实验练习04 参考答案

1. 十进制转二进制

```
class Stack:
    def __init__(self):
        self.__items = []
    def isempty(self):  # 返回堆栈是否为空
        if len(self.__items) == 0:
            return True
        else:
            return False
    def length(self):
        return len(self.__items)
    def push(self,element):  # 压入堆栈
        self.__items.append(element)
    def pop(self):  # 弹出堆栈，注意需要处理堆栈为空的情况
        try:
            return self.__items.pop()
        except:
            print('ERROR: Stack is empty now!')


def dec2bin(num):
    s = Stack()
    while num != 0:
        s.push(num % 2)
        num = num // 2
    while s.length() != 8:
        s.push(0)
    result = ''
    while s.isempty() == False:
        result += str(s.pop())
    return result


ip_dec = '203.179.25.37'
ip_dec_list = ip_dec.split('.')
result = ''
for ip in ip_dec_list:
    result += dec2bin(int(ip))
print(result)
```

```
11001011101100110001100100100101
```

---

2. 队列

```
class Queue:
    def __init__(self):
        self.__items = []
    def length(self):
```

```python
        return len(self.__items)
    def isempty(self):
        if len(self.__items) == 0:
            return True
        else:
            return False
    def enqueue(self, element):
        self.__items.append(element)
    def dequeue(self):
        try:
            return self.__items.pop(0)
        except:
            print('ERROR:Queue is empty now!')
    def get_head(self):
        try:
            return self.__items[0]
        except:
            print('ERROR:Queue is empty now!')
    def get_tail(self):
        try:
            return self.__items[-1]
        except:
            print('ERROR:Queue is empty now!')


q = Queue()
q.enqueue(7)
q.enqueue(5)
q.enqueue(8)
print(q.length())
print(q.isempty())
print(q.get_head())
print(q.get_tail())
q.dequeue()
q.dequeue()
q.dequeue()
q.dequeue()
print(q.isempty())
```

```
3
False
7
8
ERROR:Queue is empty now!
True
```

3. 树的层序遍历

```python
class Queue:
    def __init__(self):
        self.__items = []
    def length(self):
        return len(self.__items)
    def isempty(self):
        if len(self.__items) == 0:
            return True
        else:
            return False
    def enqueue(self, element):
        self.__items.append(element)
    def dequeue(self):
        try:
            return self.__items.pop(0)
        except:
            print('ERROR:Queue is empty now!')
    def get_head(self):
        try:
            return self.__items[0]
        except:
            print('ERROR:Queue is empty now!')
    def get_tail(self):
        try:
            return self.__items[-1]
        except:
            print('ERROR:Queue is empty now!')


class BinaryTree:
    Q = Queue()  # 创建一个类属性，该队列为BinaryTree类派生的所有对象共有
    def __init__(self,data=None,left=None,right=None):  # 如果创建节点对象时left或right参数为空，则默认该节点没有左或右子树
        self.data = data
        self.left = left
        self.right = right
    def level_order(self):
        BinaryTree.Q.enqueue(self)  # 先将根节点加入队列
        while BinaryTree.Q.isempty() == False:
            node = BinaryTree.Q.dequeue()
            print(node.data,end=' ')
            if node.left != None:
                BinaryTree.Q.enqueue(node.left)
            if node.right != None:
                BinaryTree.Q.enqueue(node.right)


layer3_2 = BinaryTree(2,BinaryTree(7),BinaryTree(4))
layer2_5 = BinaryTree(5,BinaryTree(6),layer3_2)
layer2_1 = BinaryTree(1,BinaryTree(0),BinaryTree(8))
layer1_3 = BinaryTree(3,layer2_5,layer2_1)

layer1_3.level_order()
```

```
3 5 1 6 2 0 8 7 4
```

4. 输出树的叶子节点

```
class BinaryTree:
    def __init__(self,data=None,left=None,right=None):  # 如果创建节点对象时left或right参数为空，则默认该节点没有左或右子树
        self.data = data
        self.left = left
        self.right = right
    def print_leaf(self):  # 由前序遍历改编
        if self.left == None and self.right == None:
            print(self.data,end=' ')
        if self.left != None:
            self.left.print_leaf()
        if self.right != None:
            self.right.print_leaf()


layer3_2 = BinaryTree(2,BinaryTree(7),BinaryTree(4))
layer2_5 = BinaryTree(5,BinaryTree(6),layer3_2)
layer2_1 = BinaryTree(1,BinaryTree(0),BinaryTree(8))
layer1_3 = BinaryTree(3,layer2_5,layer2_1)

layer1_3.print_leaf()
```

```
6 7 4 0 8
```

5. 手机号正则表达式

```
import re
p = '^1(3[0-35-9]|47|5[0-35-9]|66|7[2-35-8]|8[0-9]|9[1389])\d{8}$|^134[0-
8]\d{7}$'  # 答案不唯一
t1 = '13112345678'
t2 = '13498765432'
t3 = '12345678901'
t4 = '1370211234#'
print(re.match(p,t1))
print(re.match(p,t2))
print(re.match(p,t3))
print(re.match(p,t4))
```

```
<re.Match object; span=(0, 11), match='13112345678'>
None
None
None
```