武汉大学数学与统计学院2017-2018学年第一学期期末考试
数据结构与算法（A卷答案）

1. （20分） Python相关

   (a) （5分）

```
dict = {'Name':'Li Ming', 'Sex':'Male',
        'HomeTown':'Wuhan', 'Age':20}

set = {2, 'Python', True, 3.14}
```

   (b) （5分）

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
[0, 2, 4, 6, 8]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

   (c) （5分）

```
alist = [x * x for x in range(1, 11) if x % 2 == 0]

alist = [m + n for m in 'abc' for n in '1234']
```

   (d) （5分）

```
class Rectangle(object):
    def __init__(self, length, width):
        self.length = length
        self.width  = width
    def area(self):
        return self.length *  self.width
    def circum(self):
        return 2 * (self.length + self.width)
```

2. （15分）

   (a) （6分）

```
        return self.next
        self.data = newdata
        self.next = newnext
```

   (b) （9分）

```
        Node(item)
        self.head
        self.head = temp

        current != None and not found:
        current.getData() == item:
        current = current.getNext()


        previous = current
        current = current.getNext()
        self.head = current.getNext()
```

3. （15分）

(a) （6分）

```
        self.items.append(item)
        return self.items.pop()
        return self.items[len(self.items)-1]
```

(b) （9分）

```
        stack.push(rem)
        not stack.isEmpty():
        binString = binString + str(stack.pop())
```

4. （20分） 队列与二叉树

(a) （4分）

```
      self.items.insert(0, item)
      return self.items.pop()
```

(b) （4分）

```
      self.lchild = BinaryTree(newNode)
      t = BinaryTree(newNode)
      t.lchild = self.lchild
      self.lchild = t
```

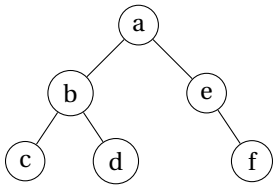
(c) （4分）

```
      tree
      inorder(tree.getLeftChild())
      print(tree.getRootVal())
      inorder(tree.getRightChild())
```

(d) （4分）

```
      q.enqueue(node)
      node = q.dequeue()
      q.enquque(node.lchild)
      q.enqueue(node.rchild)
```

(e) （4分）




```
tree = BinaryTree('a')
tree.insertLeft('b')
tree.insertRight('e')
tree.getLeftChild().insertLeft('c')
tree.getLeftChild().insertRight('d')
tree.getRightChild().insertRight('f')

inorder(tree)
levelorder(tree)
```

运行结果

```
cbdaef
abecdf
```

5. （15分）

(a) （5分）

```
def sequentialSearch(alist, item):
    pos = 0
    found = False
    while pos < len(alist) and not found:
        if alist[pos] == item:
            return True
        else:
            pos = pos + 1
    return found
```

(b) （5分）

```
def mergeSort(alist):
    if len(alist) > 1:
        mid = len(alist) // 2
        left = alist[:mid]
        right = alist[mid:]

        mergeSort(left)
        mergeSort(right)

        i = 0
        j = 0
        k = 0
        while i < len(left) and j < len(right):
            if left[i] < right[i]:
                alsit[k] = left[i]
                i += 1
            else:
                alist[k] = right[j]
                j += 1
        k += 1

        while i < len(left):
            alist[k] = left[i]
            i += 1
            k += 1

        while j < len(right):
            alist[k] = right[j]
            j += 1
            k += 1
```

(c) （5分）

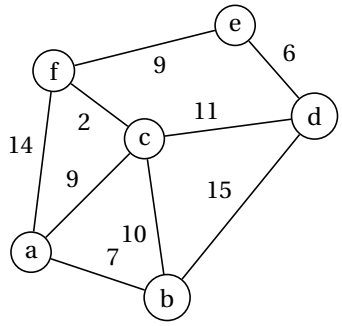
```
alist = [53, 24, 91, 88, 34, 71, 44, 18]

if sequentialSearch(alist, 88):
```

```
    print('Exist!')
else:
    print('Failed!')

print(alist)
mergeSort(alist)
print(alist)
```

6. （15分）



(a) （5分）

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 7 & 9 & 0 & 0 & 14 \\ 7 & 0 & 10 & 15 & 0 & 0 \\ 9 & 10 & 0 & 11 & 0 & 2 \\ 0 & 15 & 11 & 0 & 6 & 0 \\ 0 & 0 & 0 & 6 & 0 & 9 \\ 14 & 0 & 2 & 0 & 9 & 14 \end{pmatrix} \end{array}$$

(b) （5分）

```
{'a': 3, 'b': 3, 'c': 4, 'd': 3, 'e': 2, 'f': 3}
```

(c) （5分）

深度优先

```
a, b, c, d, e, f
```

广度优先

```
a, b, c, f, d, e
```