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

Python 五级

2024 年 09 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	B	C	A	C	C	B	C	C	A	D	D	C	D	D	C

第 1 题 在升序数组 nums 中寻找目标值 target，下列程序可以填入的是（ ）

```
class Search(object):
    def search(self, nums, target):
        left, right = 0, len(nums) - 1
        while left <= right:
            ____________________________________
            if nums[mid] == target:
                return mid
            elif nums[mid] > target:
                right = mid - 1
            else:
                left = mid + 1
        return -1
```

- A. `mid = (right + left) // 2 + left`
- B. `mid = (right - left) // 2 + left`
- C. `mid = (right - left) // 2 -right`
- D. `mid = (right + left) // 2 - left`

第 2 题 500个病毒样本中，已知有一个是病毒检测呈阳性，用试纸测试阳性病毒以后，试纸在3天以后会变色，用试纸测试时间不计，三天以后要出结果，请问最少用多少个试纸能够找出哪一个病毒样本有毒（）

- A. `499`
- B. `250`
- C. `9`
- D. `125`

第 3 题 一名收银员，给顾客找零，找零的目标是给出确定金额的同时，使用尽可能少的硬币。有不同面额的硬币：1分，5分，10分，25分.如果需要给顾客准确的零钱77分，同时使用最少的硬币下列程序中横线应该填写（ ）。

```
def coin_change(amount, coins):
    result = []
    for coin in sorted(coins, reverse=True):
        while amount >= coin:
            ____________________
            result.append(coin)
    return result

coins = [1, 5, 10, 25]
amount = 63
```

- [ ] **A.** `amount -= coin`
- [ ] **B.** `amount <= coin`
- [ ] **C.** `amount >= coin`
- [ ] **D.** `amount += coin`

第 4 题 下列程序是素数筛的程序，横线处应该填上（ ）。

```
def sieve(n):
    if n < 2:
        return []
    prime = [True] * (n+1)
    prime[0] = prime[1] = False
    for i in range(2, int(math.sqrt(n)) + 1):
        if prime[i]:
            ________________________
                prime[j] = False
    return [p for p in range(2, n+1) if prime[p]]
for prime in sieve_of_eratosthenes(100):
    print(prime)
```

- [ ] **A.** `for j in range(i, n+1, i):`
- [ ] **B.** `for j in range(i*i, 1, n):`
- [ ] **C.** `for j in range(i*i, n+1, i):`
- [ ] **D.** `for j in range(i, n, i):`

第 5 题 下面程序是埃氏筛的一个实现，横线处应该填写（ ）。

```
n = 10**8
s = [0]*(n+1)
k=0
for i in range(2,n+1):
    if s[i]==0:
        k+=1
        ________________________
            s[j]=1
```

- [ ] **A.** `for i in range(i*i,n+1,i):`
- [ ] **B.** `for j in range(i*i,n,j):`

C. `for j in range(i*i,n+1,i):`

D. `for j in range(j*j,n+1,i):`

第 6 题 下列程序中，使用了二分查找算法，横线处应该填写的是（）。

```
def search(arr, x):
    low = 0
    high = len(arr) - 1
    while low <= high:

        ________________
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

A. `mid = (low - high) // 2`

B. `mid = (low + high) // 2`

C. `mid = (low + high) / 2`

D. `mid = (low - high) / 2`

第 7 题 正整数1024的所有约数的和为多少( )。

A. `2050`

B. `2059`

C. `2047`

D. `2044`

第 8 题 下面程序是对n！进行唯一分解，横线处应该填入的是（ ）。

```
def unique_fac(n):
    print(n, '=', end='')
    for i in range(2, n + 1):

        ________________________________
            print(' {}*'.format(i), end='')
            n //= i
        if n % i == 0 and i == n:
            print(' {}'.format(i), end='')
            break

unique_fac(math.factorial(5))
```

A. `while n % i != 0 and i != n:`

B. `while n % i == 0 and i == n:`

C. `while n % i == 0 and i != n:`

D. `while n % i != 0 and i == n:`

第 9 题 假设有一些物品，每个物品都有自己的重量，我们需要将这些物品装入箱子中，每个箱子也有自己的重量限制。贪心算法每次都选择重量最轻的物品放入当前最轻的箱子中，如果箱子可以装下，就放入；如果箱子不能装下，就尝试下一个箱子，直到找到可以放入的箱子。下列贪心算法程序中，横线处应该填入的是（ ）。

```
def box_packing(items, boxes):
  def greedy_box_packing(items, boxes):
    boxes.sort(key=lambda x: x[0])
    items.sort()
    taken = [False] * len(items)
    for i, item in enumerate(items):
        for j, box in enumerate(boxes):
            ____________________________
                taken[i] = True
                break
        else:
            raise ValueError("No box can hold the item")
    return taken
```

A. `if not taken[i] and box[0] >= items[i]:`

B. `if not taken[0] and box[0] >= items[i]:`

C. `if not taken[i] and box[i] >= items[0]:`

D. `if not taken[0] and box[0] >= items[i]:`

第 10 题 下列归并算法程序中，横线处应该填入的是（ ）。

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]
    merge_sort(left)
    merge_sort(right)
    return merge(left, right)

def merge(left, right):
    result = []
    i, j = 0, 0
    ____________________________
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    result += left[i:]
    result += right[j:]
    return result
```

A. `while i > len(left) and j < len(right):`

B. `while i < len(left) and j > len(right):`

- [ ] **C.** `while i > len(left) and j > len(right):`

- [ ] **D.** `while i < len(left) and j < len(right):`

第 11 题 下列快速排序算法中，横线处应该填入的是（ ）。

```
def quick(arr):
    if len(arr) <= 1:
        return arr
    ____________________
    left = [x for x in arr if x < p]
    middle = [x for x in arr if x == p]
    right = [x for x in arr if x > p]
    return quick(left) + middle + quick(right)
```

- [ ] **A.** `p = arr[len() // 2]`

- [ ] **B.** `p = arr[len(arr)+1 // 2]`

- [ ] **C.** `p = arr[len(arr)-1 // 2]`

- [ ] **D.** `p = arr[len(arr) // 2]`

第 12 题 下列二分枚举算法中，{ }处应该填入的程序是（{}不算做程序的一部分）（ ）。

```
def binary_search(arr, x):
    low = 0
    high = len(arr) - 1
    while low <= high:
       {

       }
    return -1
```

- [ ] **A.** ```Python
mid = (low + high) // 2
    if arr[mid] == x:
        return mid
    elif arr[mid+1] > x:
        high = mid - 1
    else:
        low = mid + 1

```
- [ ] **B.** ```Python
 mid = (low + high) // 2
        if arr[mid] != x:
            return mid
        elif arr[mid+1] > x:
            high = mid - 1
        else:
            low = mid + 1
```

- [ ] **C.** ```Python
mid = (low + high) // 2
    if arr[mid] == x:
        return mid
    elif arr[mid] > x:
        high = mid - 1
    else:
        low = mid + 1

```
- [ ] **D.** ```Python
 mid = (low + high) // 2
        if arr[mid] ! = x:
            return mid
        elif arr[mid] > x:
            high = mid - 1
        else:
            low = mid + 1
```

第 13 题 下面代码是寻找水仙花数的程序，横线处应该填写的代码是（ ）。【是指一个n位数（n≥3），其每位数字的n次幂之和等于它本身】

```
def is_narcissistic_num(num):
    str_num = str(num)
    num_digits = len(str_num)
    ____________________________
    return num == sum_of_powers

for i in range(100, 10000):
    if is_narcissistic_num(i):
        print(i, "是水仙花数")
```

- [ ] **A.** ```Python
sum_of_powers = sum(int(digit) ** num_digits for digit in num)

```
- [ ] **B.** ```Python
sum_of_powers = sum(int(digit) ** num for digit in str_num)
```

- [ ] **C.** ```Python
sum_of_powers = sum(int(num) ** num_digits for digit in str_num)

```
- [ ] **D.** ```Python
sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
```

第 14 题 对于正整数n，欧拉函数f(n),表示小于或等于n的正整数中与n互质的数的数目，例如f(8)=4。f(100)=（ ）。

- [ ] **A.** `50`
- [ ] **B.** `55`
- [ ] **C.** `45`

- [ ] **D.** `40`

第 15 题 下列程序输出的是( )。

```
def reverse(string):
    if len(string) == 0:
        return
    temp = string[0]
    reverse(string[1:])
    print(temp, end='')
string = "chen a dai"
reverse(string)
```

- [ ] **A.** `chen a dai`
- [ ] **B.** `dai a chen`
- [ ] **C.** `iad a nehc`
- [ ] **D.** `chenadai`

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 `(-1) mod 127和126 mod 127 的结果是一样的`

第 2 题 `一个数的反码，实际上是这个数对于一个模的同余数`

第 3 题 `1997和615用欧几里得算法计算最大公约数的过程如下：`

```
1997/615=3(余152)
615/152=4(余7)
152/7=21(余5)
7/5=1(余2)
5/2=2(余1)
2/1=2(余0)
得出最大公约是是1
```

第 4 题 `欧几里得算法适用于实数`

第 5 题 `每个大于1的整数可以唯一地写成质数的乘积的形式`

第 6 题 `贪婪算法的复杂度通常是线性的，即O(n)，其中n是输入的大小`

第 7 题 `归并排序的时间复杂度为O(n log n)`

第 8 题 `根据同余计算，可以推导出(a*b)%m=(a%m*b%m)%m`

第 9 题 `二分查找算法的复杂度通常表示为O(log n)，其中n是数组的长度`

第 10 题 `def(十六进制) = 103231(五进制)`

3　编程题（每题 25 分，共 50 分）

3.1　编程题 1

- **试题名称**：小杨的武器
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1　题面描述

小杨有 n 种不同的武器，他对第 i 种武器的初始熟练度为 c_i。

小杨会依次参加 m 场战斗，每场战斗小杨只能且必须选择一种武器使用，假设小杨使用了第 i 种武器参加了第 j 场战斗，战斗前该武器的熟练度为 c_i'，则战斗后小杨对该武器的熟练度会变为 $c_i'+a_j$。需要注意的是，a_j 可能是正数，0 或负数，这意味着小杨参加战斗后对武器的熟练度可能会提高，也可能会不变，还有可能降低。

小杨想请你编写程序帮他计算出如何选择武器才能使得 m 场战斗后，自己对 n 种武器的熟练度的**最大值尽可能大**。

3.1.2　输入格式

第一行包含两个正整数 n, m，含义如题面所示。

第二行包含 n 个正整数 $c_1, c_2, \ldots, c_n$，代表小杨对武器的初始熟练度。

第三行包含 m 个正整数 $a_1, a_2, \ldots, a_m$，代表每场战斗后武器熟练度的变化值。

3.1.3　输出格式

输出一个整数，代表 m 场战斗后小杨对 n 种武器的熟练度的最大值最大是多少。

3.1.4　样例1

```
2 2
9 9
1 -1
```

```
10
```

一种最优的选择方案为，第一场战斗小杨选择第一种武器，第二场战斗小杨选择第二种武器。

子任务编号	数据点占比	n	m
1	20%	$=1$	$\leq 10^5$
2	20%	$\leq 10^5$	$=2$
3	60%	$\leq 10^5$	$\leq 10^5$

对于全部数据，保证有 $1 \leq n, m \leq 10^5$，$-10^4 \leq c_i, a_i \leq 10^4$。

3.1.5　参考程序

```python
N = 100010

a = [0] * N
c = [0] * N

n, m = map(int, input().split())
```

```
mx = -10000
s = input().split()
for i in range(1, n + 1):
    c[i] = int(s[i-1])
    mx = max(mx, c[i])
s = input().split()
for i in range(1, m + 1):
    a[i] = int(s[i-1])

for i in range(1, m + 1):
    if n == 1 or a[i] > 0:
        mx += a[i]

print(mx)
```

3.2 编程题 2

- **试题名称**：挑战怪物
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题面描述

小杨正在和一个怪物战斗，怪物的血量为 h，只有当怪物的血量**恰好**为 0 时小杨才能够成功击败怪物。

小杨有两种攻击怪物的方式：

- 物理攻击。假设当前为小杨第 i 次使用物理攻击，则会对怪物造成 2^{i-1} 点伤害。
- 魔法攻击。小杨选择任意一个质数 x（x 不能超过怪物当前血量），对怪物造成 x 点伤害。由于小杨并不擅长魔法，他只能使用**至多一次**魔法攻击。

小杨想知道自己能否击败怪物，如果能，小杨想知道自己最少需要多少次攻击。

3.2.2 输入格式

第一行包含一个正整数 t，代表测试用例组数。

接下来是 t 组测试用例。对于每组测试用例，第一行包含一个正整数 h，代表怪物血量。

3.2.3 输出格式

对于每组测试用例，如果小杨能够击败怪物，输出一个整数，代表小杨需要的最少攻击次数，如果不能击败怪物，输出 -1。

3.2.4 样例1

```
3
6
188
9999
```

```
2
4
-1
```

对于第一组测试用例，一种可能的最优方案为，小杨先对怪物使用魔法攻击，选择质数 5 造成 5 点伤害，之后对怪物使用第 1 次物理攻击，造成 $2^{1-1}=1$ 点伤害，怪物血量恰好为 0，小杨成功击败怪物。

子任务编号	数据点占比	t	h
1	20%	≤ 5	≤ 10
2	20%	≤ 10	≤ 100
3	60%	≤ 10	$\leq 10^5$

对于全部数据，保证有 $1 \leq t \leq 10, 1 \leq h \leq 10^5$。

3.2.5 参考程序

```python
prime = []
is_prime = [False] * 100010

def Eratosthenes(n):
    is_prime[0] = is_prime[1] = False
    for i in range(2, n + 1):
        is_prime[i] = True
    for i in range(2, n + 1):
        if is_prime[i]:
            prime.append(i)
            if i * i > n:
                continue
            for j in range(i * i, n + 1, i):
                is_prime[j] = False

Eratosthenes(100000)

t = int(input())
for _ in range(t):
    tmp = 1
    x = int(input())
    ans = 0
    while True:
        if is_prime[x]:
            ans += 1
            break
        x -= tmp
        ans += 1
        if x <= 0:
            if x < 0:
                ans = -1
            break
        tmp *= 2
    print(ans)
```