Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Модульна контрольна робота
з дисципліни “Мультипарадигменне програмування”

Виконав:
студент групи ІО-12
Ільйов Д.А.
Залікова книжка №1214

Київ 2024 р.

Завдання

На імперативній мові програмування реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів з подальшою побудовою матриці передування.

Варіант:

10	Розподіл Рейлі

10	F	B-C-D-E-F-Dow Jones Industrial Average Historical Data

Час виконання

Парадигма програмування	Мова програмування	Час виконання
Імперативна парадигма	C	0.066391 secs
Функціональна парадигма	Common Lisp	0.165329 secs
Гібридна парадигма	R	0.3166871 secs
Логічна парадигма	Prolog	0.260000 secs
Продуційна парадигма	CLIPS	0.189213 secs

Лінгвістичні ланцюжки та матриці передування

C

Лінгвістичний ланцюжок

char array:
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Y Z Z Z Z Z Z Z Z Z Z Z Y Y Z Z Z Z Y Z Z Z Z Z Z Z Z Z Z Z Z Z Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Z Z Z Z Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Z Z Z Z Z Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Z Y Y Y Y Y Y Z Y Y Y Y Y Y Y Z Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y X X Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y X X X Y X Y Y Y X X X X X X X
X X X X X X X X X X X X X X X X X X X X X W W W X X W X W W W X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X X X X X X X W X W W W X X X W W W W W W W W W W W W W W W W W W W W W W W W
W W W W W W W W W W W W V V V V V V V V W W W W V V V V V V V V V V V V V V V V V V V V V V V V V V V V V U V U U V V V V
V V V V V V V V V V V V V V V U U U U U U U U U U U U U T U U U U U U V V V V V V V V V V V V V V V V V V V U V V V V V V V V V V
V V V V V V U U U U U U U U U U U U U U U U U U U U U U U U U U U U U V V V U U U V V V V V U U U U U U V U U U U U U U U U U U U
U U U U U U U U U U U U U U U T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T S S S S S S S
S S S S S S R R S S S S S S S S S S S R S S S S S S S S S S S S S S S S S S S S R R R R R R R R R R R R R R R R R R R R R R R R R R R R R S S R
R R R R R R S R S R R R R R R R R R R R R R R R R R R R R Q Q Q Q R Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q P
P P P P P P P P P P P P P P P P P P P P P P P P P P O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
O O O O O O O O O O O N N N N N N N N N N N N N N N N N N N N N N N N O N N N O O N N N O O O O O O O O O O O O O O O O O O N N
N N N O O O O O O O O O O O O O O O O O O O O O O O O O O O N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N
N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N M M M N N N N N N N N N M
M M N N N N N O O O O O O O O O O O O O O N N N O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O N N N N N
N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N M N N N N N N N N N N N N N N N M M M M M M M M M
M M M M M M M M M M M N M M N M M M N N N N N N N N M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M
M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M L M M M M M M M M M M L M L L L L M
M M M M M M M L L L L L M M M M M M M M M M M M M L M M M L L L L L L L L L L L L L L L L L L M L L L L L L L L L L L L L L
L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L K K K K K K K L K K L L K K L K K L K K K K K K K K K K K K K K

K K K K K L L L L L L L L L L L L L L L L L L L L L L L L K K K L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L
L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L K K K K L K K K L L K L L L L L L L
K K K K K K K L L L L L L L L L L K K K L L L K K L L L K K K K L L K K K K L L K L K K K K K K K K K K K K K J J J J J J J J I I J J J J J J
J J J J J J J J K J K J K J J J J K K J J J J J K K J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J
J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J I J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J I I J J J J J J J J J J J J J J J J J
J J J J J J J I I I I I I I I H H H H H H H H H H H G G H G G H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H
G G G H H G G G G G G G H H H G G G G G G G G G H G G H H G G G G G G H H H H H H H H H H H H H H H H H H H H H H H H H H I I I I I I I I
I I I J J J J J J I J J J J J J J J J J J J J J J I I I I J J J J J J J J J J J J J J J I I I I I I I I I I I I I I I I I I H H H H H H H I I I I I I I I I I I I I I I I I I I I I I I I
I I H I I I H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H I I I I I I I I I I I I I I I I H I I I I I I I I I I I I I I I I H H H H
H H H H H H H H H H H H H H H H H H H H H H H H H H H I I I I I I I I I I I I H H H I I I I I H H I I I I I I I I I I I I I I H H H I I I I H I I I I H H H H H H H
H H H H H H H H H H H H H H G G G G G G H H H H G G G G G G H H H H H H G H H G G G H G G G H G G G G G G G G G G G G G G G G G G G
G G G G G G G G G G G G G G G G G F F F F F F F F F F G G G F F G G G F F F F F F F F F G G G F F F G G G G G G G G G G G G F F F F F F F F F F
F F F F F F F F F F F F F F F F F F F F F F F F F F F F F E F F E E E E E E E E E E E E E E F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F E E E E
E F E E F F E E E E E E E E E E E E E E E E E E E E E E E E E E E F F F F F F E E E E E E E F F F F F F E E F F F F F F F F F F F F F F F E E E F F F F F F F E
E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E F F E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
E E E E E E E E E E E E E E E E E E E E E E E D D E E E E E E E E E E E E E E E E E E E E E E E E E E E D D D D D D D D D D E E E E D E E E E E E E E
D E E E E E E E D D D D E E D C D F G G G G H H H H H H H H H H H H H G H H H H H H H H H H H H H H H H I I I I I H H H I I I H H H H H H
H H H H H H G G G G H H H G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G F F F G F G F F F F F F F G G G G G G G G G F F F
F F F F F G F F F F F G G G G G G G G F F F G G G G G G G F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F E E E E E E
E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E D D D D D D D E E E E E E E E E D D D D D D D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D D D E E E E E D D E E E E D D E D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D E E E
E E D E D D D D D D D D D D D D D D E D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B A B A A A A A A A A B B A A A A A B B B B B B B B B B B B B B A B A A A A A A B B B B B B B A A A A A A A B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B A B
B B B B B B B B B B B B B B A B B B B B B B B B B B B A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A

Матриця передування

result_matrix:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	815	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	9	651	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	66	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	2	178	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	14	233	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	11	178	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	12	141	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	16	225	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	13	136	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	7	236	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	6	64	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	16	216	6	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	7	125	6	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	7	194	6	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	7	143	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	27	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	34	1	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	55	5	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	44	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	46	1	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	79	7	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	93	1	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	44	5	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	90	3	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	374	8
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	198

Common Lisp

Лінгвістичний ланцюжок

char list: (Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Y Z Z
Z Z Z Z Z Z Z Z Z Y Y Z Z Z Z Y Z Z Z Z Z Z Z Z Z Z Z Z Z Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Z Z Z Z Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Z Z Z Z Z Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Z Y Y Y Y Y Y Z Y Y Y Y Y Y Y Z Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y X X Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y X X X Y X Y Y Y X X X X X X X X X X X X X
X X X X X X X X X X X X X X X W W W X X W X W W W X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X W X W W W X X X W W W W W W W W W W W
W W W W W W W W W W W W W W W W W W W W W W W W W V V V V V V V V W
W W W V V V V V V V V V V V V V V V V V V V V V V V V V V V V V U V
U U V V V V V V V V V V V V V V V V V V V U U U U U U U U U U U U U
T U U U U U U U V V V V V V V V V V V V V V V V V V U V V V V V V V
V V V V V V V V V U U U U U U U U U U U U U U U U U U U U U U U U U
U U V V V U U U V V V V V V V U U U U U U U U V U U U U U U U U U U
U U U U U U U U U U U U U U U U T T T T T T T T T T T T T T T T T T
T T T T T T T T T T T T T T T T T T T T T T T T T T T T T S S S S S
S S S S S S S S S S R R S S S S S S S S S S S R S S S S S S S S S S
S S S S S S S S S S R R R R R R R R R R R R R R R R R R R R R R R R
R R R R R R S S R R R R R R R S R S R R R R R R R R R R R R R R R R

R R R R Q Q Q Q Q R Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q
Q Q Q Q Q Q Q P P P P P P P P P P P P P P P P P P P P P P P P P P P P P P
P O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
O O O O O O O O O O O O O O O O O O O O O O O N N N N N N N N N N N N N N
N N N N N N N N N N N O N N N O O N N N O O O O O O O O O O O O O O O O
O O O O N N N N N O O O O O O O O O O O O O O O O O O O O O O O O O O O
O O N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N
N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N
N N N N N N N N N N N N N N N N N N N N M M M N N N N N N N N N M M
M N N N N N O O O O O O O O O O O O O O O N N N O O O O O O O O O O O
O O O O O O O O O O O O O O O O O O O O O O O O O O N N N N N N N N N
N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N N M
N N N N N N N N N N N N N N N N M M M M M M M M M M M M M M M M M M M M
M N M M N M M M M N N N N N N N N N M M M M M M M M M M M M M M M M M M
M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M
M M M M M M M M M M M M M M M M M M L M M M M M M M M M M M L M L L L L
L M M M M M M M L L L L L L M M M M M M M M M M M M M L M M M L
L L L L L L L L L L L L L L L L L M L L L L L L L L L L L L L L
L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L
L K K K K K K K L K K L L K K L K K L K K K K K K K K K K K K
K K K K L L L L L L L L L L L L L L L L L L L L L L L L K K K L
L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L
L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L L
L L L L L L L L L L L L L L L L L L L L L K K K K L K K K L L K
L L L L L L L L K K K K K K K L L L L L L L L L L K K K L L L K K
L L L K K K K L L K K K K L L K L K K K K K K K K K K K K J J J J
J J J J J I I J J J J J J J J J J J J J J K J K J K J J J J J K
K J J J J J J K K J J J J J J J J J J J J J J J J J J J J J J J
J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J
J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J J
J J J J J J J J J J J J J J J J I J J J J J J J J J J J J J J J
J J J J J J J J J J J J J J J J J I I J J J J J J J J J J J J J
J J J J J J J J J J J J I I I I I I I I I H H H H H H H H H H G G H
G G H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H H
H H H H H H H G G G H H G G G G G G G G H H H G G G G G G G G G H G G
H H G G G G G G G H H H H H H H H H H H H H H H H H H H H H H H H H I
I I I I I I I I I J J J J J J I J J J J J J J J J J J J J
I I I I J J J J J J J J J J J J J J J I I I I I I I I I I I I I I I
I I I I I H H H H H H H I I I I I I I I I I I I I I I I I I I
I I I I I H I I I H H H H H H H H H H H H H H H H H H H H H H H H H H
H H H H H H H H H H H H H I I I I I I I I I I I I I I I H I I I I
I I I I I I I I I I H H H H H H H H H H H H H H H H H H H H H H
H H H H H H H H I I I I I I I I I I I H H H I I I I I H H I I I I I
I I I I I I I I H H H I I I I H I I I I H H H H H H H H H H H H
H H H H H H H G G G G G G G H H H H G G G G G G G H H H H H G H H G
G G H G G G H G G G G G G G G G G G G G G G G G G G G G G G G G G G
G G G G G G G G F F F F F F F F F F G G G F F G G G F F F F F F F F
F G G G F F F G G G G G G G G G G F F F F F F F F F F F F F F F F
F F F F F F F F F F F F F F F F F F F E F F E E E E E E E E E E
E E F F F F F F F F F F F F F F F F F F F F F F F F F F F F F E
E E E E F E E F F E E E E E E E E E E E E E E E E E E E E E E E
E F F F F F F E E E E E E F F F F F E E F F F F F F F F F F F
E E E F F F F F F E E E E E E E E E E E E E E E E E E E E E E
E E E E E E E E E E E E E E F F E E E E E E E E E E E E E E E
E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
E E E D D E E E E E E E E E E E E E E E E E E E E E E D D D D D
D D D D D D E E E E D E E E E E E E D E E E E E E E D D D D E E D C
D F G G G G H H H H H H H H H H H H G H H H H H H H H H H H H
H H I I I I I H H H I I I H H H H H H H H H H H G G G G H H H G
G G G G G G G G G G G G G G G G G G G G G G G G G G G F F F G F G
F F F F F F F G G G G G G G G F F F F F F F F G F F F F F G G G G G
G G G F F F G G G G G G G F F F F F F F F F F F F F F F F F F F
F F F F F F F F F F F F F F F F F F F E E E E E E E E E E E E E
E E E E E E E E E E E E E E E E E E E E E E E D D D D D D D E E E
E E E E E D D D D D D D D D D D D D D D D D D D D D D D D D D D
D D D D D D D D D E E E E D D E E E D D E D D D D D D D D D D D D
D D D D D D D D D D D D D D D D D D D D D E E E E E D E D D D D D D

D D D D D D E D D D D D D D D D D D D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D C C C C C C C C C C C C C C C C C C C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C C C C C C C C C C C C C C B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B A B
A A A A A A A B B A A A A B B B B B B B B B B B B B A B A A A A A B
B B B B B A A A A A A B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B A B B B B B B B
B B B B B B B B A B B B B B B B B B B B A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A)

Матриця передування

```
815 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9 651 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 66 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 2 178 12 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 14 233 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 11 178 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 12 141 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 16 225 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 13 136 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 7 236 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 6 64 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 16 216 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 7 125 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 7 194 6 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 7 143 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 27 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 34 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 55 5 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 44 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 46 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 79 7 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 93 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 44 5 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 90 3 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 374 8 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 198 0 0
```

R

Лінгвістичний ланцюжок

```
[1] "char array:"
   [1] "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z"
  [33] "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z"
  [65] "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z"
  [97] "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z"
 [129] "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z"
 [161] "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Y" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Y" "Y" "Z" "Z" "Z" "Z" "Y" "Z" "Z" "Z" "Z" "Z" "Z"
 [193] "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Z" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Z" "Z" "Z" "Z" "Y" "Y" "Y"
 [225] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [257] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [289] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Z" "Z" "Z" "Z" "Z" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [321] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [353] "Y" "Y" "Y" "Z" "Y" "Y" "Y" "Y" "Y" "Y" "Z" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Z" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [385] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [417] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [449] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [481] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "X" "X" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [513] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [545] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y"
 [577] "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "Y" "X" "X" "X" "Y" "X" "Y" "Y" "Y" "X" "X" "X" "X" "X" "X" "X" "X" "X"
 [609] "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "W" "W" "W" "X" "X" "W" "X" "W" "W" "W" "X" "X" "X"
 [641] "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X"
 [673] "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "X" "W" "X" "W" "W" "W" "X" "X" "X" "W"
 [705] "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W" "W"
"W" "W"
 [737] "W" "W" "W" "V" "V" "V" "V" "V" "V" "V" "V" "W" "W" "W" "W" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V"
 [769] "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "U" "V" "U" "U" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V"
 [801] "V" "V" "V" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "T" "U" "U" "U" "U" "U" "U" "U" "V" "V" "V" "V" "V" "V" "V" "V" "V"
 [833] "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "U" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "V" "U" "U" "U" "U" "U" "U" "U"
 [865] "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "V" "V" "V" "U" "U" "U" "V" "V" "V" "V"
 [897] "V" "V" "V" "U" "U" "U" "U" "U" "U" "U" "U" "V" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U" "U"
 [929] "U" "U" "U" "U" "U" "U" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T"
 [961] "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "T" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S"
 [993] "S" "S" "S" "S" "R" "R" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "R" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S" "S"
[1025] "S" "S" "S" "S" "S" "S" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R"
```

[1057] "R" "R" "R" "R" "S" "S" "R" "R" "R" "R" "R" "R" "R" "S" "R" "S" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R" "R"
[1089] "R" "R" "R" "R" "Q" "Q" "Q" "Q" "Q" "R" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q"
[1121] "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "Q" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P" "P"
[1153] "P" "P" "P" "P" "P" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[1185] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "N" "N" "N" "N" "N" "N"
[1217] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "O" "N" "N" "N" "O" "O" "N" "N" "N" "O" "O" "O" "O"
[1249] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "N" "N" "N" "N" "N" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[1281] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[1313] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[1345] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[1377] "N" "N" "N" "N" "M" "M" "M" "N" "N" "N" "N" "N" "N" "N" "N" "N" "M" "M" "M" "N" "N" "N" "N" "N" "O" "O" "O" "O" "O" "O" "O" "O"
[1409] "O" "O" "O" "O" "O" "O" "N" "N" "N" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[1441] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[1473] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "M" "N" "N" "N" "N" "N" "N" "N"
[1505] "N" "N" "N" "N" "N" "N" "N" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "N" "M" "M" "N"
"M"
[1537] "M" "M" "M" "N" "N" "N" "N" "N" "N" "N" "N" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M"
"M"
[1569] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M"
"M" "M"
[1601] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "L" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "L" "M" "L" "L" "L"
"L"
[1633] "L" "M" "M" "M" "M" "M" "M" "M" "M" "L" "L" "L" "L" "L" "L" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "L" "M" "M"
[1665] "M" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "M" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
[1697] "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
[1729] "L" "L" "L" "L" "L" "L" "L" "K" "K" "K" "K" "K" "K" "K" "K" "L" "K" "K" "L" "L" "K" "K" "L" "K" "K" "L" "K" "K" "K" "K" "K"
[1761] "K" "K" "K" "K" "K" "K" "K" "K" "K" "K" "K" "K" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
[1793] "L" "L" "L" "L" "L" "L" "K" "K" "K" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
[1825] "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
[1857] "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
[1889] "L" "L" "L" "L" "L" "K" "K" "K" "K" "L" "K" "K" "K" "L" "L" "K" "L" "L" "L" "L" "L" "L" "L" "L" "K" "K" "K" "K" "K" "K" "K" "L"
[1921] "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "K" "K" "K" "L" "L" "L" "K" "K" "L" "L" "L" "K" "K" "K" "K" "L" "L" "K" "K" "K" "K" "L"
[1953] "L" "K" "L" "K" "K" "K" "K" "K" "K" "K" "K" "K" "K" "K" "K" "K" "J" "J" "J" "J" "J" "J" "J" "J" "J" "I" "I" "J" "J" "J" "J" "J"
[1985] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "K" "J" "K" "J" "K" "J" "J" "J" "J" "J" "K" "K" "J" "J" "J" "J" "J" "J" "J" "K" "K" "J"
[2017] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J"
[2049] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J"
[2081] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J"
[2113] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "I" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J"
[2145] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "I" "I" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J"
[2177] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "I" "I" "I" "I" "I" "I" "I" "I" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "G"
[2209] "G" "H" "G" "G" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H"
[2241] "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "G" "G" "G" "H" "H" "G" "G" "G" "G" "G" "G" "G" "H" "H" "H" "G" "G" "G" "G" "G" "G"
[2273] "G" "G" "G" "H" "G" "G" "H" "H" "G" "G" "G" "G" "G" "G" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H"
[2305] "H" "H" "H" "H" "H" "H" "H" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "J" "J" "J" "J" "J" "J" "J" "I" "J" "J" "J" "J" "J" "J"
[2337] "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "I" "I" "I" "I" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "J" "I"
[2369] "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "H" "H" "H" "H" "H" "H" "H" "H" "I" "I" "I" "I" "I" "I" "I"
[2401] "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "H" "I" "I" "I" "H" "H" "H" "H" "H" "H" "H" "H"
[2433] "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "I" "I" "I"
[2465] "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "H" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "H" "H"
[2497] "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "I" "I" "I" "I"
[2529] "I" "I" "I" "I" "I" "I" "I" "I" "H" "H" "H" "I" "I" "I" "I" "I" "H" "H" "H" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "I" "H"
[2561] "H" "H" "I" "I" "I" "I" "I" "H" "I" "I" "I" "I" "I" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H"
[2593] "G" "G" "G" "G" "G" "G" "G" "H" "H" "H" "H" "G" "G" "G" "G" "G" "G" "G" "H" "H" "H" "H" "H" "H" "H" "G" "H" "H" "G" "G" "G" "H" "G" "G" "G"
[2625] "H" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G"
[2657] "G" "G" "G" "G" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "G" "G" "G" "F" "F" "G" "G" "G" "F" "F" "F" "F" "F" "F" "F" "F" "F" "G"
[2689] "G" "G" "F" "F" "F" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F"
[2721] "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "E" "F" "F" "E" "E" "E" "E" "E" "E" "E"
[2753] "E" "E" "E" "E" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F"
[2785] "F" "F" "F" "E" "E" "E" "E" "E" "F" "E" "E" "F" "F" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E"
[2817] "E" "E" "E" "E" "E" "E" "E" "F" "F" "F" "F" "F" "F" "E" "E" "E" "E" "E" "E" "F" "F" "F" "F" "F" "F" "E" "E" "F" "F" "F" "F" "F"
[2849] "F" "F" "F" "F" "F" "F" "F" "F" "E" "E" "E" "F" "F" "F" "F" "F" "F" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E"
[2881] "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "F" "F" "E" "E" "E" "E" "E" "E"
[2913] "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E"
[2945] "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "D" "D" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E"
[2977] "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "E" "E" "E" "E" "D" "E" "E" "E" "E" "E"
[3009] "E" "E" "D" "E" "E" "E" "E" "E" "E" "E" "E" "D" "D" "D" "D" "E" "E" "D" "C" "D" "F" "G" "G" "G" "G" "H" "H" "H" "H" "H" "H" "H" "H"
[3041] "H" "H" "H" "H" "H" "G" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "I" "I" "I" "I" "I" "H" "H" "H" "I" "I"

[3073] "I" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "G" "G" "G" "G" "H" "H" "H" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G"
[3105] "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "G" "F" "F" "F" "G" "F" "G" "F" "F" "F" "F" "F" "F" "F" "G"
[3137] "G" "G" "G" "G" "G" "G" "G" "F" "F" "F" "F" "F" "F" "F" "F" "F" "G" "F" "F" "F" "F" "F" "F" "G" "G" "G" "G" "G" "G" "G" "G" "F" "F" "F"
[3169] "G" "G" "G" "G" "G" "G" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F"
[3201] "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E"
[3233] "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "E" "D" "D" "D" "D" "D" "D" "D" "E" "E" "E"
[3265] "E" "E" "E" "E" "E" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D"
[3297] "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "E" "E" "E" "E" "D" "D" "E" "E" "E" "D" "D" "E" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D"
[3329] "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "E" "E" "E" "E" "E" "D" "E"
[3361] "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "E" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D"
[3393] "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D"
[3425] "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "D" "C" "C" "C" "C" "C" "C" "C"
[3457] "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C"
[3489] "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "C" "B" "B" "B" "B"
[3521] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3553] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3585] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3617] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3649] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3681] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3713] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3745] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3777] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3809] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3841] "A" "B" "A" "A" "A" "A" "A" "A" "A" "B" "B" "A" "A" "A" "A" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "A" "B" "A" "A"
[3873] "A" "A" "A" "B" "B" "B" "B" "B" "B" "A" "A" "A" "A" "A" "A" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3905] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3937] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3969] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[4001] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[4033] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[4065] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[4097] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[4129] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[4161] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "A" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "A" "B"
[4193] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4225] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4257] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4289] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4321] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4353] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4385] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4417] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4449] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4481] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4513] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4545] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4577] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4609] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4641] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4673] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4705] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4737] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4769] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4801] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4833] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4865] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4897] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4929] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4961] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[4993] "A" "A" "A" "A" "A" "A" "A" "A"

Матриця передування

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	815	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	9	651	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	66	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	2	178	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	14	233	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	11	178	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	12	141	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	16	225	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	13	136	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	7	236	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	6	64	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	16	216	6	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	7	125	6	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	7	194	6	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	7	143	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	27	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	34	1	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	55	5	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	44	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	46	1	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	79	7	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	93	1	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	44	5	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	90	3	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	374	8
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	198

Prolog

Лінгвістичний ланцюжок

char array:

[z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,
z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,y,z,z,z,z,z,z,z,z,z,z,z,z,y,y,
z,z,z,z,y,z,z,z,z,z,z,z,z,z,z,z,z,z,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,z,z,z,z,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y
,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,z,z,z,z,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,z,y,y,y,y,y,y,z,y,y
,y,y,y,y,y,z,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y
,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,x,x,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,
y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,x,x,x,y,x,y,y,y,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,w,w,w,x,x,w,x,w,
w,w,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,w,x,w,w,w,x,x,x,w,w,w,w,w,w,w,w,w,w,w,w,w,
w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,w,v,v,v,v,v,v,v,v,v,v,w,w,w,w,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,u,v,u,u,v,v,v,v,v,v,v,v,v,v,v,v,v,v
,v,v,v,v,v,u,u,u,u,u,u,u,u,u,u,u,u,u,u,t,u,u,u,u,u,u,u,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,u,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,
u,v,v,v,u,u,u,v,v,v,v,v,v,v,v,u,u,u,u,u,u,u,u,u,v,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,u,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,s,s,s,s
,s,s,s,s,s,s,s,s,s,s,s,s,r,r,s,s,s,s,s,s,s,s,s,s,s,s,s,r,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,s,s,r,r,r,r,r,r,s,r,s,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,r,
q,q,q,q,q,r,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,q,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,p,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,
o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,o,n,n,n,o,o,n,n,n,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,
o,o,n,n,n,n,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,m,n,n,n,n,n,n,n,n
,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,m,m,m,n,n,n,n,n,n,n,n,n,n,m,m,m,n,n,n,n,n,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,n,n,n,o,o,o,o,o,o,o,o,o,o,o
,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,m,n,n,n,n,n,n,n,n,n,n,n,n,n,
n,n,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,n,m,m,n,m,m,m,m,m,n,n,n,n,n,n,n,n,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,
m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,l,m,m,m,m,m,m,m,m,m,m,m,m,m,l,m,l,l,l,l,l,m,m,m,m,m
,m,m,m,l,l,l,l,l,l,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,l,m,m,m,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,m,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,k,k
,k,k,k,k,k,k,l,k,k,l,l,k,k,l,k,k,l,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,k,k,k,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l
,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,l,k,k,k,k,l,k,k,k,l,l,k,l,l,l,l,l,l,l,l,k,k,k,k,k,k,k,l,l,l,l,l,l,l,l,l,l,l,l,k,k,k,l,l,l,k,k,l,l,l,k,k,k,k,l,l,k,k,k
,k,l,l,k,l,k,k,k,k,k,k,k,k,k,k,k,k,j,j,j,j,j,j,j,j,j,j,i,i,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,k,j,k,j,k,j,j,j,j,j,k,k,j,j,j,j,j,j,k,k,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,
j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,i,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,i,i,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j
,j,j,j,j,j,j,j,i,i,i,i,i,i,i,h,h,h,h,h,h,h,h,h,h,h,g,g,h,g,g,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,g,g,g,h,h,g,g,g,g,g,g,h,h,h,g,g,g
,g,g,g,g,g,g,h,g,g,h,h,g,g,g,g,g,g,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,i,i,i,i,i,i,i,i,i,i,i,i,j,j,j,j,j,j,j,j,i,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,i,i,i,i,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,j,i,i,i,i

,i,i,i,i,i,i,i,i,i,i,i,i,i,h,h,h,h,h,h,h,h,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,h,i,i,i,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,i,i,i,i,i,i,i,i,i
,i,i,i,i,i,i,h,i,i,i,i,i,i,i,i,i,i,i,i,i,i,i,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,i,i,i,i,i,i,i,i,i,i,h,h,h,i,i,i,i,i,h,h,i,i,i,i,i,i,i,i,i,i,i,i,i,h,h,h,i,i,i,i,h,i,i,i,i,h,h
,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,g,g,g,g,g,g,h,h,h,h,g,g,g,g,g,g,h,h,h,h,h,h,g,h,h,g,g,g,h,g,g,g,h,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g
,g,g,g,f,f,f,f,f,f,f,f,f,f,g,g,g,f,f,g,g,g,f,f,f,f,f,f,f,f,g,g,g,f,f,f,g,g,g,g,g,g,g,g,g,g,g,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,e,f,f,e,e,e,e,e,e,e,e,e,e,e
,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,e,e,e,e,e,f,e,e,f,f,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,f,f,f,f,f,f,e,e,e,e,e,e,f,f,f,f,f,f,e,e,f,f,f,f,f,f,f,f,f,f,f,f,f,
e,e,e,f,f,f,f,f,f,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,f,f,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,
e,e,e,e,e,e,e,e,e,e,e,e,e,d,d,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,d,d,d,d,d,d,d,d,d,d,d,e,e,e,e,d,e,e,e,e,e,e,e,d,e,e,e,e,e,e,e,d,d,d,d,e,e,d,c,d,f,g,g,g,g,h,h,h,h,h
,h,h,h,h,h,h,h,h,g,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,i,i,i,i,i,h,h,h,i,i,i,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,h,g,g,g,g,h,h,h,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,f,f
,f,g,f,g,f,f,f,f,f,f,f,g,g,g,g,g,g,g,g,f,f,f,f,f,f,f,f,g,f,f,f,f,f,g,g,g,g,g,g,g,g,f,f,f,g,g,g,g,g,g,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,f,e,e,e,e,e,e,e,
e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,d,d,d,d,d,d,d,e,e,e,e,e,e,e,e,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,e,e,e,
e,d,d,e,e,e,d,d,e,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,e,e,e,e,e,d,e,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,
d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c
,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,a,b,a,a,a,a,a,a,a,b,b,a,a,a,a,b,b,b,b,b,b,b,b,b,b,b,b,b,a,b,a,a,a,a,a,b,b,b,b,b,b,a,a,a,a,a,a,a,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,a,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,a,b,b,b,b,b,b,b,b,b,b,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a]

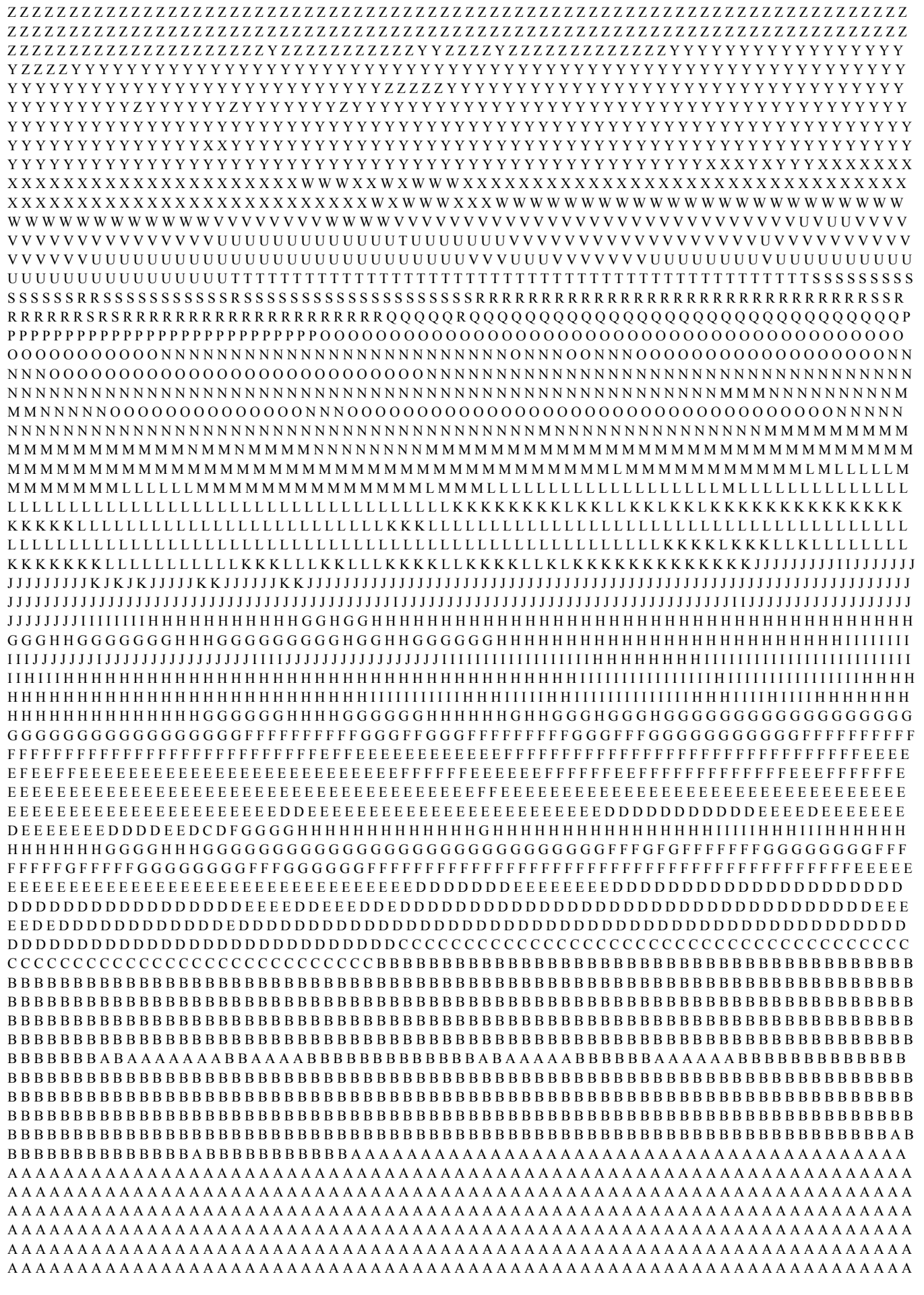
Матриця передування

```
Final matrix:
[815,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[9,651,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,1,66,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,2,180,11,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,13,233,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,11,178,11,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,12,141,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,16,225,12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,13,136,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,7,236,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,6,64,15,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,16,216,6,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,7,125,7,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,8,192,6,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,7,143,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,27,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,34,1,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,55,5,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,44,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,46,1,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,79,7,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,93,1,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,44,5,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,90,3,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,374,8]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,198]
```

CLIPS

Лінгвістичний ланцюжок

Character Array:



A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A

Матриця передування

```
Character Array:
Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	815	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	9	651	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	66	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	2	178	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	14	233	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	11	178	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	12	141	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	16	225	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	13	136	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	7	236	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	6	64	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	16	216	6	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	7	125	6	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	7	194	6	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	7	143	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	27	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	34	1	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	55	5	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	44	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	46	1	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	79	7	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	93	1	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	44	5	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	90	3	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	374	8
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	198

```
CLIPS>
```

Код

C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include <stdio.h>

/*
 * 1. user data -> DATA_ENTERING_FLAG 1
 * 2. file data -> DATA_ENTERING_FLAG 2
 * 3. random data -> DATA_ENTERING_FLAG 0
 */

#define ALPHABET_POWER 26
#define DATA_ENTERING_FLAG 2
#define DEBUGGING_FLAG 0
int size = 4;

char filename[] =
"/Users/Sayner/github_repos/multi-paradigm-programming/unit_work/mpp_get_data_script/files/f_data.t
xt";
double * array;
char alphabet[ALPHABET_POWER];
double * sorted_array;
double matrix_interval[ALPHABET_POWER][2];
char * char_array;
int result_matrix [ALPHABET_POWER + 1][ALPHABET_POWER + 1];


// values generating

void valuesByUser(){
    printf("Enter count of values: ");
    scanf("%d", &size);
    array = (double*)malloc(size * sizeof(double));
    for (int i = 0; i < size; i++){
        printf("a[%d] = ", i);
        scanf("%lf", &array[i]);
    }
}

void valuesByRandom(){
    array = (double*)malloc(size * sizeof(double));
    srand(time(NULL));
    for (int i = 0; i < size; i++) {
        array[i] = rand() % 201;
    }
}

int countLinesInFile(){
    int lineCount = 0;
    FILE *file;
    char c;

    file = fopen(filename, "r");

    if (file == NULL) {
        perror("file can't be opened\n");
        return -1;
    }
    while ((c = fgetc(file)) != EOF) {
        if (c == '\n') {
            lineCount++;
        }
    }

    fclose(file);
    return lineCount;
}

void valuesFromFile(){
```

```
    array = (double*)malloc(size * sizeof(double));
    char buffer[256];
    int count = 0;
    size = countLinesInFile();
    FILE* file = fopen(filename, "r");
    if (file == NULL) {
        perror("file can't be opened\n");
    }

    while (fgets(buffer, sizeof(buffer), file) != NULL) {
        if (count < size) {
            array[count] = atof(buffer);
            count++;
        } else {
            perror("maximum number of numbers exceeded \n");
            break;
        }
    }

    fclose(file);

    if (count == 0) {
        printf("file is empty or data not found.\n");
    }
}

// prepare data

void sortArray(){
    sorted_array = (double*)malloc(size * sizeof(double));
    for (int i = 0; i < size; i++)
        sorted_array[i] = array[i];
    for(int i = 1; i < size; i++){
        for(int k = i; k > 0 &&  sorted_array[k-1] > sorted_array[k]; k--){
            double tmp = sorted_array[k-1];
            sorted_array[k-1] = sorted_array[k];
            sorted_array[k] = tmp;
        }
    }
}

void setAlphabet(){
    for (int i = 0; i < ALPHABET_POWER; ++i) {
        alphabet[i] = 'A' + i;
    }
}

// cut to intervals

void func_to_debugging_intervals_cutting(double a, double pA, double b, double pB){
    printf("a : %f\n", a);
    printf("pA : %f\n", pA);
    printf("pB : %f\n", pB);
    printf("b : %f\n", b);
    printf("_____________________________________\n");
}

double releу_distribution(double x, double sigma) {
    if (x < 0) return 0;
    return (1 - exp(-0.5 * pow(x / sigma, 2)));
}

double inverse_releу_distribution(double P, double sigma) {
    if (P < 0 || P > 1) return -1;
    if (P == 1) return -1;
    return sigma * sqrt(-2 * log(1 - P));
}

void cutToIntervals() {
    double summ = 0;
    for (int i = 0; i < size; i++) {
        summ += sorted_array[i] * sorted_array[i];
    }

    double sigma = sqrt(summ / (2 * size));
    printf("sigma: %f\n\n", sigma);
    double interval[2];
```

```
    double a = sorted_array[0];
    for (int i = 0; i < ALPHABET_POWER; i++) {
        interval[0] = a;
        double pA = reley_distribution(interval[0], sigma);
        double pB = 1.0/ALPHABET_POWER + pA;
        double b = inverse_reley_distribution(pB, sigma);
        interval[1] = b;
        if(DEBUGGING_FLAG == 1)
            func_to_debugging_intervals_cutting(interval[0], pA, interval[1], pB);
        a = interval[1];
        matrix_interval[i][0] = interval[0];
        matrix_interval[i][1] = interval[1];
    }
    matrix_interval[ALPHABET_POWER - 1][1] = sorted_array[size - 1];
    if(matrix_interval[ALPHABET_POWER - 1][0] > matrix_interval[ALPHABET_POWER - 1][1]){
        double interval_width = (matrix_interval[ALPHABET_POWER - 1][1] -
matrix_interval[ALPHABET_POWER - 2][0])/2;
        matrix_interval[ALPHABET_POWER - 2][1] = matrix_interval[ALPHABET_POWER - 2][0] +
interval_width;
        matrix_interval[ALPHABET_POWER - 1][0] = matrix_interval[ALPHABET_POWER - 2][1];
    }

    printf("intervals:\n");
    for (int i = 0; i < ALPHABET_POWER; i++) {
        printf("%c: [%f, %f]\n", alphabet[i], matrix_interval[i][0], matrix_interval[i][1]);
    }
    printf("\n");
}

void toCharArray(){
    char_array = (char*)malloc(size * sizeof(char));
    for(int i = 0; i < size; i++){
        for(int j = 0; j < ALPHABET_POWER; j++){
            if(array[i] >= matrix_interval[j][0] &&  array[i] <= matrix_interval[j][1]){
                char_array[i] = alphabet[j];
                break;
            }
        }
    }
}

// making result matrix

int findIndex(char a, char * charArray){
    int index = -1;
    for(int i = 0; i < ALPHABET_POWER; i++){
        if(a == charArray[i])
            index = i;
    }
    return index;
}

void makeResultMatrix(){
    for(int i = 0; i < size; i++){
        if(i + 1 < size){
            int current_index = findIndex(char_array[i], alphabet);
            int next_index = findIndex(char_array[i + 1], alphabet);
            if(current_index != -1 && next_index != -1){
                result_matrix[current_index][next_index] = result_matrix[current_index][next_index]
+ 1;
            }
        }
    }
}

// prints

void printResultMatrix() {
    printf("result_matrix:\n");
    printf("%4c ", ' ');
    for(int i=0; i < ALPHABET_POWER; i++){
        printf("%4c ", alphabet[i]);
    }
    printf("\n");
    for (int i = 0; i < ALPHABET_POWER; i++) {
        for (int j = 0; j < ALPHABET_POWER; j++) {
            if(j == 0){
```

```
                printf("%4c ", alphabet[i]);
            }
            printf("%4d ", result_matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

void printIntArray(int * _array, int size){
    for (int i = 0; i < size; i++)
        printf("%d ", _array[i]);
    printf("\n");
    printf("\n");
}

void printDoubleArray(double * _array, int size){
    for (int i = 0; i < size; i++)
        printf("%.2f ", _array[i]);
    printf("\n");
    printf("\n");
}

void printCharArray(char *charArray, int size){
    for (int i = 0; i < size; ++i)
        printf("%c ", charArray[i]);
    printf("\n");
    printf("\n");
}


int main (){

    clock_t start, end;
    double cpu_time_used;
    start = clock();

    int CORRECT_DATA_FLAG = 0;
    if(DATA_ENTERING_FLAG == 1){
        valuesByUser();
        if (size > 0 && ALPHABET_POWER > 0)
            CORRECT_DATA_FLAG = 1;
    }else if(DATA_ENTERING_FLAG == 2){
        valuesFromFile();
        if(ALPHABET_POWER == 26 && size > 0)
            CORRECT_DATA_FLAG = 1;
    }else{
        valuesByRandom();
        if(ALPHABET_POWER > 0 && size > 0)
            CORRECT_DATA_FLAG = 1;
    }
    if(CORRECT_DATA_FLAG == 1){
        printf("default array:\n");
        printDoubleArray(array, size);

        sortArray();
        printf("sorted array:\n");
        printDoubleArray(sorted_array, size);

        setAlphabet();
        printf("alphabet:\n");
        printCharArray(alphabet, ALPHABET_POWER);

        cutToIntervals();

        toCharArray();
        printf("char array:\n");
        printCharArray(char_array, size);

        makeResultMatrix();
        printResultMatrix();
    }else
        printf("incorrect data.");

    end = clock();
    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
    printf("%f secs\n", cpu_time_used);
```

```
    return 0;
}
```

Common Lisp

```
(defun values-by-random (size &optional (result '()))
  (if (zerop size)
      result
      (values-by-random (1- size) (cons (random 100) result))))

(defun values-from-file (file)
  (with-open-file (stream file :direction :input)
    (loop for line = (read-line stream nil)
          while line
          collect (multiple-value-bind (value success)
                      (ignore-errors (read-from-string line))
                    (if success value nil)))))

(defun values-by-file (file)
  (let ((path-to-file (merge-pathnames file *load-truename*)))
    (values-by-file-helper (values-from-file path-to-file))))

(defun values-by-file-helper (data &optional (result '()))
  (if (endp data)
      result
      (values-by-file-helper (cdr data) (cons (car data) result))))

(defun sort-list (array)
  (sort (subseq array 0 (length array)) '<))

(defun set-alphabet (alphabet-power &optional (index 0) (alphabet '()))
  (if (< index alphabet-power)
      (set-alphabet alphabet-power (1+ index) (cons (string (code-char (+ (char-code #\A) index)))
alphabet))
      (reverse alphabet)))

(defun reley-distribution (x sigma)
  (if (< x 0)
      0
      (- 1 (exp (*  -0.5 (expt (/ x sigma) 2))))))

(defun inverse-reley-distribution (P sigma)
  (if (or (< P 0) (> P 1))
      -1
      (if (= P 1)
          -1
          (* sigma (sqrt (* -2 (log (- 1 P))))))))

(defun cut-to-intervals (sorted-array alphabet-power)
  "cut list to intervals based on rayleigh distribution"
  (let* ((size (length sorted-array))
         (summ (reduce #'+ (mapcar (lambda (x) (* x x)) sorted-array)))
         (sigma (sqrt (/ summ (* 2 size))))
         (matrix-interval '())
         (buffer (elt sorted-array 0)))
    ; (format t "sigma: ~a~%" sigma)
    (dotimes (i alphabet-power)
      (let* ((pA (reley-distribution buffer sigma))
             (b (inverse-reley-distribution (+ (/ 1.0 alphabet-power) pA) sigma)))
        ; (format t "Buffer: ~a, pA: ~a, b: ~a~%" buffer pA b)
        (push (list buffer b) matrix-interval)
        (setf buffer b)))
    (setf (cadar matrix-interval) (elt sorted-array (1- size)))
    (reverse matrix-interval)))

(defun to-char-list (start-list intervals alphabet)
  "convert list of numbers into a list of characters based on intervals"
  (let ((char-list (make-list (length start-list))))
    (dotimes (i (length start-list))
      (let ((current-number (nth i start-list)))
        (dotimes (j (length alphabet))
          (let ((a (nth 0 (nth j intervals)))
                (b (nth 1 (nth j intervals)))
                (current-char (nth j alphabet)))
            (when (and (<= a current-number b))
              (setf (nth i char-list) current-char))))))
    char-list))

(defun find-index (element lst &optional (index 0))
```

```
  (cond ((null lst) nil)
        ((eql element (car lst)) index)
        (t (find-index element (cdr lst) (+ index 1)))))

(defun make-result-matrix (char-list alphabet)
  (let* ((alphabet-power (length alphabet))
         (result-matrix (make-array (list alphabet-power alphabet-power) :initial-element 0)))
    (loop for i below (1- (length char-list))
          for current-char = (nth i char-list)
          for next-char = (if (< (1+ i) (length char-list)) (nth (1+ i) char-list) nil)
          do (let* ((current-index (find-index current-char alphabet))
                    (next-index (find-index next-char alphabet)))
               (when (and current-index next-index)
                 (incf (aref result-matrix next-index current-index)))))
    result-matrix))

(defun print-matrix (matrix)
  (loop for i below (array-dimension matrix 0)
        do (loop for j below (array-dimension matrix 1)
                 do (format t "~a " (aref matrix i j)))
           (format t "~%")))

(defun main (size alphabet-power)
  (let ((start-time (get-internal-real-time)))
    (let* ((start-list (values-by-file "f_data.txt"))
           (alphabet (set-alphabet alphabet-power))
           (sorted-list (sort-list start-list))
           (intervals (cut-to-intervals sorted-list alphabet-power))
           (char-list (to-char-list start-list intervals alphabet))
           (result-matrix (make-result-matrix char-list alphabet)))
      (format t "default list: ~a~%" start-list)
      (format t "sorted list: ~a~%" sorted-list)
      (format t "alphabet: ~a~%" alphabet)
      (dolist (interval intervals)
        (format t "interval: [~f, ~f]~%" (first interval) (second interval)))
      (format t "char list: ~a~%" char-list)
      (print-matrix result-matrix)
      (let* ((end-time (get-internal-real-time))
             (elapsed-time (- end-time start-time)))
        (format t "~a secs~%" (/ elapsed-time 1000000.0))))))

(main 4 26)
```

R

```
releyDistribution <- function(x, sigma) {
  if (x < 0) return (0);
  return (1 - exp(-0.5 * (x / sigma)^2));
}

inverseReleyDistribution <- function(p, sigma) {
  if (p < 0 || p > 1) return (-1);
  if (p == 1) return (-1);
  return (sigma * sqrt(-2 * log(1 - p)));
}

cutToIntervals <- function(arr, alphabetPower) {
  summ <- sum(arr^2)
  size <- length(arr)
  sigma <- sqrt(summ / (2 * size))
  intervalWidth <- size / alphabetPower
  interval <- matrix(NA, nrow = alphabetPower, ncol = 2)
  buffer <- arr[1]

  for (i in 1:alphabetPower) {
    a <- releyDistribution(buffer, sigma)
    b <- inverseReleyDistribution(1.0 / alphabetPower + a, sigma)

    interval[i, 1] <- buffer
    interval[i, 2] <- b
    buffer <- b
  }
  interval[alphabetPower, 2] <- arr[size]
  if(interval[alphabetPower, 1] > interval[alphabetPower, 2]){
    intervalWidth <- (interval[alphabetPower, 2] - interval[alphabetPower - 1, 1]) / 2
    interval[alphabetPower - 1, 2] <- interval[alphabetPower - 1, 1] + intervalWidth
    interval[alphabetPower, 1] <- interval[alphabetPower - 1, 2]
  }
  return (interval)
}

toCharArray <- function(defaultArray, matrixInterval, alphabet) {
  cArray <- vector(mode = "character", length = (length(defaultArray)))
  for (i in 1:length(defaultArray)) {
    for (j in 1:nrow(matrixInterval)) {
      if (defaultArray[i] >= matrixInterval[j, 1] && defaultArray[i] <= matrixInterval[j, 2]) {
        cArray[i] <- alphabet[j]
        break
      }
    }
  }
  return(cArray)
}

makeResultMatrix <- function(cArray, alphabet) {
  alphabetPower <- length(alphabet)
  resultMatrix <- matrix(0, nrow = alphabetPower, ncol = alphabetPower, dimnames =
list(alphabet, alphabet))

  for (i in 1:length(cArray)) {
    currentIndex <- match(cArray[i], alphabet)
    nextIndex <- match(cArray[i + 1], alphabet)

    if (!is.na(currentIndex) && !is.na(nextIndex)) {
      resultMatrix[currentIndex, nextIndex] <- resultMatrix[currentIndex, nextIndex] + 1
    }
  }

  return(resultMatrix)
}

SIZE <- 1
ALPHABER_POWER <- 26

setwd("/Users/Sayner/github_repos/multi-paradigm-programming/unit_work/mpp_get_data_script/files")
options(max.print = 5000)
startTime <- Sys.time()

if(SIZE > 0 && ALPHABER_POWER > 0){
```

```
  #  array <- sample(1:200, SIZE, replace = TRUE)
  array <- scan("f_data.txt")
  print("default array:")
  print(array)

  alphabet <- LETTERS[1:ALPHABER_POWER]

  print("alphabet:")
  print(alphabet)

  sorted_array <- sort(array)

  print("sorted array:")
  print(sorted_array)

  intervals <- cutToIntervals(sorted_array, ALPHABER_POWER)
  print("intervals:")
  print(intervals)

  char_array <- toCharArray(array, intervals, alphabet)
  print("char array:")
  print(char_array)

  print(makeResultMatrix(char_array, alphabet))
}else{
  print("alphabet power or array size <= 0, calculating impossible.")
}

end_time <- Sys.time()
execution_time <- end_time - start_time
print(execution_time)
```

Prolog

```
:- use_module(library(clpfd)).
:- use_module(library(lists)).

read_and_print_numbers(FileName, NumericSeries) :-
    read_file(FileName, NumericSeries),
    % print_numbers(NumericSeries),
    % writeln(' '),
    list_length(NumericSeries, Length),
    format('list size: ~w~n', [Length]).

read_file(FileName, Numbers) :-
    open(FileName, read, Stream),
    read_lines(Stream, Numbers),
    close(Stream).

read_lines(Stream, []) :-
    at_end_of_stream(Stream), !.
read_lines(Stream, [Number|Numbers]) :-
    \+ at_end_of_stream(Stream),
    read_line(Stream, Number),
    read_lines(Stream, Numbers).

read_line(Stream, Number) :-
    read_line_to_string(Stream, Line),
    atom_number(Line, Number).

print_numbers([]).
print_numbers([Number|Numbers]) :-
    format('~2f ', [Number]),
    print_numbers(Numbers).

list_length([], 0).
list_length([_|Tail], Length) :-
    list_length(Tail, TailLength),
    Length is TailLength + 1.

merge_sort([], []).
merge_sort([X], [X]).
merge_sort(List, Sorted) :-
    List = [_, _|_],
    split(List, L1, L2),
    merge_sort(L1, SortedL1),
    merge_sort(L2, SortedL2),
    merge(SortedL1, SortedL2, Sorted).

split([], [], []).
split([X], [X], []).
split([X,Y|T], [X|L1], [Y|L2]) :-
    split(T, L1, L2).

merge([], L, L).
merge(L, [], L).
merge([X|T1], [Y|T2], [X|T]) :-
    X =< Y,
    merge(T1, [Y|T2], T).
merge([X|T1], [Y|T2], [Y|T]) :-
    X > Y,
    merge([X|T1], T2, T).

truncate_decimal(Number, Precision, Truncated) :-
    (   var(Number), var(Precision)
    ->  throw(error(instantiation_error, truncate_decimal/3))
    ;   nonvar(Number), nonvar(Precision)
    ->  Truncated is round(Number * 10^Precision) / 10^Precision
    ;   throw(error(instantiation_error, truncate_decimal/3))
    ).

releу_distribution(X, Sigma, Result) :-
    (   var(X), var(Sigma)
    ->  throw(error(instantiation_error, releу_distribution/3))
    ;   nonvar(X), nonvar(Sigma)
    ->  (   X < 0
        ->  Result = 0
        ;   Intermediate is 1 - exp(-0.5 * (X / Sigma) ** 2),
```

```
            truncate_decimal(Intermediate, 6, Truncated),
            Result = Truncated
        )
    ;   throw(error(instantiation_error, releY_distribution/3))
    ).

inverse_releY_distribution(P, Sigma, Result) :-
    (   var(P), var(Sigma)
    ->  throw(error(instantiation_error, inverse_releY_distribution/3))
    ;   nonvar(P), nonvar(Sigma)
    ->  (   P < 0
        ->  Result = -1
        ;   P > 1
        ->  Result = -1
        ;   P == 1
        ->  Result = -1
        ;   Intermediate is Sigma * sqrt(-2 * log(1 - P)),
            truncate_decimal(Intermediate, 6, Truncated),
            Result = Truncated
        )
    ;   throw(error(instantiation_error, inverse_releY_distribution/3))
    ).

compute_intervals(_, _, _, AlphabetPower, []) :-
    AlphabetPower =< 0.

compute_intervals(SortedArray, Sigma, Buffer, AlphabetPower, [[Interval0, Interval1]|Intervals]) :-
    AlphabetPower > 0,
    Interval0 = Buffer,
    releY_distribution(Interval0, Sigma, Pa),
    IntermediateAlphabetPower is 1.0 / 26,
    Intermediate is IntermediateAlphabetPower + Pa,
    truncate_decimal(Intermediate, 6, Pb),
    (
        Pb > 1 ->
        last(SortedArray, LastElement),
        Interval1 = LastElement
        ;
        inverse_releY_distribution(Pb, Sigma, Interval1)
    ),
    % write('Interval0: '), write(Interval0), nl,
    % write('Pa: '), write(Pa), nl,
    % write('Pb: '), write(Pb), nl,
    % write('Interval1: '), write(Interval1), nl,
    NextPower is AlphabetPower - 1,
    compute_intervals(SortedArray, Sigma, Interval1, NextPower, Intervals).

compute_intervals_main(SortedArray, Sigma, AlphabetPower, MatrixInterval) :-
    (   SortedArray = [First|_]
    ->  compute_intervals(SortedArray, Sigma, First, AlphabetPower, MatrixInterval)
    ;   MatrixInterval = []
    ).

sum_of_squares([], 0).
sum_of_squares([H|T], Sum) :-
    sum_of_squares(T, RestSum),
    Sum is H * H + RestSum.

compute_sigma(SortedArray, Sigma) :-
    length(SortedArray, Size),
    sum_of_squares(SortedArray, Sum),
    Sigma is sqrt(Sum / (2 * Size)).

to_char_array(Array, _CharArray, MatrixInterval, Alphabet) :-
    length(Array, Size),
    to_char_array(0, Size, Array, _CharArray, MatrixInterval, Alphabet).

to_char_array(Size, Size, _, [], _, _).
to_char_array(Index, Size, Array, [Char|Rest], MatrixInterval, Alphabet) :-
    Index < Size,
    nth0(Index, Array, Element),
    map_interval(Element, Char, MatrixInterval, Alphabet),
    NextIndex is Index + 1,
    to_char_array(NextIndex, Size, Array, Rest, MatrixInterval, Alphabet).

map_interval(Element, Char, [[Low, High]|_], [LowChar|_]) :-
    Element >= Low,
```

```
    Element =< High,
    Char = LowChar.
map_interval(Element, Char, [_|Rest], [_|RestAlpha]) :-
    map_interval(Element, Char, Rest, RestAlpha).

print_intervals([], _).
print_intervals([[Interval0, Interval1]|Intervals], [Letter|Letters]) :-
    format('~w : [~f, ~f]~n', [Letter, Interval0, Interval1]),
    print_intervals(Intervals, Letters).

generate_alphabet(A) :-
    generate_alphabet(97, A).

generate_alphabet(123, []) :- !.
generate_alphabet(Curr, [Letter|Rest]) :-
    char_code(Letter, Curr),
    Next is Curr + 1,
    generate_alphabet(Next, Rest).

find_index(Element, List, Index) :-
    nth0(Index, List, Element).

make_result_matrix(CharArray, Alphabet, ResultMatrix, FinalResultMatrix) :-
    length(CharArray, Size),
    make_result_matrix_helper(CharArray, Alphabet, ResultMatrix, FinalResultMatrix, Size, 0).

make_result_matrix_helper(_, _, ResultMatrix, ResultMatrix, Size, Index) :-
    Index >= Size - 1, !.

make_result_matrix_helper(CharArray, Alphabet, ResultMatrix, FinalResultMatrix, Size, _Index) :-
    _NextIndex is _Index + 1,
    nth0(_Index, CharArray, CurrentChar),
    nth0(_NextIndex, CharArray, NextChar),
    find_index(CurrentChar, Alphabet, CurrentIndex),
    find_index(NextChar, Alphabet, NextIndex),
    increment_matrix(ResultMatrix, CurrentIndex, NextIndex, TempResultMatrix),
    make_result_matrix_helper(CharArray, Alphabet, TempResultMatrix, FinalResultMatrix, Size,
_NextIndex).

increment_matrix(ResultMatrix, CurrentIndex, NextIndex, NewResultMatrix) :-
    nth0(CurrentIndex, ResultMatrix, Row),
    nth0(NextIndex, Row, Value),
    NewValue is Value + 1,
    replace(Row, NextIndex, NewValue, NewRow),
    replace(ResultMatrix, CurrentIndex, NewRow, NewResultMatrix).

replace([_|T], 0, X, [X|T]).
replace([H|T], I, X, [H|R]) :-
    I > 0,
    I1 is I - 1,
    replace(T, I1, X, R).

initialize_result_matrix(Alphabet, ResultMatrix) :-
    length(Alphabet, Len),
    length(Row, Len),
    maplist(=(0), Row),
    length(ResultMatrix, Len),
    maplist(=(Row), ResultMatrix).

print_matrix([]).
print_matrix([Row|Rest]) :-
    writeln(Row),
    print_matrix(Rest).

main() :-
    statistics(runtime, _),
    % writeln('array:'),
    read_and_print_numbers('data.txt', NumericSeries),
    merge_sort(NumericSeries, SortedArray),
    compute_sigma(SortedArray, Sigma),
    generate_alphabet(A),
    write('alphabet: '), nl,
    writeln(A),
    write('sigma: '), write(Sigma), nl,
    % writeln('sorted array:'),
    % print_numbers(SortedArray),
    % writeln(' '),
```

```
    compute_intervals_main(SortedArray, Sigma, 26, Intervals),
    print_intervals(Intervals, A),
    to_char_array(NumericSeries, CharArray, Intervals, A),
    writeln('char array:'),
    writeln(CharArray),
    initialize_result_matrix(A, ResultMatrix),
    make_result_matrix(CharArray, A, ResultMatrix, FinalResultMatrix),
    writeln('Final matrix:'),
    print_matrix(FinalResultMatrix),
    statistics(runtime, [_, Time]),
    format('~f secs~n', [Time/1000]).


:- initialization(main()).
```

CLIPS

```
(deftemplate value
   (slot number (type FLOAT)))

(deftemplate interval
   (slot start (type FLOAT))
   (slot end (type FLOAT)))

(deftemplate alphabet
   (slot letter (type SYMBOL)))

(deftemplate transition
   (slot from (type SYMBOL))
   (slot to (type SYMBOL))
   (slot count (type INTEGER)))

(defrule read-values
   =>
   (open
"/Users/Sayner/github_repos/multi-paradigm-programming/unit_work/mpp_get_data_script/files/f_data.t
xt" data "r")
   (assert (values))
   (bind ?data (readline data))
   (while (neq ?data EOF)
       (assert (value (number (float ?data))))
       (bind ?data (readline data)))
   (close data)
)

(defrule print-initial-array
   ?values <- (values)
   ?value <- (value (number ?number))
   =>
   (printout t "Initial Array:" crlf)
   (printout t "   " ?number crlf)
   (printout t crlf)
)

(defrule sort-array
   ?values <- (values)
   =>
   (bind ?array (find-all-facts ((?f value)) TRUE))
   (bind ?sorted-array (sort ?array))
   (assert (sorted-array ?sorted-array))
)

(defrule print-sorted-array
   ?sorted-array <- (sorted-array $?array)
   =>
   (printout t "Sorted Array:" crlf)
   (printout t "   " (implode$ ?array) crlf)
   (printout t crlf)
)

(deffunction set-alphabet (?size)
   (bind ?start-char 65) ; ASCII code for 'A'
   (bind ?end-char (+ ?start-char (- ?size 1)))
   (loop-for-decimal (?ascii ?start-char ?end-char)
      (assert (alphabet (letter (char ?ascii))))
   )
)

(defrule set-alphabet
   =>
   (set-alphabet 26) ;
)

(defrule print-alphabet
   ?alphabets <- (alphabet $?letters)
   =>
   (printout t "Alphabet:" crlf)
   (printout t "   " (implode$ ?letters) crlf)
   (printout t crlf)
)
```

```
(deffunction releу-distribution (?x ?sigma)
  (if (< ?x 0) then
    0
    else
    (- 1 (exp (* -0.5 (pow (/ ?x ?sigma) 2))))))

(deffunction inverse-releу-distribution (?P ?sigma)
  (if (or (< ?P 0) (> ?P 1)) then
    -1
    else
    (if (eq ?P 1) then
      -1
      else
      (* ?sigma (sqrt (* -2 (log (- 1 ?P))))))))

(deffunction cut-to-intervals ()
  (bind ?summ 0)
  (loop-for-count (?i ?size)
    (bind ?summ (+ ?summ (* (nth$ ?i ?sorted_array) (nth$ ?i ?sorted_array)))))
  (bind ?sigma (sqrt (/ ?summ (* 2 ?size))))
  (printout t "sigma: " ?sigma crlf crlf)
  (bind ?interval (create$ 0 0))
  (bind ?a (nth$ 1 ?sorted_array))
  (loop-for-count (?i ?ALPHABET_POWER)
    (bind ?interval (create$ ?a))
    (bind ?pA (releу-distribution (nth$ 1 ?interval) ?sigma))
    (bind ?pB (+ (/ 1.0 ?ALPHABET_POWER) ?pA))
    (bind ?b (inverse-releу-distribution ?pB ?sigma))
    (bind ?interval (create$ ?a ?b))
    (bind ?a (nth$ 2 ?interval))
    (bind ?matrix_interval (create$ (nth$ 1 ?interval) (nth$ 2 ?interval)))
    (bind ?matrix_intervals (replace$ ?matrix_intervals ?i ?matrix_interval)))
  (bind ?i (- ?ALPHABET_POWER 1))
  (bind ?last_interval (nth$ ?i ?matrix_intervals))
  (if (> (nth$ 1 ?last_interval) (nth$ 2 ?last_interval)) then
    (bind ?interval_width (/ (- (nth$ 2 ?last_interval) (nth$ 1 (nth$ (- ?i 1) ?matrix_intervals)))
2))
    (bind ?last_interval (replace$ ?last_interval 2 (+ (nth$ 1 ?last_interval) ?interval_width)))
    (bind ?last_interval (replace$ ?last_interval 1 (nth$ 2 (nth$ (- ?i 1) ?matrix_intervals))))
    (bind ?matrix_intervals (replace$ ?matrix_intervals ?i ?last_interval)))
  (printout t "intervals:" crlf)
  (loop-for-count (?i ?ALPHABET_POWER)
    (printout t (str-cat (nth$ ?i ?alphabet) ": ["
                         (nth$ 1 (nth$ ?i ?matrix_intervals)) ", "
                         (nth$ 2 (nth$ ?i ?matrix_intervals)) "]\n")))
  (printout t crlf))


(defrule to-char-array
   ?intervals <- (interval $?)
   ?sorted-array <- (sorted-array ?array)
   =>
   (bind ?char-array "")
   (foreach ?number ?array
      (loop-for-count (?i 0 (length$ ?intervals))
         (if (<= ?number (fact-slot-value (nth$ ?i ?intervals) end))
            then
            (bind ?char-array (str-cat ?char-array (symbol-to-string (deftemplate-slot-value
(fact-name (nth$ ?i ?intervals)) start))))
            break
         )
      )
   )
   (assert (char-array ?char-array))
)

(defrule print-char-array
   ?char-array <- (char-array ?array)
   =>
   (printout t "Character Array:" crlf)
   (printout t "   " ?array crlf)
   (printout t crlf)
)

(defrule make-result-matrix
   ?char-array <- (char-array ?array)
   =>
```

```
   (bind ?size (length$ ?char-array))
   (bind ?result-matrix (create$ (do-for-all-facts ((?f alphabet)) (fact-slot-value ?f letter))))
   (loop-for-count (?i 0 ?size)
      (bind ?from (nth$ ?i ?char-array))
      (bind ?to (nth$ (+ ?i 1) ?char-array))
      (if (neq ?to "")
         then
         (bind ?from-index (find-index-in-fact-slot-value ?from ?result-matrix))
         (bind ?to-index (find-index-in-fact-slot-value ?to ?result-matrix))
         (bind ?count (fact-slot-value (nth$ ?from-index ?result-matrix) ?to))
         (modify-item ?result-matrix ?from-index (create$ ?to (+ ?count 1)))
      )
   )
   (assert (result-matrix ?result-matrix))
)

(defrule print-result-matrix
   ?result-matrix <- (result-matrix $?matrix)
   =>
   (printout t "Result Matrix:" crlf)
   (printout t "   " (implode$ (do-for-all-facts ((?f alphabet)) (fact-slot-value ?f letter)))
crlf)
   (foreach ?row ?matrix
      (printout t (nth$ 0 ?row) " " (implode$ (rest$ ?row)) crlf)
   )
   (printout t crlf)
)
```

[github](#) - скрипти для пре обробки даних та розрахунку за формулою суми різниці елементів.

	Mi	Mф	MR	Мл	Мп
Mi	0	0	0	0	0
Mф	0	0	0	0	0
MR	0	0	0	0	0
Мл	0	0	0	0	0
Мп	0	0	0	0	0