CCF 编程能力等级认证
Grade Examination of Software Programming

Python　五级

2024 年 06 月

1　单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	D	B	A	C	B	D	C	D	A	D	D	B	C	B	C

第 1 题 在Python中，print((c for c in "GESP"))的输出是（ ）。

- **A.** `('G', 'E', 'S', 'P')`
- **B.** `['G', 'E', 'S', 'P']`
- **C.** `{'G', 'E', 'S', 'P'}`
- **D.** 以上选项均不正确

第 2 题 下面有关快速排序的说法，错误的是（　）。

- **A.** 快速排序算法通常采用递归实现。
- **B.** 快速排序算法是一种稳定排序算法。
- **C.** 如果被排序数组或者list已排序或逆序，其时间复杂度是$O\left(N^2\right)$。
- **D.** 快速排序是一种原地（in-place）排序算法。

第 3 题 内排序有不同的类别，从排序算法的实现思路上考虑，下面哪种排序算法和插入排序是同一类？（ ）

- **A.** 希尔排序
- **B.** 快速排序
- **C.** 堆排序
- **D.** 冒泡排序

第 4 题 下面Python代码用于求斐波那契数列，该数列第1、2项为1，以后各项均是前两项之和。函数Fibo()属于(　)。

```python
def Fibo(N):
    if N == 1 or N == 2:
        return 1

    fiboList = [1, 1]
    for i in range(2, N):
        fiboList.append(fiboList[i - 1] + fiboList[i - 2])

    return fiboList[N-1]
```

- [ ] **A.** 枚举算法
- [ ] **B.** 贪心算法
- [ ] **C.** 迭代算法
- [ ] **D.** 递归算法

第 5 题 下面Python代码用于将输入金额换成最少币种组合方案，其实现算法是(　　)。

```python
def findCoins(coins, Money):
    coins_used = []
    for coin in coins:
        while Money >= coin:
            coins_used.append(coin)
            Money -= coin
    return coins_used

coins = [100, 50, 20, 10, 5, 2, 1] #货币种类，单位相同
M = int(input()) #输入换算的金额
coins_needed = find_coins(coins, M)

result = [(c,coins_needed.count(c)) for c in coins]
result = [x for x in result if x[1] > 0]
```

- [ ] **A.** 枚举算法
- [ ] **B.** 贪心算法
- [ ] **C.** 迭代算法
- [ ] **D.** 递归算法

第 6 题 有关下面Python的代码，错误的是(　　)。

```python
def count_if(iterData,*,key=None):
    if key == None:
        return len(iterData)

    Count = 0
    for i in iterData:
        Count += bool(key(i))
    return Count
```

- [ ] **A.** 执行 `print(count_if(range(100)))` 将输出 `100`

☐ **B.** 执行 `print(count_if(range(-10,10), key = abs))` 将输出 19

☐ **C.** 执行 `print(count_if(range(-100,10),key = lambda x:x > 5))` 将输出 4

☐ **D.** 代码 `Count += bool(key(i))` 存在错误

第 7 题 在下面的Python代码中，最后一行用于输出小于0的list，横线处不能填入的代码是(　)。

```
def LT(a, b):
    return a < b

lstData = list(range(-100,100))
print(_____________________________)
```

☐ **A.** `[x for x in lstData if x < 0 ]`

☐ **B.** `list(filter(lambda x: x < 0, lstData))`

☐ **C.** `list(filter(LT(x,0), lstData))`

☐ **D.** `[x for x in lstData if LT(x, 0)]`

第 8 题 汉字的unicode编码界于0x4E00和0x9FA5之间。下面Python的代码用于读取红楼们和水浒传文本。如果要能完整阅读这两本小说，求出需要认识的汉字集合，横线处应填入代码是（　）。

```
shzFile = open("水浒传.txt", "r", encoding = "utf-8")
hlmFile = open("红楼梦.txt", "r", encoding = "utf-8")
sSet = set(shzFile.read())
hSet = set(hlmFile.read())
shzFile.close()
hlmFile.close()

print(_____________________________________)
```

☐ **A.** `{x for x in (sSet + hSet) if 0x4E00 <= ord(x) <= 0x9FA5 }`

☐ **B.** `{x for x in (sSet | hSet) if 0x4E00 <= x <= 0x9FA5 }`

☐ **C.** `{x for x in (sSet + hSet) if 0x4E00 <= x <= 0x9FA5 }`

☐ **D.** `{x for x in (sSet | hSet) if 0x4E00 <= ord(x) <= 0x9FA5 }`

第 9 题 求回文子字符串，如：在ABCDDCBAXz中，DD、CDDC、BCDDCB、ABCDDCBA均为回文子字符串。下面Python代码是其实现，横线处应填入的代码是（　）。

```
srcStr = input()

symList = [] #保存回文子字符串
for i in range(len(srcStr)):
    for j in range(i + 2, len(srcStr) + 1):
        subStr = ____________
        if subStr == _____________:
            symList.append(subStr)

for i in sorted(symList, key = lambda x: len(x)):
    print(i)
```

- [ ] **A.** `srcStr[i:j]`， `subStr[::-1]`
- [ ] **B.** `srcStr[i:j]`， `subStr[j:i:-1]`
- [ ] **C.** `srcStr[i+2:j]`， `subStr[j-1:i:-1]`
- [ ] **D.** `srcStr[i:j+2]`， `subStr[j-1:i-1:-1]`

第 10 题 上面代码的时间复杂度是（ ）。

- [ ] **A.** $O(\log N)$
- [ ] **B.** $O(N \log N)$
- [ ] **C.** $O(N)$
- [ ] **D.** $O(N^2)$

第 11 题 有关下面Python代码的说法，错误的是（ ）。

```
def Sort(lst):
    for i in range(1, len(lst)):
        key = lst[i]
        j = i - 1
        while j >= 0 and key < lst[j]:
            lst[j + 1] = lst[j]
            j -= 1
        lst[j + 1] = key
lst = [4,5,13,2,7,10,1,3,8,11,6,9,12]
lst = Sort(lst)
print("sorted list:", lst)
```

- [ ] **A.** 该段代码是插入排序算法的实现
- [ ] **B.** 如果lst完全有序，则时间复杂度为$O(N)$
- [ ] **C.** 如果lst完全逆序，则时间复杂度为$O(N^2)$
- [ ] **D.** 由于Sort()函数没有返回值，没有最终达到排序效果

第 12 题 下面Python函数nGram()用于逐一从字符串中截取n个字符，如：nGram("ABCDEF",2)将逐一截取为AB、BC、CD、DE、EF，如：nGram("ABCDEF",3)将逐一截取为ABC、BCD、CDE、DEF，并统计每种截取的数量，横线处应填入代码是（ ）。

```
def nGram(S,n):
    Result = {}#保存截取字符串及其数量
    for i in range(_________________):
        nChar = _________________
        Result[nChar] = Result.get(nChar,0) + 1
    return Result
```

- [ ] **A.** `len(S)-n`， `S[i:n]`
- [ ] **B.** `len(S)-n+1`， `S[i:i+n]`
- [ ] **C.** `len(S)`， `S[i:i+n]`
- [ ] **D.** `len(S)-n`， `S[i:i+n]`

第 13 题 上题代码的时间复杂度是（　）。

- ☐ **A.** $O(\log N)$
- ☐ **B.** $O(N \log N)$
- ☐ **C.** $O(N)$
- ☐ **D.** $O(N^2)$

第 14 题 下面是埃氏素数筛的Python实现，横线上应填入的代码是（　）。

```
def listPrime(N):
    primeList = list(range(N+1))
    primeList[0] = primeList[1] = False
    for i in range(2,int(N ** 0.5) + 1):
        if primeList[i] != False:
            for j in range(_____________):
                primeList[j] = False
    return [x for x in primeList if x != False]
```

- ☐ **A.** i + i, N + 1, 2
- ☐ **B.** i * i, N + 1, i
- ☐ **C.** i * i, N, i * i
- ☐ **D.** i, N + 1, i

第 15 题 上题代码的时间复杂度是（　）。

- ☐ **A.** $O(N^2)$
- ☐ **B.** $O(N \log N)$
- ☐ **C.** $O(N \log \log N)$
- ☐ **D.** $O(N)$

2　判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	×	√	√	×	√	×	√	×

第 1 题 在程序设计中，i * i的效率通常比i ** 2 更高。（ ）

第 2 题 求解指定正整数范围内所有质数，采用线性筛算法比埃氏筛效率更高。（ ）

第 3 题 Python没有指针语法，不能实现C++中涉及指针的算法。（ ）

第 4 题 如果将双向链表的最后一个元素指向第一个元素，则构成环状链表。（ ）

第 5 题 链表不能采用快速排序或堆排序，但可以采用插入排序。(　)

第 6 题 在Python中，set或dict因为存储时即自动排序，因此可以用二分法查找，时间复杂度为$O(\log N)$。（ ）

第 7 题 如果自定义class已经定义了 `__lt__()` 魔术方法，则包含该类实例的数据结构，则将自动支持内置函数sorted()。（ ）

第 8 题 归并排序和快速排序都采用递归实现，也都是不稳定排序。（ ）

第 9 题 下面的Python代码能实现十进制正整数N转换为2、8、10、16，可适用于16进制以内进制。其中n和ds分别表示将转换的数以及目标进制。（ ）

```
n,ds = map(int,input().split())
rst = "" #保存转换结果

digDict = {i:c for i,c in enumerate("0123456789ABCDEF")}
while n != 0:
    rst = digDict[n % ds] + rst
    n //= ds
print(rst)
```

第 10 题 Python代码 `print(sorted(range(10),key=lambda x:x%5))` 执行时将报错。（ ）

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：黑白格
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题面描述

小杨有一个 n 行 m 列的网格图，其中每个格子要么是白色，要么是黑色。

小杨想知道至少包含 k 个黑色格子的最小子矩形包含了多少个格子。

3.1.2 输入格式

第一行包含三个正整数 n, m, k，含义如题面所示。

之后 n 行，每行一个长度为 m 的 `01` 串，代表网格图第 i 行格子的颜色，如果为 0，则对应格子为白色，否则为黑色。

3.1.3 输出格式

输出一个整数，代表至少包含 k 个黑色格子的最小子矩形包含格子的数量，如果不存在则输出 0。

3.1.4 样例1

```
4 5 5
00000
01111
00011
00011
```

```
6
```

3.1.5 样例解释

对于样例1，假设 (i,j) 代表第 i 行第 j 列，至少包含 5 个黑色格子的最小子矩形的四个顶点为 $(2,4)$，$(2,5)$，$(4,4)$，$(4,5)$，共包含 6 个格子。

3.1.6 数据范围

子任务编号	数据点占比	n,m
1	20%	≤ 10
2	40%	$n=1,1\leq m\leq 100$
3	40%	≤ 100

对于全部数据，保证有 $1\leq n,m\leq 100,1\leq k\leq n\times m$。

3.1.7 参考程序

```python
N = 110
w = [[0] * N for _ in range(N)]
sum_w = [[0] * N for _ in range(N)]

def main():
    n, m, k = map(int, input().split())
    for i in range(1, n+1):
        s = input()
        for j in range(1, m+1):
            w[i][j] = int(s[j-1])
            sum_w[i][j] = sum_w[i][j-1] + w[i][j]

    ans = 0
    for i in range(1, m+1):
        for j in range(i, m+1):
            num = []
            now = 0
            for l in range(1, n+1):
                tmp = sum_w[l][j] - sum_w[l][i-1]
                now += tmp
                num.append(now)
                if now >= k:
                    if ans == 0:
                        ans = (j-i+1) * l
                    else:
                        ans = min(ans, (j-i+1) * l)
                    L, R = 1, l
                    while L < R:
                        mid = (L + R + 1) // 2
                        if now - num[mid-1] >= k:
                            L = mid
                        else:
                            R = mid - 1
                    if now - num[L-1] >= k:
                        if ans == 0:
                            ans = (j-i+1) * (l-L)
                        else:
                            ans = min(ans, (j-i+1) * (l-L))
    print(ans)

```

```python
if __name__ == "__main__":
    main()
```

3.2 编程题 2

- **试题名称**：小杨的幸运数字
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题面描述

小杨认为他的幸运数字应该恰好有两种不同的质因子，例如，$12 = 2 \times 2 \times 3$ 的质因子有 $2, 3$，恰好为两种不同的质因子，因此 12 是幸运数字，而 $30 = 2 \times 3 \times 5$ 的质因子有 $2, 3, 5$，不符合要求，不为幸运数字。

小杨现在有 n 个正整数，他想知道每个正整数是否是他的幸运数字。

3.2.2 输入格式

第一行包含一个正整数 n，代表正整数个数。

之后 n 行，每行一个正整数。

3.2.3 输出格式

输出 n 行，对于每个正整数，如果是幸运数字，输出 1，否则输出 0。

3.2.4 样例1

```
3
7
12
30
```

```
0
1
0
```

3.2.5 样例解释

7 的质因子有 7，只有一种。

12 的质因子有 2，3，恰好有两种。

30 的质因子有 2，3，5，有三种。

3.2.6 数据范围

子任务编号	数据点占比	n	正整数值域
1	40%	≤ 100	$\leq 10^5$
2	60%	$\leq 10^4$	$\leq 10^6$

对于全部数据，保证有 $1 \leq n \leq 10^4$，每个正整数 a_i 满足 $2 \leq a_i \leq 10^6$。

3.2.7 参考程序

```python
N = 10**5 + 10
a = [0] * N

def calc(x):
    i = 2
    mp = {}
    while i * i <= x:
        if x % i == 0:
            mp[i]=1
            while x % i == 0:
                x //= i
        i += 1
    if x != 1:
        mp[x]=1
    return int(len(mp.items()))


def main():
    n = int(input())

    for i in range(1, n + 1):
        a[i] = int(input())
        x = calc(a[i])
        if x == 2:
            print("1")
        else:
            print("0")


if __name__ == "__main__":
    main()
```