  $n$  表示折线数量,  $Z(n)$  表示通过  $n$  根折线可以划分出的最多的空间数量
2. 使用 C 语言实现计算  $Z(n)$  的函数

```
int calc_zig_spaces(int n); // n >= 0
```

## 9 打印三角形

```

      1
    2  2
  3  4  3
4  7  7  4
5 11 14 11 5
.  .  .  .  .

```

观察上图三角形的规律, 实现函数根据输入  $n$  打印  $n$  行如图所示三角形.

```
void draw(unsigned int n); //  $n > 0$ 
```

## 10 实现 atof 函数

- 函数定义

```
double my_atof(char *nptr);
```

- 函数描述

`my_atof()` 会扫描参数 `nptr` 字符串, 跳过前面的空格字符, 直到遇上数字或 `.` 符号才开始做转换, 而遇到非数字或字符串结束时 (`'\0'`) 才结束转换, 并将结果返回。

以下都是合法输入:

```
0.123
.123
16.4
16.
0.0
0.
```

注意: 不考虑 `+-` 符号, 不考虑输入非法的情况

## 11 使用栈的数据结构实现队列的功能

1. 你有完整的栈的数据结构可以使用:

```
stack.c
```

```
stack.h
```

2. 只能使用上面文件中提供的方法来实现队列的 enqueue 和 dequeue 方法, 函数声明类似如下:

```
enqueue(Queue* queue, int data); // 函数定义请自己考虑  
int dequeue(Queue* queue); // 函数定义请自己考虑
```

- Queue 结构体的定义在 stack.h 文件中
- 上面的两个函数里面只能调用已有的函数, 不能使用其他方法对入参 queue 进行操作
- 测试用例类似如下:

```
int main(void) {  
    Queue* queue = init_stack();  
    int a[5] = {1, 2, 3, 4, 5};  
    for( int i = 0; i < 5; i++) {  
        enqueue(queue, a[i]); // 可按照自己的函数定义进行修改  
    }  
  
    for (int i = 0; i < 5; i++) {  
        int out = dequeue(queue); // 可按照自己的函数定义进行修改  
        printf("%3d", out);  
    }  
    printf("\n");  
    return 0;  
}
```

程序应当输出类似结果: 1 2 3 4 5