**Министерство цифрового развития, связи и массовых**

**коммуникаций Российской Федерации**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра “Математическая кибернетика и информационные технологии”

Курсовая работа

по дисциплине: «Структуры и алгоритмы обработки данных»

Вариант №9

Выполнил: студент группы БФИ 2001

Климов Н.Ю.

Проверил: Мкртчян Г.М.

Москва 2022

**Оглавление**

[Задание №1 3](#_Toc103102214)

[Задание №2 6](#_Toc103102215)

[Задание №3 11](#_Toc103102216)

[Задание №4 13](#_Toc103102278)

[Задание №5 20](#_Toc103102279)

[Задание №6 24](#_Toc103102280)

[Задание №7 29](#_Toc103102281)

[Задание №8 32](#_Toc103102282)

[Задание №9 34](#_Toc103102283)

[Задание №10 40](#_Toc103102284)

[Вывод 48](#_Toc103102285)

**Цель работы:** требуется найти решение десяти задач девятого варианта

# **Задание №1**

**Условие задания**

Перестановкой длины 𝑛 называется такой массив 𝑝=[𝑝1,𝑝2,…,𝑝𝑛], который содержит каждое число от 1 до 𝑛 (включительно) и притом ровно по одному разу. Например, 𝑝=[3,1,4,2,5] — перестановка длины 5.

Для заданного числа 𝑛 (𝑛≥2) найдите такую перестановку 𝑝, в которой разница (то есть модуль разности) любых двух соседних элементов находится в диапазоне от 2 до 4, включительно. Формально, для перестановки 𝑝 должно выполняться 2≤|𝑝𝑖−𝑝𝑖+1|≤4 для всех 𝑖 (1≤𝑖<𝑛).

Выведите любую такую перестановку для заданного значения 𝑛 или определите, что ее не существует.

**Входные данные**

В первой строке записано целое число 𝑡 (1≤𝑡≤100) — количество наборов входных данных в тесте. Далее следуют описания 𝑡 наборов входных данных.

Каждый набор описывается единственной строкой, в которой содержится целое число 𝑛 (2≤𝑛≤1000).

**Выходные данные**  
Выведите 𝑡 строк. Очередная строка должна содержать -1, если для соответствующего набора входных данных искомой перестановки не существует. В противном случае выведите искомую перестановку. Если таких перестановок существует несколько, то выведите любую из них.

**Пример**  
**Входные данные**  
6  
10  
2  
4  
6  
7  
13  
**Выходные данные**  
9 6 10 8 4 7 3 1 5 2  
-1  
3 1 4 2  
5 3 6 2 4 1

5 1 3 6 2 4 7

13 9 7 11 8 4 1 3 5 2 6 10 12

**Решение задания**

def permutations(n):

output\_string = ""

if massiv\_n[i] <= 3:

output\_string = -1

else:

for j in range(massiv\_n[i],0,-1):

if (j % 2 == 1):

output\_string += str(j) + " ";

output\_string += "4 2 "

for j in range(6,massiv\_n[i]+1):

if (j % 2 == 0):

output\_string += str(j) + " ";

return output\_string

def correct\_t(t):

if t.isdigit() and int(t) <= 100 and int(t) >=1:

return int(t)

else:

return -1

def massiv\_n\_input(t):

massiv\_n = []

for i in range(t):

massiv\_n.append(input("Введите длинну(n): "))

return massiv\_n

def correct\_massiv\_n(massiv\_n,t):

counter\_correct\_input = 0

for i in range(t):

if massiv\_n[i].isdigit() and int(massiv\_n[i]) >= 2 and int(massiv\_n[i]) <= 1000:

counter\_correct\_input += 1

if (counter\_correct\_input == t):

for i in range(t):

massiv\_n[i] = int(massiv\_n[i])

return massiv\_n

else:

return -1

def main():

t = input("Введите количество наборов входных данных: ")

t = correct\_t(t)

if t != -1:

massiv\_n = massiv\_n\_input(t)

if correct\_massiv\_n(massiv\_n,t) != -1:

for i in range(t):

print(permutations(massiv\_n[i]))

else:

print("Некорректный ввод значений длины(n)!")

else:

print("Неверное значение количества тестовых наборов!")

main()

# **Задание №2**

**Условие задания**

Вам задана массив 𝑎 длины 𝑛, состоящий из нулей. Вы выполняете 𝑛 действий с этим массивом: в течение 𝑖-го действия происходит следующая последовательность операций:

Выбирается максимальный по длине подмассив (последовательный подотрезок), состоящий только из нулей, среди всех таких отрезков выбирается самый левый; Пусть этот отрезок равен [𝑙;𝑟]. Если 𝑟−𝑙+1 нечетно (не делится на 2), то присваивается 𝑎[𝑙+𝑟2]:=𝑖 (где 𝑖 — номер текущего действия), иначе (если 𝑟−𝑙+1 четно) присваивается 𝑎[𝑙+𝑟−12]:=𝑖.

Рассмотрим массив 𝑎 длины 5 (изачально 𝑎=[0,0,0,0,0]). Тогда он меняется следующим образом:

Сначала мы выбираем отрезок [1;5] и присваиваем 𝑎[3]:=1, таким образом 𝑎 становится равен [0,0,1,0,0];

затем мы выбираем отрезок [1;2] и присваиваем 𝑎[1]:=2, таким образом 𝑎 становится равен [2,0,1,0,0];

затем мы выбираем отрезок [4;5] и присваиваем 𝑎[4]:=3, таким образом 𝑎 становится равен [2,0,1,3,0];

затем мы выбираем отрезок [2;2] и присваиваем 𝑎[2]:=4, таким образом 𝑎 становится равен [2,4,1,3,0];

и наконец мы выбираем отрезок [5;5] и присваиваем 𝑎[5]:=5, таким образом 𝑎 становится равен [2,4,1,3,5].

Ваша задача — найти массив 𝑎 длины 𝑛 после выполнения всех 𝑛 действий. Заметьте, что ответ существует и единственен.

Вам необходимо ответить на 𝑡 независимых наборов тестовых данных.

**Входные данные**

Первая строка входных данных содержит одно целое число 𝑡 (1≤𝑡≤104) — количество наборов тестовых данных. Затем следуют 𝑡 наборов тестовых данных.

Единственная строка набора тестовых данных содержит одно целое число 𝑛 (1≤𝑛≤2⋅105) — длину 𝑎.

Гарантируется, что сумма 𝑛 по всем наборам тестовых данных не превосходит 2⋅105 (∑𝑛≤2⋅105).

**Выходные данные**

Для каждого набора тестовых данных выведите ответ — массив 𝑎 длины 𝑛 после выполнения 𝑛 действий, описанных в условии задачи. Заметьте, что ответ существует и единственен.

**Пример**

**Входные данные**  
6  
1  
2  
3  
4  
5  
6  
**Выходные данные**  
1  
1 2  
2 1 3  
3 1 2 4  
2 4 1 3 5  
3 4 1 5 2 6

**Решение задания**

def massiv\_input\_data\_gen(t):

massiv\_input\_data = []

for i in range(t):

massiv\_input\_data.append(input("Введите длинну(n): "))

return massiv\_input\_data

def correct\_t(t):

if t.isdigit() and int(t) <= 104 and int(t) >=1:

return int(t)

else:

return -1

def correct\_massiv\_input\_data(massiv\_input\_data,t):

counter\_correct\_input = 0

for i in range(t):

if massiv\_input\_data[i].isdigit() and int(massiv\_input\_data[i]) >= 1 and int(massiv\_input\_data[i]) <= 200000:

counter\_correct\_input += 1

if (counter\_correct\_input == t):

for i in range(t):

massiv\_input\_data[i] = int(massiv\_input\_data[i])

return massiv\_input\_data

else:

return -1

def correct\_permutations\_output(result):

result = str(result)

result = result.replace(',','')

result = result.replace('[','')

result = result.replace(']','')

return result

def permutations(massiv\_input\_data):

result = []

substring\_of\_nulls = []

current\_string = []

left\_edge = 0

right\_edge = 0

counter\_pair\_nulls = 0

index = 1

for j in range(massiv\_input\_data):

result.append(0)

substring\_of\_nulls = "".join(map(str, result[0:(len(result)//2)]))

if (massiv\_input\_data % 2 == 0):

result[(massiv\_input\_data - 2) // 2] = index

index += 1

else:

result[(massiv\_input\_data - 1) // 2] = index

index += 1

current\_string = "".join(map(str, result[0:len(result)]))

while substring\_of\_nulls.count('0') != 0:

counter\_pair\_nulls = current\_string.count(substring\_of\_nulls)

for i in range(counter\_pair\_nulls):

left\_edge = current\_string.find(substring\_of\_nulls)

right\_edge = left\_edge + len(substring\_of\_nulls) - 1

if substring\_of\_nulls.count('0') % 2 == 0:

result[(left\_edge + right\_edge - 1) // 2] = index

current\_string = "".join(map(str, result[0:len(result)]))

index += 1

else:

result[(left\_edge + right\_edge) // 2] = index

current\_string = "".join(map(str, result[0:len(result)]))

index += 1

substring\_of\_nulls = substring\_of\_nulls[:len(substring\_of\_nulls) - 1]

return correct\_permutations\_output(result)

def main():

t = input("Введите количество наборов текстовых данных: ")

t = correct\_t(t)

massiv\_input\_data = []

if t != -1:

massiv\_input\_data = massiv\_input\_data\_gen(t)

if correct\_massiv\_input\_data(massiv\_input\_data,t) != -1:

for i in range(t):

print(permutations(massiv\_input\_data[i]))

else:

print("Некорректный ввод значений n!")

else:

print("Неверное значение количества тестовых наборов!")

main()

# **Задание №3**

**Условие задания**

Поликарп очень любит гамбургеры, особенно приготовленные собственноручно. Поликарп считает, что существует только три достойных ингредиента для приготовления гамбургера: хлеб, колбаса и сыр. Рецепт своего знаменитого «гамбургера от Поликарпа» он записывает в виде строки из букв 'B' (хлеб), 'S' (колбаса) и 'C' (сыр). В рецепте ингредиенты перечисляются снизу вверх, например, рецепт «ВSCBS» обозначает гамбургер в котором снизу вверх идут: хлеб, колбаса, сыр, хлеб и снова колбаса.

На кухне у Поликарпа в наличии nb кусочков хлеба, ns кусочков колбасы и nc кусочков сыра. Кроме того, в магазине неподалеку есть в продаже все три ингредиента по цене: pb рублей за кусок хлеба, ps — за кусок колбасы и pc — за кусочек сыра.

У Поликарпа есть r рублей, которые он готов потратить в магазине. Какое наибольшее количество гамбургеров он сможет приготовить? Считается, что ломать или резать любой из кусочков (хлеба, колбасы, сыра) нельзя, а также что в магазине есть бесконечно много кусочков каждого из ингредиентов.

**Входные данные**

В первой строке входных данных содержится непустая строка, описывающая рецепт «гамбургера от Поликарпа». Длина строки не превосходит 100, строка содержит только буквы 'B' (прописная латинская B), 'S' (прописная латинская S) и 'C' (прописная латинская C).

Вторая строка содержит три целых числа nb, ns, nc (1 ≤ nb, ns, nc ≤ 100) — количество кусочков хлеба, колбасы и сыра на кухне Поликарпа. Третья строка содержит три целых числа pb, ps, pc (1 ≤ pb, ps, pc ≤ 100) — цена одного кусочка хлеба, колбасы и сыра в магазине. Наконец, четвертая строка содержит целое число r (1 ≤ r ≤ 1012) — количество рублей у Поликарпа.

Пожалуйста, не используйте спецификатор %lld для чтения или записи 64-битных чисел на С++. Рекомендуется использовать потоки cin, cout или спецификатор %I64d.

**Выходные данные**

Выведите наибольшее количество гамбургеров, которое Поликарп может приготовить. Если он не может приготовить ни одного гамбургера, выведите 0.

**Примеры**

**Входные данные**  
BBBSSC  
6 4 1  
1 2 3  
4  
**Выходные данные**2  
**Входные данные**  
BBC  
1 10 1  
1 10 1  
21  
**Выходные данные**7  
**Входные данные**BSC  
1 1 1  
1 1 3  
1000000000000  
**Выходные данные**200000000001

**Решение задания**

import sys

def recipt\_input():

return input("Введите рецепт гамбургера: ")

def count\_of\_ingridient\_input():

return input("Введите количество ингредиентов: ").split()

def price\_of\_ingridient\_input():

return input("Введите стоимость ингредиенов: ").split()

def amount\_of\_money\_input():

return input("Введите количество денег: ")

def correct\_recipt(recipt):

counter = 0

for i in range(len(recipt)):

if recipt[i].upper() == "B" or recipt[i].upper() == "S" or recipt[i].upper() == "C":

counter += 1

if counter == len(recipt):

return recipt

else:

return -1

def correct\_count\_of\_ingridient(count\_of\_ingridient):

counter\_correct\_input = 0

if len(count\_of\_ingridient) == 3:

for i in range(3):

if count\_of\_ingridient[i].isdigit() and int(count\_of\_ingridient[i]) >= 1 and int(count\_of\_ingridient[i]) <= 100:

counter\_correct\_input += 1

if counter\_correct\_input == 3:

for i in range(3):

count\_of\_ingridient[i] = int(count\_of\_ingridient[i])

return count\_of\_ingridient

else:

return -1

else:

return -1

def correct\_price\_of\_ingridient(price\_of\_ingridient):

counter\_correct\_input = 0

if len(price\_of\_ingridient) == 3:

for i in range(3):

if price\_of\_ingridient[i].isdigit() and int(price\_of\_ingridient[i]) >= 1 and int(price\_of\_ingridient[i]) <= 100:

counter\_correct\_input += 1

if counter\_correct\_input == 3:

for i in range(3):

price\_of\_ingridient[i] = int(price\_of\_ingridient[i])

return price\_of\_ingridient

else:

return -1

else:

return -1

def correct\_amount\_of\_money(amount\_of\_money):

if amount\_of\_money.isdigit() and int(amount\_of\_money) <= 1024 and int(amount\_of\_money) >=1:

return int(amount\_of\_money)

else:

return -1

def polycarp(recipt, count\_of\_ingridient, price\_of\_ingridient, amount\_of\_money):

b\_in\_reciept = 0

s\_in\_reciept = 0

c\_in\_reciept = 0

massiv\_for\_calc\_result = []

massiv\_for\_search\_min = [sys.maxsize,sys.maxsize,sys.maxsize]

current\_min\_ingridient\_index = 0

for i in recipt:

if i.upper() == 'B':

b\_in\_reciept += 1

elif i.upper() == 'S':

s\_in\_reciept += 1

elif i.upper() == 'C':

c\_in\_reciept += 1

if b\_in\_reciept != 0:

massiv\_for\_search\_min[0] = (count\_of\_ingridient[0] / b\_in\_reciept)

if s\_in\_reciept != 0:

massiv\_for\_search\_min[1] = (count\_of\_ingridient[1] / s\_in\_reciept)

if c\_in\_reciept != 0:

massiv\_for\_search\_min[2] = (count\_of\_ingridient[2] / c\_in\_reciept)

if amount\_of\_money != 0:

while amount\_of\_money > 0:

current\_min\_ingridient\_index = massiv\_for\_search\_min.index(min(massiv\_for\_search\_min))

if current\_min\_ingridient\_index == 0 and amount\_of\_money-price\_of\_ingridient[0] >= 0 and b\_in\_reciept != 0:

count\_of\_ingridient[0] += 1

massiv\_for\_search\_min[0] = (count\_of\_ingridient[0] / b\_in\_reciept)

amount\_of\_money -= price\_of\_ingridient[0]

elif current\_min\_ingridient\_index == 1 and amount\_of\_money-price\_of\_ingridient[1] >= 0 and s\_in\_reciept != 0:

count\_of\_ingridient[1] += 1

massiv\_for\_search\_min[0] = (count\_of\_ingridient[1] / s\_in\_reciept)

amount\_of\_money -= price\_of\_ingridient[1]

elif current\_min\_ingridient\_index == 2 and amount\_of\_money-price\_of\_ingridient[2] >= 0 and c\_in\_reciept != 0:

count\_of\_ingridient[2] += 1

massiv\_for\_search\_min[2] = (count\_of\_ingridient[2] / c\_in\_reciept)

amount\_of\_money -= price\_of\_ingridient[2]

if b\_in\_reciept != 0:

massiv\_for\_calc\_result.append(count\_of\_ingridient[0] // b\_in\_reciept)

if s\_in\_reciept != 0:

massiv\_for\_calc\_result.append(count\_of\_ingridient[1] // s\_in\_reciept)

if c\_in\_reciept != 0:

massiv\_for\_calc\_result.append(count\_of\_ingridient[2] // c\_in\_reciept)

result = min(massiv\_for\_calc\_result)

else:

if massiv\_for\_search\_min[0] != 0:

massiv\_for\_calc\_result.append(count\_of\_ingridient[0] // b\_in\_reciept)

if massiv\_for\_search\_min[1] != 0:

massiv\_for\_calc\_result.append(count\_of\_ingridient[1] // s\_in\_reciept)

if massiv\_for\_search\_min[2] != 0:

massiv\_for\_calc\_result.append(count\_of\_ingridient[2] // c\_in\_reciept)

return min(massiv\_for\_calc\_result)

def main():

recipt = recipt\_input()

if correct\_recipt(recipt) != -1:

count\_of\_ingridient = count\_of\_ingridient\_input()

if correct\_count\_of\_ingridient(count\_of\_ingridient) != -1:

price\_of\_ingridient = price\_of\_ingridient\_input()

if correct\_price\_of\_ingridient(price\_of\_ingridient) != -1:

amount\_of\_money = amount\_of\_money\_input()

if correct\_amount\_of\_money(amount\_of\_money) != -1:

amount\_of\_money = correct\_amount\_of\_money(amount\_of\_money)

print("Наибольшее количество гамбургеров для готовки: " + str(polycarp(recipt, count\_of\_ingridient, price\_of\_ingridient, amount\_of\_money)))

else:

print("Некорректный ввод количества денег!")

else:

print("Некорректный ввод стоимости ингридиентов!")

else:

print("Некорректный ввод количества ингридиентов!")

else:

print("Некорректный ввод рецепта!")

main()

**Задание №4**

**Условие задания**

Ayush и Ashish играют в игру на некорневом дереве, состоящем из 𝑛 вершин, пронумерованных от 1 до 𝑛. Игроки делают следующий ход по очереди:

Выберите любой лист в дереве и удалите его вместе со всеми ребрами, для которых этот лист является одним из концов. Лист — это вершина со степенью, не превосходящей 1. Дерево — это связный неориентированный граф без циклов.

Дана специальная вершина с номером 𝑥. Игрок, который удаляет эту вершину, выигрывает игру.

Ayush ходит первым. Определите победителя игры, если каждый игрок играет оптимально.

Входные данные

В первой строке входных данных содержится одно целое число 𝑡 (1≤𝑡≤10) — количество наборов входных данных. Далее следуют описания наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа 𝑛 и 𝑥 (1≤𝑛≤1000,1≤𝑥≤𝑛) — количество вершин в дереве и специальную вершину, соответственно.

Каждая из следующих 𝑛−1 строк содержит два целых числа 𝑢, 𝑣 (1≤𝑢,𝑣≤𝑛, 𝑢≠𝑣), что означает, что между вершинами 𝑢 и 𝑣 есть ребро.

Выходные данные

Для каждого набора входных данных, если побеждает Ayush, выведите "Ayush", иначе выведите "Ashish" (без кавычек).

Примеры

**Входные данные**

1

3 1

2 1

3 1

**Выходные данные**

Ashish

**Входные данные**

1

3 2

1 2

1 3

**Выходные данные**

Ayush

**Примечание**

В первом наборе входных данных Ayush может удалить только вершину 2 или 3, после чего вершина 1 становится листом, и Ashish может удалить ее в свою очередь.

Во втором наборе входных данных Ayush может удалить вершину 2 на самом первом шаге.

**Решение задания**

def correct\_t(t):

if t.isdigit() and int(t) <= 10 and int(t) >=1:

return int(t)

else:

return -1

def input\_n\_and\_x():

return input("Введите n и x: ").split()

def input\_vertexes(n):

vertexes = []

for i in range(n - 1):

vertexes.append(input("Введите значения вершины: ").split())

return vertexes

def correct\_n\_and\_x(n\_and\_x):

if len(n\_and\_x) == 2:

if n\_and\_x[0].isdigit() and n\_and\_x[1].isdigit():

n\_and\_x[0] = int(n\_and\_x[0])

n\_and\_x[1] = int(n\_and\_x[1])

if n\_and\_x[0] >= 1 and n\_and\_x[0] <= 1000 and n\_and\_x[1] >= 1 and n\_and\_x[0] >= n\_and\_x[1]:

return n\_and\_x

else:

return -1

else:

return -1

else:

return -1

def correct\_vertexes(vertexes,n,x):

counter\_correct\_input = 0

for i in range(n - 1):

if len(vertexes[i]) == 2:

if vertexes[i][0].isdigit() and vertexes[i][1].isdigit():

vertexes[i][0] = int(vertexes[i][0])

vertexes[i][1] = int(vertexes[i][1])

if vertexes[i][0] >= 1 and vertexes[i][1] <= n and vertexes[i][0] != vertexes[i][1]:

counter\_correct\_input += 1

else:

return -1

if counter\_correct\_input == n - 1:

return vertexes

else:

return -1

def leaf\_game(n\_and\_x,vertexes):

n,x\_special= n\_and\_x[0], n\_and\_x[1]

tree = [[] for \_ in range(n+1)]

for i in range(n\_and\_x[0]-1):

x,y = vertexes[i][0], vertexes[i][1]

tree[x].append(y)

tree[y].append(x)

if len(tree[x\_special]) <= 1 or n\_and\_x[0] % 2 == 0:

return "Ayush"

else:

return "Ashish"

def main():

t = input("Введите количество тестовых наборов: ")

t = correct\_t(t)

if t != -1:

n\_and\_x = input\_n\_and\_x()

if correct\_n\_and\_x(n\_and\_x) != -1:

vertexes = input\_vertexes(n\_and\_x[0])

if correct\_vertexes(vertexes,n\_and\_x[0],n\_and\_x[1]) != -1:

for i in range(t):

print("Победитель: " + str(leaf\_game(n\_and\_x,vertexes)))

else:

print("Некорректные значения вершин!")

else:

print("Некорректные входные значения n и x!")

else:

print("Неверное значение количества тестовых наборов!")

main()

**Задание №5**

**Условие задания**

Вы играете в очередную компьютерную игру, и теперь вам предстоит убить 𝑛 монстров. Эти монстры стоят в круге, пронумерованном по часовой стрелке от 1 до 𝑛. Изначально 𝑖-й монстр имеет 𝑎𝑖 единиц здоровья.

Вы можете стрелять в монстров, чтобы убить их. Каждый выстрел требует ровно одной пули и уменьшает здоровье монстра на 1 (наносит ему 1 единицу урона). Кроме того, когда здоровье некоторого монстра 𝑖 становится 0 или меньше 0, он умирает и взрывается, нанося 𝑏𝑖 урон следующему монстру (монстру под номером 𝑖+1, если 𝑖<𝑛, или монстру под номером 1, если 𝑖=𝑛). Если следующий монстр уже мертв, то ничего не происходит. Если взрыв убивает следующего монстра, он тоже взрывается, повреждая монстра после него и, возможно, вызывая еще один взрыв, и так далее.

Вы должны посчитать минимальное количество пуль, которое нужно выстрелить, чтобы убить всех 𝑛 монстров в кругу.

**Входные данные**

Первая строка содержит одно целое число 𝑇 (1≤𝑇≤150000) — количество наборов входных данных.

Затем следуют наборы входных данных, каждый из них начинается со строки, содержащей одно целое число 𝑛 (2≤𝑛≤300000) — количество монстров. Затем следуют 𝑛 строк, каждая из которых содержит два целых числа 𝑎𝑖 и 𝑏𝑖 (1≤𝑎𝑖,𝑏𝑖≤1012) — параметры 𝑖-го монстра в круге.

Гарантируется, что общее количество монстров во всех тестовых случаях не превышает 300000.

**Выходные данные**

Для каждого набора входных данных выведите одно целое число — минимальное количество пуль, которые нужно выстрелить, чтобы убить всех монстров.

**Пример**  
**Входные данные**1  
3  
7 15  
2 14  
5 3  
**Выходные данные**  
6

**Решение задания**

def correct\_t(t):

if t.isdigit() and int(t) <= 150000 and int(t) >=1:

return int(t)

else:

return -1

def correct\_n(n):

if n.isdigit() and int(n) <= 300000 and int(n) >=2:

return int(n)

else:

return -1

def input\_monster\_propertys(n):

monster\_propertys = []

for i in range(n):

monster\_propertys.append(input("Введите hp и damage монстра: ").split())

return monster\_propertys

def correct\_monster\_propertys(monsters\_hp\_and\_damage, n):

counter\_correct\_input = 0

for k in range(n):

if len(monsters\_hp\_and\_damage[k]) == 2:

for i in range(2):

if int(monsters\_hp\_and\_damage[k][i]) >= 1 and int(monsters\_hp\_and\_damage[k][i]) <= 1012 and monsters\_hp\_and\_damage[k][i].isdigit():

counter\_correct\_input += 1

else:

return -1

if counter\_correct\_input == n \* 2:

for i in range(n):

for j in range(2):

monsters\_hp\_and\_damage[i][j] = int(monsters\_hp\_and\_damage[i][j])

return monsters\_hp\_and\_damage

else:

return -1

def computer\_game(n, monster\_propertys):

least\_hp,bs\_damage= monster\_propertys[1][0], monster\_propertys[1][1]

first\_monster=least\_hp

all\_patrons=0

for i in range(1,n-1):

a,b= monster\_propertys[i][0], monster\_propertys[i][1]

least\_hp=min(least\_hp,a,bs\_damage)

all\_patrons+=max(0,a-bs\_damage)

bs\_damage=b

all\_patrons+=max(0,first\_monster-bs\_damage)

return all\_patrons+min(least\_hp,first\_monster,bs\_damage)

def main():

t = input("Введите количество тестовых наборов: ")

t = correct\_t(t)

if t != -1:

n = input("Введите количество монстров: ")

n = correct\_n(n)

if n != -1:

monster\_propertys = input\_monster\_propertys(n)

if correct\_monster\_propertys(monster\_propertys, n) != -1:

for i in range(t):

print("Минимальное количество пуль: " + str(computer\_game(n, monster\_propertys)))

else:

print("Некорректный ввод параметров монстров!")

else:

print("Неверное значение количества монстров!")

else:

print("Неверное значение количества тестовых наборов!")

main()

**Задание №6**

**Условие задания**

Отель Гильберта это очень необычный отель, потому что количество комнат в нем бесконечно! Для каждого целого числа существует ровно одна комната с таким номером, включая ноль и отрицательные числа. Не менее странно то, что сейчас отель полностью заполнен, что означает, что в каждой комнате находится ровно один гость. Менеджер отеля, сам Давид Гильберт, решил переместить гостей, потому что у него есть предположение, что за счет этого образуются свободные места.

Для любого целого числа 𝑘 и положительного целого числа 𝑛 обозначим за 𝑘mod𝑛 остаток при делении числа 𝑘 на число 𝑛. Более формально, 𝑟=𝑘mod𝑛 это наименьшее неотрицательное целое число такое, что 𝑘−𝑟 делится на 𝑛. Всегда выполнено, что 0≤𝑘mod𝑛≤𝑛−1. Например, 100mod12=4 и (−1337)mod3=1.

Процесс перемещения гостей выглядит следующим образом: есть массив, состоящий из 𝑛 целых чисел 𝑎0,𝑎1,…,𝑎𝑛−1. Тогда для всех целых чисел 𝑘 гость из комнаты с номером 𝑘 перемещается в комнату с номером 𝑘+𝑎𝑘mod𝑛.

Определите, верно ли, что после этого процесса перемещения в каждой комнате по-прежнему находится ровно один гость. Это означает, что нет пустых комнат и комнат, в которых больше одного гостя.

**Входные данные**

Каждый тест состоит из нескольких тестовых случаев. Первая строка содержит единственное целое число 𝑡 (1≤𝑡≤104) — количество тестовых случаев. Следующие 2𝑡 строк содержат описания тестовых случаев.

В первой строке описания каждого тестового случая находится единственное целое число 𝑛 (1≤𝑛≤2⋅105) — длина массива.

Во второй строке описания каждого тестового случая находятся 𝑛 целых чисел 𝑎0,𝑎1,…,𝑎𝑛−1 (−109≤𝑎𝑖≤109).

Гарантируется, что сумма 𝑛 по всем тестовым случаям не превосходит 2⋅105.

**Выходные данные**

Для каждого тестового случая выведите единственную строку, содержащую «YES», если в каждой комнате после перемещения находится ровно один гость, и «NO» иначе. Вы можете выводить каждый символ в любом регистре.

**Пример**

**Входные данные**

6

1

14

2

1 -1

4

5 5 5 1

3

3 2 1

2

0 1

5

-239 -2 -100 -3 -11

**Выходные данные**

YES

YES

YES

NO

NO

YES

**Примечание**

В первом тестовом случае номер комнаты каждого гостя увеличился на 14, поэтому по-прежнему в каждой комнате находится ровно один гость.

Во втором тестовом случае гости в комнатах с четными номерами перемещаются в комнату с номером, на 1 большим исходного; гости в комнатах с нечетными номерами перемещаются в комнату с номером, на 1 меньшим исходного. Можно показать, что по-прежнему в каждой комнате находится ровно один гость.

В третьем тестовом случае каждый четвертый гость перемещается в комнату с номером, на 1 большим исходного, а остальные гости перемещаются в комнату с номером, на 5 большим. Можно показать, что по-прежнему в каждой комнате находится ровно один гость.

В четвертом тестовом случае гости, исходно находящиеся в комнатах 0 и 1, перемещаются в комнату с номером 3.

В пятом тестовом случае гости, находящиеся в комнатах 1 и 2, перемещаются в комнату с номером 2.

**Решение задания**

def correct\_t(t):

if t.isdigit() and int(t) <= 104 and int(t) >=1:

return int(t)

else:

return -1

def correct\_n(n):

if n.isdigit() and int(n) <= 200000 and int(n) >=1:

return int(n)

else:

return -1

def a\_input():

return input("Введите массив чисел: ").split()

def correct\_a(a,n):

if len(a) != n:

return -1

counter\_correct\_input = 0

for i in range(n):

if a[i].isdigit() and int(a[i]) >= -109 and int(a[i]) <= 109:

counter\_correct\_input += 1

if (counter\_correct\_input == n):

for i in range(n):

a[i] = int(a[i])

return a

else:

return -1

def gilberts\_hotel(a,n):

set\_of\_relocation=set()

for i in range(n):

r=(i+a[i%n])%n

set\_of\_relocation.add(r)

if len(set\_of\_relocation)!=n:

return "NO"

else:

return "YES"

def main():

t = input("Введите количество тестовых наборов: ")

t = correct\_t(t)

if t != -1:

for j in range(t):

n = input("Введите длину массива чисел: ")

n = correct\_n(n)

if n != -1:

a = a\_input()

if correct\_a(a,n) != -1:

print("Результат: " + str(gilberts\_hotel(a,n)))

else:

print("Некорректный ввод значений массива чисел!")

else:

print("Некорректный ввод длинны массива чисел!")

else:

print("Неверное значение длины массива или количества тестовых наборов!")

main()

**Задание №7**

Несколько дней назад я узнал, что существует такая штука как наименьшее общее кратное (НОК). Теперь я часто играю с этим понятием — хочу сделать большое число с помощью НОК.

Но я не хочу использовать слишком много чисел, поэтому я выберу три целых положительных числа (необязательно различных), каждое из которых не превышает n. Помогите мне найти максимально возможное наименьшее общее кратное этих трех целых чисел.

**Входные данные**

В первой строке записано целое число n (1 ≤ n ≤ 106) — переменная n из условия.

**Выходные данные**

Выведите единственное целое число — максимально возможное наименьшее общее кратное трех необязательно различных целых чисел, которые не превышают n.

**Примеры**

**Входные данные**

9

**Выходные данные**

504

**Входные данные**

7

**Выходные данные**

210

**Примечание**

Наименьшее общее кратное нескольких положительных целых чисел — это наименьшее положительное целое число, кратное им всем.

Результат может получиться достаточно большим. Возможно, 32-битного целого числа не будет достаточно для его хранения. Поэтому рекомендуется использовать 64-битные целые числа.

В последнем примере мы можем выбрать числа 7, 6, 5, их НОК равен 7·6·5 = 210. Это — максимальный НОК, который мы можем получить.

**Решение задания**

def correct\_n(n):

if n.isdigit() and int(n) <= 106 and int(n) >=1:

return int(n)

else:

return -1

def maxNOK(n):

if n < 3:

return str(n)

elif n % 2 == 1:

return str(n \* (n - 1) \* (n - 2))

elif n % 3 == 1:

return str(n \* (n - 1) \* (n - 3))

else:

return str((n - 1) \* (n - 2) \* (n - 3))

def main():

n = input("Введите число n: ")

n = correct\_n(n)

if n != -1:

print("Максимальное НОК равен: " + maxNOK(n))

else:

print("Неверное значение n!")

main()

**Задание №8**

**Условие задания**

Для мультимножества натуральных чисел 𝑠={𝑠1,𝑠2,…,𝑠𝑘}, определим наименьшее общее кратное («LCM» по-английски) и наибольший общий делитель («GCD» по-английски) 𝑠 следующим образом:

gcd(𝑠) это максимальное натуральное число 𝑥, такое что все числа из 𝑠 делятся на 𝑥.

lcm(𝑠) это минимальное натуральное число 𝑥, которое делится на все числа из 𝑠.

Например, gcd({8,12})=4,gcd({12,18,6})=6 и lcm({4,6})=12. Обратите внимание, что для любого натурального числа 𝑥, gcd({𝑥})=lcm({𝑥})=𝑥.

У Орака есть последовательность 𝑎 длины 𝑛. Он придумал мультимножество 𝑡={lcm({𝑎𝑖,𝑎𝑗}) | 𝑖<𝑗} и попросил вас найти gcd(𝑡) для него. Иначе говоря, вам нужно найти НОД НОКов всех пар элементов в данной последовательности.

**Входные данные**

В первой строке записано одно целое число 𝑛 (2≤𝑛≤100000).

Во второй строке записаны 𝑛 целых чисел, 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤200000).

**Выходные данные**

Выведите одно целое число: gcd({lcm({𝑎𝑖,𝑎𝑗}) | 𝑖<𝑗}).

**Примеры**

**Входные данные**

2

1 1

**Выходные данные**

1

**Входные данные**

4

10 24 40 80

**Выходные данные**

40

**Входные данные**

10

540 648 810 648 720 540 594 864 972 648

**Выходные данные**

54

**Примечание**

В первом примере 𝑡={lcm({1,1})}={1}, и gcd(𝑡)=1.

Во втором примере 𝑡={120,40,80,120,240,80}. Нетрудно видеть, что gcd(𝑡)=40.

**Решение задания**

def correct\_n(n):

if n.isdigit() and int(n) <= 100000 and int(n) >=2:

return int(n)

else:

return -1

def massiv\_s\_input(n):

massiv\_s\_input = []

massiv\_s\_input = input("Введите " + str(n) + " целых чисел: ")

massiv\_s\_input = massiv\_s\_input.split()

return massiv\_s\_input

def correct\_massiv\_s(massiv\_s,n):

counter\_correct\_input = 0

if len(massiv\_s) == n:

for i in range(n):

if massiv\_s[i].isdigit() and int(massiv\_s[i]) >= 1 and int(massiv\_s[i]) <= 200000:

counter\_correct\_input += 1

if (counter\_correct\_input == n):

for i in range(n):

massiv\_s[i] = int(massiv\_s[i])

return massiv\_s

else:

return -1

else:

return -1

def search\_NOKs(s):

massiv\_of\_NOKs = []

max\_number\_couple = 0

min\_number\_couple = 0

step\_search\_NOK = 0

flag = 0

for i in range(len(s)):

for j in range(i+1,len(s)):

step\_search\_NOK = max(s[i],s[j])

max\_number\_couple = step\_search\_NOK

min\_number\_couple = min(s[i],s[j])

while flag == 0:

if max\_number\_couple % min\_number\_couple == 0 and massiv\_of\_NOKs.count(max\_number\_couple) == 0:

massiv\_of\_NOKs.append(max\_number\_couple)

flag = 1

elif max\_number\_couple % min\_number\_couple == 0 and massiv\_of\_NOKs.count(max\_number\_couple) != 0:

flag = 1

else:

max\_number\_couple += step\_search\_NOK

step\_search\_NOK = 0

min\_number\_couple = 0

max\_number\_couple = 0

flag = 0

return massiv\_of\_NOKs

def search\_HOD\_of\_NOKs(s):

max\_del = 0

current\_del = 0

counter\_every\_del = 0

massiv\_of\_NOKs = search\_NOKs(s)

current\_del = min(massiv\_of\_NOKs)

while current\_del != 0:

for i in massiv\_of\_NOKs:

if i % current\_del == 0:

counter\_every\_del += 1

if counter\_every\_del == len(massiv\_of\_NOKs):

max\_del = current\_del

break

else:

counter\_every\_del = 0

current\_del -= 1

return max\_del

def main():

n = input("Введите длину последовательности: ")

n = correct\_n(n)

s = []

if n != -1:

s = massiv\_s\_input(n)

if correct\_massiv\_s(s,n) != -1:

print("НОД НОКов всех пар равен: " + str(search\_HOD\_of\_NOKs(s)))

else:

print("Некорректный ввод массива чисел!")

else:

print("Неверное значение длины последовательности!")

main()

**Задание №9**

**Условие задания**

Василий очень любит разные загадки. Сегодня он нашёл загадку, которую не смог решить сам, поэтому он просит вас помочь ему.

У Василия есть n строк, состоящих из строчных букв английского алфавита. Он хочет, чтобы строки располагались в лексикографическом порядке (как в словаре), но при этом не хочет менять их местами. Единственное, что Василий может делать, это разворачивать строки (первая буква становится последней, вторая предпоследней и так далее).

Чтобы развернуть i-ю строку Василию, надо потратить ci единиц энергии. Василия интересует минимальное количество энергии, которое необходимо потратить, чтобы строки шли в лексикографическом порядке.

Строка A лексикографически меньше строки B, если она короче B (|A| < |B|) и является её префиксом, либо ни одна из них не является префиксом другой, и в первой позиции, где они различаются, в строке A стоит символ с меньшим номером.

В данной задаче две одинаковые строки на соседних позициях не нарушают порядок лексикографической сортировки.

**Входные данные**

В первой строке входных данных записано единственное целое число n (2 ≤ n ≤ 100 000) — количество строк.

Во второй строке записаны n целых чисел ci (0 ≤ ci ≤ 109), i-е из которых равняется количеству энергии, необходимому Василию для разворота i-й строки.

Далее следуют n строк, состоящих из строчных букв английского алфавита. Суммарная длина всех строк не превышает 100 000.

**Выходные данные**

Если, разворачивая какие-либо из данных строк, невозможно добиться, чтобы они следовали в лексикографическом порядке, то выведите  - 1. В противном случае выведите минимальное количество энергии, которое придётся потратить Василию.

**Примеры**

**Входные данные**

2

1 2

Ba

Ac

**Выходные данные**

1

**Входные данные**

3

1 3 1

aa

ba

ac

**Выходные данные**

1

**Входные данные**

2

5 5

Bbb

Aaa

**Выходные данные**

-1

**Входные данные**

2

3 3

Aaa

Aa

**Выходные данные**

-1

**Примечание**

Во втором примере можно развернуть строку 2 или строку 3. На разворот строки 3 тратится меньше энергии, поэтому правильным ответом будет развернуть её. В третьем примере обе строки не изменяются после разворота и расположены в неправильном порядке, поэтому ответом является  - 1. В четвёртом примере обе строки состоят из букв «a», но в отсортированном порядке строка «aa» должна располагаться раньше строки «aaa», поэтому ответ  - 1.

**Решение задания**

def correct\_n(n):

if n.isdigit() and int(n) <= 100000 and int(n) >=2:

return int(n)

else:

return -1

def input\_ci():

return input("Введите количество энергии для разворота i-ой строки: ").split()

def input\_strings(n):

strings = []

for i in range(n):

strings.append(input("Введите строку: "))

return strings

def correct\_ci(ci,n):

if len(ci) != n:

return -1

counter\_correct\_input = 0

for i in range(n):

if ci[i].isdigit() and int(ci[i]) >= 0 and int(ci[i]) <= 109:

counter\_correct\_input += 1

if (counter\_correct\_input == n):

for i in range(n):

ci[i] = int(ci[i])

return ci

else:

return -1

def correct\_strings(strings,n):

counter\_correct\_lc = 0

counter\_correct\_input = 0

for i in range(n):

for j in range(len(strings[i])):

if strings[i][j].isalpha() and strings[i][j].islower():

counter\_correct\_lc += 1

if len(strings[i]) == counter\_correct\_lc:

counter\_correct\_input += 1

counter\_correct\_lc = 0

else:

return -1

if (counter\_correct\_input == n):

return strings

else:

return -1

def energy\_strings(n,ci,strings):

start\_and\_end = []

result = 0

current\_string = 0

flags = []

for i in range(len(ci)):

start\_and\_end.append(list(strings[i][::]))

flags.append(0)

while current\_string < len(strings) - 1:

for i in range(min(len(start\_and\_end[current\_string]),len(start\_and\_end[current\_string+1])) - 1):

if ord(start\_and\_end[current\_string][i]) > ord(start\_and\_end[current\_string+1][i]) and ci[current\_string] < ci[current\_string + 1] and flags[current\_string] != 1:

start\_and\_end[current\_string].reverse()

flags[current\_string] = 1

result += ci[current\_string]

if ord(start\_and\_end[current\_string][i]) > ord(start\_and\_end[current\_string+1][i]) and ci[current\_string] > ci[current\_string + 1] and flags[current\_string] != 1:

start\_and\_end[current\_string+1].reverse()

flags[current\_string + 1] = 1

result += ci[current\_string + 1]

elif not(ord(start\_and\_end[current\_string][i]) <= ord(start\_and\_end[current\_string+1][i])):

result = -1

break

if "".join(map(str,start\_and\_end[current\_string])).find("".join(map(str,start\_and\_end[current\_string + 1]))) != -1 and len(start\_and\_end[current\_string]) > len(start\_and\_end[current\_string+1]):

result = -1

break

current\_string += 1

return result

def main():

n = input("Введите количество строк: ")

n = correct\_n(n)

if n != -1:

ci = input\_ci()

strings = input\_strings(n)

if correct\_ci(ci,n) != -1 and correct\_strings(strings,n) != -1:

print("Минимальное количество энергии: " + str(energy\_strings(n,ci,strings)))

else:

print("Неверный ввод значений энергии или строк!")

else:

print("Неверное значение количества строк!")

main()

**Задание №10**

**Условие задания**

У Феникса есть строка 𝑠, состоящая из строчных букв латинского алфавита. Он хочет распределить все буквы своей строки по 𝑘 непустым строкам 𝑎1,𝑎2,…,𝑎𝑘 так, что каждая буква из 𝑠 попадет ровно в одну из строк 𝑎𝑖. Строки 𝑎𝑖 не обязаны быть подстроками 𝑠. Феникс может распределить буквы 𝑠 и переупорядочить их внутри каждой строки 𝑎𝑖 так как захочет.

Например, если 𝑠= baba и 𝑘=2, Феникс может распределить буквы своей строки множеством способов, в том числе:

ba и ba a и abb ab и ab bb и aa Однако получить такие варианты он не может:

baa и ba b и ba baba и пустая строка (𝑎𝑖 должны быть непустыми) Феникс хочет разделить свою строку 𝑠 на 𝑘 строк 𝑎1,𝑎2,…,𝑎𝑘 так, чтобы минимизировать лексикографически максимальную строку среди них, т. е. минимизировать 𝑚𝑎𝑥(𝑎1,𝑎2,…,𝑎𝑘). Помогите ему найти оптимальное распределение и выведите минимально возможное значение 𝑚𝑎𝑥(𝑎1,𝑎2,…,𝑎𝑘).

Строка 𝑥 лексикографически меньше, чем строка 𝑦, если либо 𝑥 является префиксом 𝑦 (и 𝑥≠𝑦), либо существует такой индекс 𝑖 (1≤𝑖≤𝑚𝑖𝑛(|𝑥|,|𝑦|)), что 𝑥𝑖 < 𝑦𝑖 и для всех 𝑗 (1≤𝑗<𝑖) 𝑥𝑗=𝑦𝑗. Здесь |𝑥| обозначает длину строки 𝑥.

**Входные данные**

Входные данные состоят из нескольких наборов. В первой строке задано целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных. Каждый набор состоит из двух строк.

В первой строке каждого набора задано два целых числа 𝑛 и 𝑘 (1≤𝑘≤𝑛≤105) — длина строки 𝑠 и количество не пустых строк, в которые Феникс хочет распределить буквы 𝑠, соответственно.

Во второй строке каждого набора задана строка 𝑠 длины 𝑛, состоящая из строчных латинских букв.

Гарантируется, что сумма 𝑛 по всем наборам входных данных ≤105.

**Выходные данные**

Выведите 𝑡 ответов — по одному на набор входных данных; 𝑖-й ответ — минимально возможный 𝑚𝑎𝑥(𝑎1,𝑎2,…,𝑎𝑘) в 𝑖-м наборе.

**Пример**

**Входные данные**

6

4 2

Baba

5 2

Baacb

5 3

Baacb

5 3

Aaaaa

6 4

Aaxxzz

7 1

Phoenix

**Выходные данные**

ab

abbc

b

aa

x

ehinopx

**Примечание**

В первом наборе входных данных, одно из оптимальных решений — разбить baba на ab и ab.

Во втором наборе входных данных, одно из оптимальных решений — разбить baacb на abbc и a.

В третьем наборе, одно из оптимальных решений — разбить baacb на ac, ab и b.

В четвертом наборе, одно из оптимальных решений — разбить aaaaa на aa, aa и a.

В пятом наборе, одно из оптимальных решений — разбить aaxxzz на az, az, x и x.

В шестом наборе, одно из оптимальных решений — разбить phoenix на ehinopx.

**Решение задания**

def correct\_t(t):

if t.isdigit() and int(t) <= 1000 and int(t) >=1:

return int(t)

else:

return -1

def correct\_n\_and\_k(n\_and\_k):

counter\_correct\_input = 0

if len(n\_and\_k) > 2:

return -1

if n\_and\_k[0].isdigit() and int(n\_and\_k[0]) <= 105 and int(n\_and\_k[0]) >= 1:

counter\_correct\_input += 1

else:

return -1

if n\_and\_k[0].isdigit() and int(n\_and\_k[1]) <= 105 and int(n\_and\_k[1]) >= 1 and int(n\_and\_k[0]) >= int(n\_and\_k[1]):

counter\_correct\_input += 1

else:

return -1

if (counter\_correct\_input == 2):

for i in range(2):

n\_and\_k[i] = int(n\_and\_k[i])

return n\_and\_k

else:

return -1

def correct\_a(a):

counter = 0

if a.count(' ') != 0:

return -1

for i in range(len(a)):

if a[i].islower() and a[i].isalpha():

counter += 1

if counter == len(a):

return a

else:

return -1

def minmax\_combination(t,n,k,a):

a = list(a)

a.sort()

result1 = [''] \* k

result2 = [''] \* k

for i in range(n):

result1[i % k] += a[i]

for i in range(k):

result2[i] += a[i]

for i in range(k, n):

result2[0] += a[i]

return min(max(result1), max(result2))

def main():

t = input("Введите количество наборов входных данных: ")

n\_and\_k = input("Введите n и k: ").split()

a = input("Введите набор символов: ")

t = correct\_t(t)

if t != -1:

if correct\_n\_and\_k(n\_and\_k) != -1:

if correct\_a(a) != -1:

for i in range(t):

print("Результат: " + str(minmax\_combination(t,n\_and\_k[0],n\_and\_k[1],a))

else:

print("Последовательность a неверно введена!")

else:

print("Некорректный ввод длины строки или количества не пустых строк")

else:

print("Неверное значение количества тестовых наборов!")

main()

# **Вывод**

В качестве результата выполнения данной курсовой работы были разработаны алгоритмы для решения десяти задач из девятого варианта курсовой работы