

# 复旦大学计算机科学技术学院

## 2021-2022 学年第二学期期末网上模拟试卷

课程名称：\_\_\_\_\_ 课程代码：\_\_\_\_\_

卷 别： ☒ A 卷    ☐ B 卷    ☐ C 卷

姓 名：\_\_\_\_\_ 学 号：\_\_\_\_\_

提示：请同学们秉持诚实守信宗旨，谨守考试纪律，摒弃考试作弊。学生如有违反学校考试纪律的行为，学校将按《复旦大学学生纪律处分条例》规定予以严肃处理。

题号	一			二		三	四			总分
	1	2	3	1	2	1	1	2	3	
得分										

### 一、阅读程序写输出（30%）

#### 第 1 题（10%）

```
ls = list(range(10))
```

```
for i in range(0, 6, 3):
```

```
    print(ls[j::i+1])
```

答案：

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[3, 7]

#### 第 2 题（10%）

```
for i in range(1,10):
```

```
    if i % 7 == 0:
```

```
        break
```

```
if i % 2:

    continue

print(i)
```

答案:

2  
4  
6

### 第 3 题 (10%)

```
B = [1, 2, 3]
try:
    a = b[3]
except IndexError:
    print('IndexError')
except:
    print('Error')
else:
    print('OK')
finally:
    print('Exit')
```

答案:

Error  
Exit

## 二、程序填空（20%）

### 第 1 题（10%）

请输入两个正整数 n 和 k（输入时 n 和 k 之间用英文逗号隔开）。要求生成 n 以内的所有是 k 的倍数的数, 并打印输出。如果输入错误, 则提示用户“输入错误, 请重试!”, 并要求用户重新输入。下面给出了运行的示例。

```
请输入n和k两个正整数,用英文逗号隔开:50,a
输入错误, 请重试!
请输入n和k两个正整数,用英文逗号隔开:50,-1
输入错误, 请重试!
请输入n和k两个正整数,用英文逗号隔开:50 6
输入错误, 请重试!
请输入n和k两个正整数,用英文逗号隔开:50,6
[6, 12, 18, 24, 30, 36, 42, 48]
```

```
# -*- coding:cp936 -*-
```

```
'''
```

```
    【程序填空】
```

```
-----
```

注意:

(1) 每行限填一行代码。

(2) 除要求填空的位置之外, 请勿改动程序中的其他内容。

```
-----'''
```

```
def gernerateList():
```

```
    n,k = 0,0
```

```
    finished = False
```

```
    while not finished:
```

```
        line = input('请输入 n 和 k 两个正整数,用英文逗号隔开:')
```

```
        nums = _____ 1 _____
```

```
        try:
```

```
            n = int(nums[0])
```

```
            k = int(nums[1])
```

```
            if n > 0 and k > 0:
```

```
                lst = _____ 2 _____ #要求使用列表推导式
```

```
                print(lst)
```

```
                finished = True
```

```
            else:
```

```
                print('输入错误, 请重试!')
```

```
        except:
```

```
            pass
```

```
            print('输入错误, 请重试!')
```

```
if __name__ == '__main__':
```

```
    gernerateList()
```

答案：

序号	填空
1	line.split(',')
2	[i for i in range(1,n+1) if i%k==0]

## 第 2 题 (10%)

一个 python 模块会提供多个函数供其他模块调用，我们希望能够找到那些在行首以 `def` 关键字开始所定义的函数，比如下面所定义的 `f1`、`f2` 和 `f3`，而 `t1` 并不包含在内。下面的程序找到这些函数，并且输出。对于下面的 `text` 而言，最后的输出如下：

1 f1

2 f2

3 f3

下列程序中有两个空格，请根据上下文以及代码中的注释部分，填充合适的代码。

```
'''-----
```

【程序填空】

```
-----
```

注意：

(1) 每行限填一行代码。

(2) 除要求填空的位置之外，请勿改动程序中的其他内容。

```
-----'''
```

```
text = """
```

```
# def f():
```

```
def f1(): # f1
```

```
    pass
```

```
def f2 (x ) :
```

```
    def t1(x):
```

```
        pass
```

```
    return t1
```

```
def f3(x, y, z):
```

```
    print(x, y, z)
```

```
def f4 f5(): pass
```

```
"""
```

```
import re
```

```
count = pos = 0
```

```
""" 提取函数名的正则表达式中，有如下要求：某一行以 def 开始，函数名必须是合法的
```

python 标识符；在此之后有一个左括号表示参数部分的开始。 """

```
regex = re.compile(_____1_____, re.M)
```

```
while True:
```

```
    match = regex.search(text, pos)
```

```
    if not match:
```

```
        break
```

```
    count += 1
```

```
    print(count, match.group(1))
```

```
    pos = _____2_____
```

答案：

序号	填空
1	<code>r'^def\s+([a-zA-Z_][a-zA-Z0-9_]*)\s*\('</code>
2	<code>match.end(1)</code>

### 三、程序改错（10%）

文本 logText 是某个监测机器人的运行日志，记录其运行期间所发生事件【如 Warning，Error，Reboot 等】及发生的时间。

下面的程序将分析 logText，并最终打印出发生的事件，次数，以及具体的发生日期，打印效果如下【注意 logText 中的日期格式是年-月-日，且月和日可能是 1 位，处理时规范成 mm/dd/yyyy】：

1. 事件Warning发生次数为4:

12/04/2019

01/23/2020

04/03/2020

06/20/2020

2. 事件Error发生次数为2:

05/03/2020

06/12/2020

3. 事件Reboot发生次数为2:

06/12/2020

06/12/2020

下述代码中两处有错误，请在后面的表格中填写错误行的位置（每行最右边注释里的数字。如果代码行后面没有注释，表示该行已经确定不会出错）和改正后的整行代码，改正后的代码只能是一行，不允许跨越多行。

```
import re
```

```
def getEvents(logText, eventsDict):
```

```
    ...
```

```
    获得日志中的事件与发生的日期，并将日期转换格式 mm/dd/yyyy，如
```

```
    2019-12-4 --> 12/04/2019
```

```
    ...
```

```
    pattern = r'(\w+):(\d{4})-(\d{1,2})-(\d{1,2})'
```

```
    matchItems = re.findall(pattern, logText) #1
```

```
    for item in matchItems: #2
```

```
        event, year, month, day = item #3
```

```
        month = '0' + month if len(month)==1 else month #4
```

```
        day = '0' + day if len(day)==1 else day #5
```

```
        dateStr = '/'.join((month, day, year)) #6
```

```
        dateList = eventsDict[event] #7
```

```
        dateList.append(dateStr) #8
```

```
        eventsDict[event] = dateList #9
```

```

def main():
    logText= '''
Robot007's WORKING... 2019-12-3
    Warning:2019-12-4
    Warning:2020-1-23
    Warning:2020-4-3
    Error:2020-5-3
    Reboot:2020-6-12
    Error:2020-6-12
    Reboot:2020-6-12
Robot007's WORKING... 2020-6-15
    Warning:2020-6-20
    ...

    eventsDict = {} #10
    getEvents(logText, eventsDict) #11

    for k,v in enumerate(eventsDict,1): #12
        print('%d. 事件%s 发生次数为%d:' % (k,v, eventsDict[v].size())) #13
        [print(date) for date in eventsDict[v]] #14

if __name__ == '__main__':
    main()

```

解答:

考察知识点: 字典、序列相关内置函数

错误行位置	改正
7	dateList = eventsDict.get(event, []) 或者 dateList = eventsDict[event] if event in eventsDict else []
13	print('%d. 事件%s 发生次数为%d:' % (k, v, len(eventsDict[v])))

## 四、编程（40%）

### 第 1 题（10%）

文本文件 example-1.txt 中有若干行字符串，由数字、字母和特殊字符组成。编写程序输出最长数字串所在的行号、长度和该数字串，如有并列最长，则输出第一个。其中行号从 0 开始计。

例如，文件中的内容如下：

```
231dfjksf12231
j12af..121011a
0091,*ajkafn
faf
9
1009001ab
```

则输出：

```
5 7 1009001
```

### # 参考答案

```
import re
maxlen=0
maxs=""
maxlineid=0
with open("example-1.txt") as f:
    lineind=0
    for line in f:
        ds=re.findall('[0-9]+',line)
        for d in ds:
            if len(d)>maxlen:
                maxlen=len(d)
                maxs=d
                maxlineid=lineind
        lineind+=1
    f.close()
print(maxlineid,maxlen,maxs)
```



## 第2题 (10%)

两个二进制整数之间的汉明距离 (Hamming distance) 指的是对应二进制位不相同的数量。  
例如:

```
117 = 0 1 1 1 0 1 0 1
17  = 0 0 0 1 0 0 0 1
H   = 0+1+1+0+0+1+0+0 = 3
```

以下程序输入二个十进制正整数, 计算它们的二进制形式的汉明距离, 请实现 `hamming` 函数。提示: 可使用函数 `bin(n)` 将十进制整数 `n` 转换为二进制整数, 如: `bin(17)` 返回 `'0b10001'`。

```
def hamming(n, m):
    # 此处添加代码, 不得修改程序的其它部分
```

```
if __name__ == '__main__':
    x, y = input('Enter 2 positive integers: ').split()
    print('The hamming distance: {}'.format(hamming(int(x),
int(y))))
```

### 【参考答案】

```
def hamming(n, m):
    if n < m:
        n, m = m, n
    n, m = bin(n)[2:], bin(m)[2:]
    m = '0' * (len(n) - len(m)) + m

    h = 0
    for i in range(len(n)):
        if n[i] != m[i]:
            h += 1
    return h
```

### 第 3 题 (20%)

小明在研究数字的低中高问题。对于两个数字，计算两个数字之差的绝对值，定义：

1) 如果该绝对值 $\geq 0$  且 $\leq 3$ ，对应'低'；2) 如果该绝对值 $> 3$  且 $\leq 6$ ，对应'中'；3) 如果该绝对值 $> 6$  且 $\leq 9$ ，对应'高'。

对于  $k$  位数  $n$ ，从左到右每两个数字依次按上述规则进行转换，可以得到长度为  $k-1$  的字符串。例如：4 位数 3812，其相邻的两个数字之差的绝对值从左到右分别是 5(对应 3 和 8)、7(对应 8 和 1)和 1(对应 1 和 2)；按上述规则转换 5、7 和 1 分别对应'中'、'高'和'低'，最终得到长度为 3 的字符串'中高低'。

在上述定义和说明的基础上，小明计划实现如下三个函数：

- 1) 函数 `get_dzg(n)`：当  $n \geq 10$  时， $n$  按上述规则转换为由低中高字符组成的字符串，我们称其为  $n$  对应的低中高值，函数返回该字符串；当  $n < 10$  时，触发异常 `ValueError`，原因为'参数值不能小于整数 10'；
- 2) 函数 `stat_dzg(k)`：其中  $k$  代表数字的位数。函数对每一个可能的  $k$  位数，比如为  $m$ ，调用函数 `get_dzg(m)` 得到其低中高值。我们想要统计各种不同的低中高值所出现的次数，采用字典记录这些信息，低中高值作为字典的 `key`，其所对应的 `value` 为低中高值为 `key` 的  $k$  位数的个数。函数返回该字典。
- 3) 函数 `stat_dzg_probability(k)`：统计所有  $k$  位数中'低'、'中'或'高'连续出现至少两次的概率，函数返回该概率值。提示：可以在调用函数 `stat_dzg` 后进行处理。

请帮助小明编写代码实现这三个函数。

# 此处添加代码，不得修改程序的其它部分

```
if "__main__" == __name__:
    probability = stat_dzg_probability(4)
    print("{}位数: {:.2f}".format(4, probability))
```

【参考答案】

```
def get_dzg(n):
    '''
    >>> get_dzg(3812)
    '中高低'
    '''
    if (not isinstance(n, int)) or (n<10):
        raise ValueError("参数值不能小于 10")
    # assert n >= 10, "n{}应当大于 10".format(n)
    maplist = '低低低低中中中高高高'
    dct = {}
    result = ""
    while n>=10:
        a, b = n//10%10, n%10
        n //= 10
        result = maplist[abs(a-b)] + result

    return result

def stat_dzg(k):
    dct = {}
    for n in range(10**(k-1), 10**(k)):
        dzg = get_dzg(n)
        dct[dzg] = dct.get(dzg, 0) + 1
    return dct

def stat_dzg_probability(k):
    dct = stat_dzg(k)
    keys = dct.keys()
    part, total = 0, sum(dct.values())
    keys = [key for key in keys if ('低低' in key) or ('中中'
' in key) or ('高高' in key)]
    for key in keys:
        part += dct.get(key,0)
    prob = part/total*100
    return prob
```