1. **阅读程序(程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填X，每题均为4分)**

1.

|  |  |
| --- | --- |
| 1  2 | a, b = list(map(int,input().split()))  print(a+b) |

1. 该代码适用于求解两个整形的数之和。..........................................( ) 对
2. 读入“114 514”，输出“628”。.....................................................( )

2.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | n = int(input())  assert n % 2 == 0  a = list(map(int, input().split()))  a.insert(0, 0) # 数组里面的值从下标1开始  for i in range(1, n+1, 2):  if a[i] > a[i+1]:  a[i], a[i+1] = a[i+1], a[i]  minn, maxx = a[1], a[2]  for i in range(1, n+1, 2):  minn = min(minn, a[i])  maxx = max(maxx, a[i+1])  print("min = %d" % (minn))  print("max = %d" % (maxx)) |

1. 读入的n为奇数也不妨碍程序运行。..................................................( )
2. 该代码可求出数组的最小值和最大值。................................................( )
3. 求出n个数(n为偶数)的最大和最小值最少要比较次。.............................( )

3.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | string = list(input()) # 保证输入为字符串  n = len(string)  string.insert(0, 0) # 下标从1开始  ans = 0  cnt = [0] \* 255  for i in range(1, n+1):  cnt[ord(string[i])] += 1  ans = max(ans, cnt[ord(string[i])])  print(ans) |

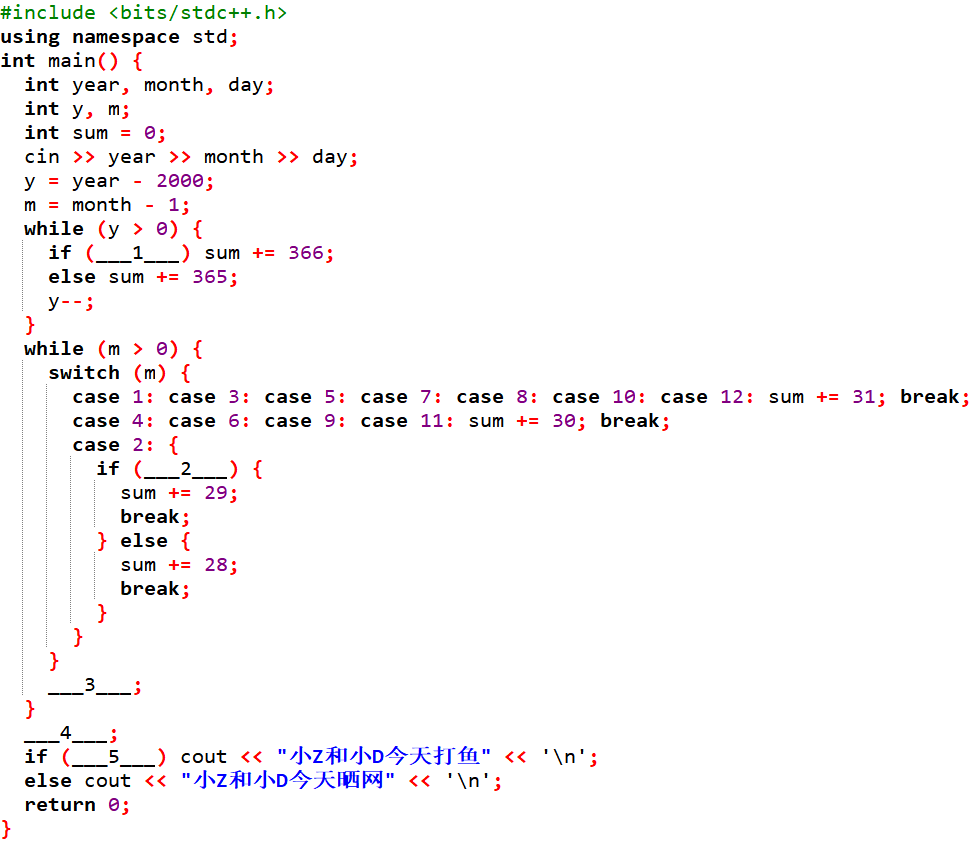
1. 代码可读入一个空串。..............................................................( )
2. 最终cnt[字母]表示该字母在字符串中出现的次数。....................................( )
3. 输入串“dadkhfkjdfdsfhdsaufuhsadusaudaufewurufewufewifh”，输出“8”。..............( )

4.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | def popcount(x: int):  ret = 0  while x > 0:  ret += 1  x &= x-1  return ret  ans = 0  for i in range(1, 1025):  ans += popcount(i)  print(ans) |

1. 该代码可以求1~1024这些数的二进制位为1的个数和。.................................( )
2. 该代码的运行结果是( ).
3. 5120 B. 5121 C. 2560 D. 2561
4. **完善程序(单选题，每小题3分，共计30分)**
5. (三天打鱼，两天晒网)

中国有句俗话叫做“三天打鱼，两天晒网”。妹子小Z和汉子小D从2000年的1月1号开始，按“三天打鱼，两天晒网”的方式生活。试通过编程来判断，在未来某一天，小Z和小D是打鱼还是晒网。



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | year, month, day = list(map(int, input().split()))  y, m = year-2000, month – 1  sum = 0  while y > 0:  if ①:  sum += 366  else:  sum += 365  y -= 1  common\_mon = [4, 6, 9, 11]  greater\_mon = [1, 3, 5, 7, 8, 10, 12]  while m > 0:  if m == 2:  if ②:  sum += 29  else:  sum += 28  elif m in common\_mon:  sum += 30  elif m in common\_mon:  sum += 31  ③  ④  if ⑤:  print("小Z和小D今天打鱼")  else:  print("小Z和小D今天晒网") |

1. sum += day
2. sum%5==1 or sum%5==2 or sum%5==3

C. m -= 1

1. (year%4==0 and year%100!=0) or (year%400==0)
2. (y-1)%4==0 or y==1
3. y%4==0 or y==1
4. m += 1
5. year%4 == 0
6. (反击的号角)



《「空之轨迹」百日战役》背景介绍：

七耀历1192年春，位于利贝尔北面的埃雷波尼亚帝国对王国突然宣战，「百日战役」正式爆发。战争开始后的仅仅一个月内，帝国军便依仗强大的兵力快速突破哈肯大门，然后占领了除王都格兰赛尔以外的利贝尔整片国土。但是两个月之后，利贝尔的王国军利用飞艇对帝国军进行“闪电作战”，并封锁地区之间的关所，令到在各处作战的帝国军部队——被孤立。再加上得到卡尔瓦德共和国的协助，断绝了帝国的补给线，并将被侵占的地方逐个收复。之后，由于七耀教会和游击士协会从中居间调停，两国缔结了和平条约，持续了大约一百天的战争终告结束。

「问题描述」

现在你知道利贝尔总共有m个地区，现在已经有n个地区被帝国军控制了，每个地区的帝国军军力为a\_i。由于帝国军对他们的军力进行了加码，所以军力少的地方也有可能有重兵把守。但是你有一种特殊的解码方法，这种方法为：如果(a\_i + lasta) mod k为一个质数，则i读取没有重兵把守(lasta为上一个没有重兵把守地区的军力大小)。你需要求出的是总共有多少个地区没有重兵把守。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | def init(n: int):  global vis, pr, length  vis[1] = 1  for i in range(①, n+1):  if not vis[i]:  ②  for j in range(length):  if ③:  break  vis[pr[j]\*i] = 1  if i % pr[j] == 0:  break  N = 100005  vis, pr = [0] \* N, [0] \* N  len = 0  init(int(1e5))  m, n, k = list(map(int, input().split()))  ans, lasta = m, 0  for i in range(1, n+1):  x = int(input())  x = ④  if ⑤:  ans -= 1  else:  lasta = x  print(ans) |

1. 2 B. 0

C. pr[length] = i D. length += 1

length += 1 pr[length] = i

1. pr[i] \* j > n F. pr[j] \* i > n

G. (x + lasta) % k H. x % k

I. vis[x] J. not vis[x]

二、阅读程序（程序输入不超过数组定义的范围；选择题单选；共计40分）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | a, b = list(map(int, input().split())) # 0 <= a,b <= 10^9  while 1:  x = a^b  y = a&b  if y == 0:  break  a = x  b = y << 1  print(x) |

* 判断题

1. （1分）第8行的 << 意义是移位运算符。( ) 答案：对
2. （1.5分）第3行计算得到的x每次比上一次计算得到的大。( )
3. （1.5分）存在一组注释所示范围内的a, b使程序会陷入死循环。( )
4. （2分）若输入a = 155, b = 229，程序各执行第7,8行八次。( )

* 选择题

1. （3分）设为 ，最优情况下，此程序的时间复杂度是( )。  
   A. B. C. D.
2. （4分）设为 ，最坏情况下，此程序的时间复杂度是( )。  
   A. B. C. D.

2.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | import queue as Queue  def solve(s: int, res: list):  que = Queue.Queue()  for u in range(n):  res[u] = INF  res[s] = 0  que.put(s)  while not que.empty():  u = que.get()  for v in range(n):  if res[v] > res[u] + a[u][v]:  res[v] = res[u] + a[u][v]  que.put(v)  INF = int(1e9)  n, m = list(map(int, input().split()))  a = [[INF] \* n for i in range(n)]  f, g = [0]\*1000, [0]\*1000  for i in range(m):  u, v, w = list(map(int, input().split()))  if a[u][v] > w:  a[u][v] = w  if a[v][u] > w:  a[v][u] = w  solve(0, f)  solve(1, g)  ans = 0  for u in range(n):  if ans < f[u]+g[u]:  ans = f[u]+g[u]  print(ans) |

* 判断题

1. （1分）a数组为邻接表，是一种存储无向图的结构。( )
2. （1分）将第16行INF的取值改成2e9不影响程序运行结果。( )

答案：对，

注：python没有精度限制

1. （1.5分）函数solve(s: int, res: list)中的res是list引用传递，在函数内对res数组的修改会同时作用于作为实参的数组。( )
2. （1.5分）若输出大于等于INF，则0号结点不与1号结点连通。( )

* 选择题

1. （4分）若图连通且边权全为1，此时程序输出6，n 最小为( )。  
   A. 4 B. 5 C. 6 D. 7
2. （4分）忽略m，最坏情况下，此程序的时间复杂度是( )。  
   A. B. C. D.

3.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | maxn = 1000002;  n, a, first, last, pre, suf = 0, [0]\*maxn, [0]\*maxn, [0]\*maxn, [0]\*maxn, [0]\*maxn  def main():  global n, a, first, last, pre, suf  n = int(input()) # 1 <= n <= 1000000  for i in range(1, n+1):  a[i] = int(input()) # 1 <= a[i] <= n  first[i] = n + 1;  for i in range(1, n+1):  if first[a[i]] == n + 1:  first[a[i]] = i  for i in range(n, 0, -1):  if last[a[i]] == 0:  last[a[i]] = i  pre[0] = 0  for i in range(1, n+1):  pre[i] = pre[i - 1];  if pre[i] > first[i]:  pre[i] = n + 1;  if pre[i] < last[i]:  pre[i] = last[i]  suf[n + 1] = n + 1;  for i in range(n, 0, -1):  suf[i] = suf[i + 1]  if suf[i] < last[i]:  suf[i] = 0;  if suf[i] > first[i]:  suf[i] = first[i]  ans = 0;  j = 1  for i in range(1, n+1):  while j <= n and suf[j + 1] <= pre[i - 1]:  j += 1  if j < i:  j = i  ans += n + 1 - j  print(ans)  main() |

* 判断题

1. （1分）对于，若不存在a[i] = x，则程序执行到第15行  
   时last[x] 为0。 ( )
2. （1.5分）若对满足a[i-1] <= a[i]，则输出为 。( )
3. （1.5分）存在一组输入使得输出为2。( )
4. （2分）程序输出时，对一定满足pre[i] >= suf[i]。( )

* 选择题

1. （4分）若n = 10且程序输出18，输入的a数组从1到n可能是：( )。  
   A. 1 6 2 5 3 4 8 10 9 7

B. 1 6 2 5 3 4 8 9 10 7

C. 1 6 2 5 8 3 4 10 9 7

D. 1 6 2 5 8 3 4 9 10 7

1. （4分）以下哪项操作不会影响程序运行结果：( )  
   A. 交换第18-19行和第20-21行的代码

B. 将第25行的 < 修改为 <=

C. 将第32行的 <= 修改为 <

D. 移除第34-35行的代码

三、完善程序（单选；每小题3分，共计30分）  
1.（最速洗牌）一套牌由张数字牌（从1到）和张空牌（用0标注）组成，其中张牌在你手中，其余形成牌堆。一次操作可以将手中的任意一张牌置入牌堆底，并从牌堆顶摸入新的手牌。求最少多少次操作能让最终手牌均为空牌，且

牌堆**从上到下**第张牌恰好数值为

输入第一行给定数字牌个数。  
第二行有个数，分别表示初始你的每张手牌，空牌以0标注。

第三行有个数，第个数表示初始从**从上到下**第张牌的数值，空牌以0标注。

输出达到目标最少的操作数。

可以采用贪心解决这个问题，最优策略一定是下面两种之一：

* 存在，依次从手牌中取出张空牌，再依次取出数值牌
* 存在，依次从手牌中取出数值牌

试补全下面的程序。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | b, p = [0]\*100001, [0]\*100001  def main():  ans = 0  n = int(input())  a = list(map(int, input()))  a.insert(0, 0)  for i in range(1, n+1):  b[i] = int(input())  ①  for i in range(1, n+1):  t = ②  if ans < t:  ans = t  if p[1] != 0:  i = 2  while i <= n and ③  i += 1  if p[i - 1] == n:  j = i  while j <= n and ④:  j += 1  if j > n:  t = n - i + 1  if ⑤:  ans = t  print(ans)  main() |

1. ①处应填( )

A. p[i] = b[i]

B. p[n + 1 – i] = b[i]

C. p[b[i]] = i

D. p[b[i]] = n + 1 – i

1. ②处应填( )

A. p[i] + i - 1 + n

B. i – p[i] – 1 + n

C. i - p[i] + 1 + n

D. p[i] - i + 1 + n

1. ③处应填( )

A. p[i] == p[1] + i - 1

B. p[i] == p[1] - i + 1

C. p[i] >= p[1] + i - 1

D. p[i] >= p[1] - i + 1

1. ④处应填( )

A. p[j] == j – i

B. p[j] <= j – i

C. p[j] == p[1] - j + 1

D. p[j] >= p[1] + j - 1

1. ⑤处应填( )

A. ans < t

B. ans > t

C. i <= n

D. i <= t

2.（幸运数字）云云有个幸运数字。此外，他定义正整数是厄运的，当且仅当它不是任何一个幸运数字的倍数。他想知道有多少个小于等于的正整数是厄运的。

输入第一行给定幸运数字的个数和上界。

第二行有个数，即幸运数字。

输出合法正整数的个数。

试补全下面的程序。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | a, k, n, ans = [0]\*20, 0, 0, 0  def get(x: int, y: int):  return y if not x else ①  def dfs(i: int, s: int, prod: int):  global ans, a, k, n  if i == n:  ans += ②  return  dfs(i + 1, s, prod)  if k // prod >= a[i] // get(prod, a[i]):  dfs(i + 1, ③, ④)  def main():  global ans, a, k, n  n, k = list(map(int, input().split()))  a = list(map(int, input().split()))  dfs(0, ⑤, 1)  print(ans)  main() |

1. ①处应填( )

A. get(x % y, y)

B. get(y % x, x)

C. get(y, x % y)

D. get(x, y % x)

1. ②处应填( )

A. s \* k // prod

B. s \* k // get(prod, k)

C. k – s \* k // prod

D. k – s \* k // get(prod, k)

1. ③处应填( )

A. -s

B. not s

C. 0 if get(prod, a[i]) != 1 else s

D. s if get(prod, a[i]) != 1 else -s

1. ④处应填( )

A. prod \* a[i]

B. prod \* a[i] // get(prod, a[i])

C. prod \* get(prod, a[i])

D. prod // get(prod, a[i]) \* a[i]

1. ⑤处应填( )

A. 0

B. -1

C. 1

D. -1 if n % 2 else 1