МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Министерство образования и науки Республики Башкортостан
МКУ Отдел образования Администрации муниципального района
Абзелиловский район Республики Башкортостан
МБОУ СОШ д.Казмашево

РАССМОТРЕНО	СОГЛАСОВАНО	УТВЕРЖДЕНО
на заседании ШМО	Заместитель директора по УВР	Директор МБОУ СОШ д.Казмашево
Зайнуллина Ф.Ф.	Аюпова Р.И	Атауллин З.В
Протокол №1 от «30» августа 2023 г.	Протокол №1 от «31» августа 2023 г.	Протокол №1/20 от «01» сентября 2023 г.

РАБОЧАЯ ПРОГРАММА

(ID 2229190)

учебного курса «Математика»

для обучающихся 5-6 классов

за 2023- 2024 учебный год

Составитель : Зайнуллина Фаузия Фаизовна
Учитель физики и математики

Казмашево 2023

Introduction

Welcome to the Python Tutorial. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

- The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.
- The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.
- This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.
- For a description of standard objects and modules, see the Python Standard Library. The Python Language Reference gives a more formal definition of the language. To write extensions in C or C++, read Extending and Embedding the Python Interpreter and Python/C API Reference Manual. There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in the Python Standard Library.

The Glossary is also worth going through.

Whetting Your Appetite

If you ever have to write a complex program in a low-level language, you will probably feel that you are spending all your time figuring out the details. In Python, you may not have written the program yet, but you will know what you want to do, and you will be able to express your ideas clearly and concisely. Python will not get in your way.

Python is extensible: if you know how to program in C, it is easy to add a new function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form. Once you are hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

By the way, the language is named after the BBC show "Monty Python's Flying Circus" and has nothing to do with reptiles. Making references to Monty Python skits in documentation is not only allowed, it is encouraged!

Using the Python Interpreter

Invoking the Interpreter

The Python interpreter is usually installed as /usr/local/bin/python3.13 on those machines where it is available; putting /usr/local/bin in your Unix shell's search path makes it possible to start it by typing the command:

```
python3.13
```

to the shell. Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.13 command will be available. If you have the py.exe launcher installed, you can use the py command. See Python Launcher for Windows for more details.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that does not work, you can exit the interpreter by typing the following command: quit().

The interpreter's line-editing features usually aren't very sophisticated. On Unix, whoever installed the interpreter may have enabled support for the GNU readline library, which adds more elaborate interactive editing and history features. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell's -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

All command line options are described in Command line and environment.

Argument Passing

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys.argv[0] is an empty string. When the script name is given as '-' (meaning standard input), sys.argv[0] is set to '-'. When -c command is used, sys.argv[0] is set to '-c'. When -m module is used, sys.argv[0] is set to the full name of the located module. Options found after -c command or -m module are not consumed by the Python interpreter's option processing but left in sys.argv for the command or module to handle.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
python3.13
Python 3.13 (default, April 4 2024, 09:25:04)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>> the_world_is_flat = True
>>> if the_world_is_flat:
...     print("Be careful not to fall off!")
...
Be careful not to fall off!
```

The Interpreter and Its Environment

Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

```
# -*- coding: encoding -*-
```

where encoding is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

```
# -*- coding: cp1252 -*-
```

One exception to the first line rule is when the source code starts with a UNIX "shebang" line. In this case, the encoding declaration should be added as the second line of the file. For example:

```
#!/usr/bin/env python3
# -*- coding: cp1252 -*-
```

An Informal Introduction to Python

In the following examples, input and output are distinguished by the presence or absence of prompts (>>> and ...): to repeat the example, you must type everything after the prompt, when the prompt appears; lines that do not begin with a prompt are output from the interpreter. Note that a secondary prompt on a line by itself in an example means you must type a blank line; this is used to end a multi-line command.

Many of the examples in this manual, even those entered at the interactive prompt, include comments. Comments in Python start with the hash character, #, and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal. A hash character within a string literal is just a hash character. Since comments are to clarify code and are not interpreted by Python, they may be omitted when typing in examples.

Some examples:

```
# this is the first comment
spam = 1  # and this is the second comment
          # ... and now a third!
text = "# This is not a comment because it's inside quotes."
```

Using Python as a Calculator

Let's try some simple Python commands. Start the interpreter and wait for the primary prompt, >>>. (It shouldn't take long.)

Numbers

The interpreter acts as a simple calculator: you can type an expression at it and it will write the value. Expression syntax is straightforward: the operators +, -, * and / work just like in most other languages (for example, Pascal or C); parentheses (()) can be used for grouping. For example:

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
```

The integer numbers (e.g. 2, 4, 20) have type int, the ones with a fractional part (e.g. 5.0, 1.6) have type float. We will see more about numeric types later in the tutorial.

Division (/) always returns a float. To do floor division and get an integer result you can use the // operator; to calculate the remainder you can use %:

```
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>>
>>> 17 // 3  # floor division discards the fractional part
5
>>> 17 % 3  # the % operator returns the remainder of the division
2
>>> 5 * 3 + 2  # result * divisor + remainder
17
```

With Python, it is possible to use the ** operator to calculate powers:

```
>>> 5 ** 2  # 5 squared
25
>>> 2 ** 7  # 2 to the power of 7
128
```

The equal sign (=) is used to assign a value to a variable. Afterwards, no result is displayed before the next interactive prompt:

```
>>> width = 20
>>> height = 5 * 9
>>> width * height
900
```

If a variable is not "defined" (assigned a value), trying to use it will give you an error:

```
>>> n  # try to access an undefined variable
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'n' is not defined
```

There is full support for floating point; operators with mixed type operands convert the integer operand to floating point:

```
>>> 4 * 3.75 - 1
14.0
```

In interactive mode, the last printed expression is assigned to the variable _. This means that when you are using Python as a desk calculator, it is somewhat easier to continue calculations, for example:

```
>>> tax = 12.5 / 100
>>> price = 100.50
>>> price * tax
12.5625
>>> price + _
113.0625
>>> round(_, 2)
113.06
```

This variable should be treated as read-only by the user. Don't explicitly assign a value to it — you would create an independent local variable with the same name masking the built-in variable with its magic behavior.

In addition to int and float, Python supports other types of numbers, such as Decimal and Fraction. Python also has built-in support for complex numbers, and uses the j or J suffix to indicate the imaginary part (e.g. 3+5j).

Text

Python can manipulate text (represented by type str, so-called "strings") as well as numbers. This includes characters "!", words "rabbit", names "Paris", sentences "Got your back.", etc. "Yay! :)". They can be enclosed in single quotes ('...') or double quotes ("...") with the same result.

```
>>> 'spam eggs'  # single quotes
'spam eggs'
>>> "Paris rabbit got your back :)! Yay!"  # double quotes
'Paris rabbit got your back :)! Yay!'
>>> '1975'  # digits and numerals enclosed in quotes are also strings
'1975'
```

To quote a quote, we need to "escape" it, by preceding it with \. Alternatively, we can use the other type of quotation marks:

```
>>> 'doesn\'t'  # use \' to escape the single quote...
"doesn't"
>>> "doesn't"  # ...or use double quotes instead
"doesn't"
>>> '"Yes," they said.'
'"Yes," they said.'
>>> "\"Yes,\" they said."
'"Yes," they said.'
>>> '"Isn\'t," they said.'
'"Isn\'t," they said.'
```

In the interactive interpreter, the output string is enclosed in quotes and special characters are escaped with backslashes. While this might sometimes look different from the input (the enclosing quotes could change), the two strings are equivalent. The string is enclosed in double quotes if the string contains a single quote and no double quotes, otherwise it is enclosed in single quotes. The print() function produces a more readable output, by omitting the enclosing quotes and by printing escaped and special characters:

```
>>> s = 'First line.\nSecond line.'  # \n means newline
>>> s  # without print(), special characters are included in the string
'First line.\nSecond line.'
>>> print(s)  # with print(), special characters are interpreted, so \n produces new line
First line.
Second line.
```

If you don't want characters prefaced by \ to be interpreted as special characters, you can use raw strings by adding an r before the first quote:

```
>>> print('C:\some\name')  # here \n means newline!
C:\some
ame
>>> print(r'C:\some\name')  # note the r before the quote
C:\some\name
```

There is one subtle aspect to raw strings: a raw string may not end in an odd number of \ characters; see the FAQ entry for more information and workarounds.

String literals can span multiple lines. One way is using triple-quotes: """...""" or '''...'''. End of lines are automatically included in the string, but it's possible to prevent this by adding a \ at the end of the line. The following example:

```
print("""\
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
""")
```

produces the following output (note that the initial newline is not included):

```
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
```

Strings can be concatenated (glued together) with the + operator, and repeated with *:

```
>>> # 3 times 'un', followed by 'ium'
>>> 3 * 'un' + 'ium'
'unununium'
```

Two or more string literals (i.e. the ones enclosed between quotes) next to each other are automatically concatenated.

```
>>> 'Py' 'thon'
'Python'
```

This feature is particularly useful when you want to break long strings:

```
>>> text = ('Put several strings within parentheses '
...         'to have them joined together.')
>>> text
'Put several strings within parentheses to have them joined together.'
```

This only works with two literals though, not with variables or expressions:

```
>>> prefix = 'Py'
>>> prefix 'thon'  # can't concatenate a variable and a string literal
  File "<stdin>", line 1
    prefix 'thon'
           ^^^^^^
SyntaxError: invalid syntax
>>> ('un' * 3) 'ium'
  File "<stdin>", line 1
    ('un' * 3) 'ium'
               ^^^^^
SyntaxError: invalid syntax
```

If you want to concatenate variables or a variable and a literal, use +:

```
>>> prefix + 'thon'
'Python'
```

This feature can also be useful when you want to break long strings:

```
>>> text = ('Put several strings within parentheses '
...         'to have them joined together.')
>>> text
'Put several strings within parentheses to have them joined together.'
```

Strings can be indexed (subscripted), with the first character having index 0. There is no separate character type; a character is simply a string of size one:

```
>>> word = 'Python'
>>> word[0]  # character in position 0
'P'
>>> word[5]  # character in position 5
'n'
```

Indices may also be negative numbers, to start counting from the right:

```
>>> word[-1]  # last character
'n'
>>> word[-2]  # second-last character
'o'
>>> word[-6]
'P'
```

Note that since -0 is the same as 0, negative indices start from -1.

In addition to indexing, slicing is also supported. While indexing is used to obtain individual characters, slicing allows you to obtain substring:

```
>>> word[0:2]  # characters from position 0 (included) to 2 (excluded)
'Py'
>>> word[2:5]  # characters from position 2 (included) to 5 (excluded)
'tho'
```

Note how the start is always included, and the end always excluded. This makes sure that s[:i] + s[i:] is always equal to s:

```
>>> word[:2] + word[2:]
'Python'
>>> word[:4] + word[4:]
'Python'
```

Slice indices have useful defaults; an omitted first index defaults to zero, an omitted second index defaults to the size of the string being sliced.

```
>>> word[:2]  # character from the beginning to position 2 (excluded)
'Py'
>>> word[4:]  # characters from position 4 (included) to the end
'on'
>>> word[-2:]  # characters from the second-last (included) to the end
'on'
```

One way to remember how slices work is to think of the indices as pointing between characters, with the left edge of the first character numbered 0. Then the right edge of the last character of a string of n characters has index n, for example:

```
 +---+---+---+---+---+---+
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

The first row of numbers gives the position of the indices 0...6 in the string; the second row gives the corresponding negative indices. The slice from i to j consists of all characters between the edges labeled i and j, respectively.

For non-negative indices, the length of a slice is the difference of the indices, if both are within bounds. For example, the length of word[1:3] is 2.

Attempting to use an index that is too large will result in an error:

```
>>> word[42]  # the word only has 6 characters
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

However, out of range slice indexes are handled gracefully when used for slicing:

```
>>> word[4:42]
'on'
>>> word[42:]
''
```

Python strings cannot be changed — they are immutable. Therefore, assigning to an indexed position in the string results in an error:

```
>>> word[0] = 'J'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> word[2:] = 'py'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

If you need a different string, you should create a new one:

```
>>> 'J' + word[1:]
'Jython'
>>> word[:2] + 'py'
'Pypy'
```

The built-in function len() returns the length of a string:

```
>>> s = 'supercalifragilisticexpialidocious'
>>> len(s)
34
```

Lists

Python knows a number of compound data types, used to group together other values. The most versatile is the list, which can be written as a list of comma-separated values (items) between square brackets. Lists might contain items of different types, but usually the items all have the same type.

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
```

Like strings (and all other built-in sequence types), lists can be indexed and sliced:

```
>>> squares[0]  # indexing returns the item
1
>>> squares[-1]
25
>>> squares[-3:]  # slicing returns a new list
[9, 16, 25]
```

All slice operations return a new list containing the requested elements. This means that the following slice returns a new (shallow) copy of the list:

```
>>> squares[:]
[1, 4, 9, 16, 25]
```

Lists also support operations like concatenation:

```
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Unlike strings, which are immutable, lists are a mutable type, i.e. it is possible to change their content:

```
>>> cubes = [1, 8, 27, 65, 125]  # something's wrong here
>>> 4 ** 3  # the cube of 4 is 64, not 65!
64
>>> cubes[3] = 64  # replace the wrong value
>>> cubes
[1, 8, 27, 64, 125]
```

You can also add new items at the end of the list, by using the append() method (we will see more about methods later):

```
>>> cubes.append(216)  # add the cube of 6
>>> cubes.append(7 ** 3)  # and the cube of 7
>>> cubes
[1, 8, 27, 64, 125, 216, 343]
```

Assignment to slices is also possible, and this can even change the size of the list or clear it entirely:

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>> letters
['a', 'b', 'f', 'g']
>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>> letters
[]
```

The built-in function len() also applies to lists:

```
>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
4
```

It is possible to nest lists (create lists containing other lists), for example:

```
>>> a = ['a', 'b', 'c']
>>> n = [1, 2, 3]
>>> x = [a, n]
>>> x
[['a', 'b', 'c'], [1, 2, 3]]
>>> x[0]
['a', 'b', 'c']
>>> x[0][1]
'b'
```

First Steps Towards Programming

Of course, we can use Python for more complicated tasks than adding two and two together. For instance, we can write an initial sub-sequence of the Fibonacci series as follows:

```
>>> # Fibonacci series:
>>> # the sum of two elements defines the next
>>> a, b = 0, 1
>>> while a < 10:
...     print(a)
...     a, b = b, a+b
...
0
1
1
2
3
5
8
```

This example introduces several new features.

- The first line contains a multiple assignment: the variables a and b simultaneously get the new values 0 and 1. On the last line this is used again, demonstrating that the expressions on the right-hand side are all evaluated first before any of the assignments take place. The right-hand side expressions are evaluated from the left to the right.
- The while loop executes as long as the condition (here: a < 10) remains true. In Python, like in C, any non-zero integer value is true; zero is false. The condition may also be a string or list value, in fact any sequence; anything with a non-zero length is true, empty sequences are false. The test used in the example is a simple comparison. The standard comparison operators are written the same as in C: < (less than), > (greater than), == (equal to), <= (less than or equal to), >= (greater than or equal to) and != (not equal to).
- The body of the loop is indented: indentation is Python's way of grouping statements. At the interactive prompt, you have to type a tab or space(s) for each indented line. In practice you will prepare more complicated input for Python with a text editor; most text editors have an auto-indent facility. When a compound statement is entered interactively, it must be followed by a blank line to indicate completion (since the parser cannot guess when you have typed the last line). Note that each line within a basic block must be indented by the same amount.
- The print() function writes the value of the argument(s) it is given. It differs from just writing the expression you want to write (as we did earlier in the calculator examples) in the way it handles multiple arguments, floating point quantities, and strings. Strings are printed without quotes, and a space is inserted between items, so you can format things nicely, like this:

```
>>> i = 256 * 256
>>> print('The value of i is', i)
The value of i is 65536
```

The keyword argument end can be used to avoid the newline after the output, or end the output with a different string:

```
>>> a, b = 0, 1
>>> while a < 1000:
...     print(a, end=',')
...     a, b = b, a+b
...
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,
```

More Control Flow Tools

Besides the while statement just introduced, Python uses the usual flow control statements known from other languages, with some twists.

if Statements

Perhaps the most well-known statement type is the if statement. For example:

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

There can be zero or more elif parts, and the else part is optional. The keyword 'elif' is short for 'else if', and is useful to avoid excessive indentation. An if ... elif ... elif ... sequence is a substitute for the switch or case statements found in other languages.

for Statements

The for statement in Python differs a bit from what you may be used to in C or Pascal. Rather than always iterating over an arithmetic progression of numbers (like in Pascal), or giving the user the ability to define both the iteration step and halting condition (as in C), Python's for statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence. For example:

```
>>> # Measure some strings:
>>> words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

Code that modifies a collection while iterating over that same collection can be tricky to get right. Instead, it is usually more straightforward to loop over a copy of the collection or to create a new collection:

```
# Create a sample collection
users = {'Hans': 'active', 'Éléonore': 'inactive', '景太郎': 'active'}

# Strategy:  Iterate over a copy
for user, status in users.copy().items():
    if status == 'inactive':
        del users[user]

# Strategy:  Create a new collection
active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status
```

The range() Function

If you do need to iterate over a sequence of numbers, the built-in function range() comes in handy. It generates arithmetic progressions:

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
```

The given end point is never part of the generated sequence; range(10) generates 10 values, the legal indices for items of a sequence of length 10. It is possible to let the range start at another number, or to specify a different increment (even negative; sometimes this is called the 'step'):

```
range(5, 10)
   5 6 7 8 9

range(0, 10, 3)
   0 3 6 9

range(-10, -100, -30)
  -10 -40 -70
```

To iterate over the indices of a sequence, you can combine range() and len() as follows:

```
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print(i, a[i])
...
0 Mary
1 had
2 a
3 little
4 lamb
```

In most such cases, however, it is convenient to use the enumerate() function, see Looping Techniques.

break and continue Statements, and else Clauses on Loops

The break statement, like in C, breaks out of the innermost enclosing for or while loop.

Loop statements may have an else clause; it is executed when the loop terminates through exhaustion of the iterable (with for) or when the condition becomes false (with while), but not when the loop is terminated by a break statement. This is exemplified by the following loop, which searches for prime numbers:

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...             print(n, 'equals', x, '*', n//x)
...             break
...     else:
...         # loop fell through without finding a factor
...         print(n, 'is a prime number')
...
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

(Yes, this is the correct code. Look closely: the else clause belongs to the for loop, not the if statement.)

When used with a loop, the else clause has more in common with the else clause of a try statement than it does with that of if statements: a try statement's else clause runs when no exception occurs, and a loop's else clause runs when no break occurs. For more on the try statement and exceptions, see Handling Exceptions.

The continue statement, also borrowed from C, continues with the next iteration of the loop:

```
>>> for num in range(2, 10):
...     if num % 2 == 0:
...         print("Found an even number", num)
...         continue
...     print("Found a number", num)
Found an even number 2
Found a number 3
Found an even number 4
Found a number 5
Found an even number 6
Found a number 7
Found an even number 8
Found a number 9
```

pass Statements

The pass statement does nothing. It can be used when a statement is required syntactically but the program requires no action. For example:

```
>>> while True:
...     pass  # Busy-wait for keyboard interrupt (Ctrl+C)
...
```

This is commonly used for creating minimal classes:

```
>>> class MyEmptyClass:
...     pass
...
```

Another place pass can be used is as a place-holder for a function or conditional body when you are working on new code, allowing you to keep thinking at a more abstract level. The pass is silently ignored:

```
>>> def initlog(*args):
...     pass   # Remember to implement this!
...
```

match Statements

A match statement takes an expression and compares its value to successive patterns given as one or more case blocks. This is superficially similar to a switch statement in C, Java or JavaScript (and many other languages), but it's more similar to pattern matching in languages like Rust or Haskell. Only the first pattern that matches gets executed and it can also extract components (sequence elements or object attributes) from the value into variables.

The simplest form compares a subject value against one or more literals:

```
def http_error(status):
    match status:
        case 400:
            return "Bad request"
        case 404:
            return "Not found"
        case 418:
            return "I'm a teapot"
        case _:
            return "Something's wrong with the internet"
```

Note the last block: the "variable name" _ acts as a wildcard and never fails to match. If no case matches, none of the branches is executed.

You can combine several literals in a single pattern using | ("or"):

```
case 401 | 403 | 404:
    return "Not allowed"
```

Patterns can look like unpacking assignments, and can be used to bind variables:

```
# point is an (x, y) tuple
match point:
    case (0, 0):
        print("Origin")
    case (0, y):
        print(f"Y={y}")
    case (x, 0):
        print(f"X={x}")
    case (x, y):
        print(f"X={x}, Y={y}")
    case _:
        raise ValueError("Not a point")
```

Study that one carefully! The first two cases have two points, and can be thought of as matching a point and its coordinates. But the third case is a single point, whose coordinates are needed, so the coordinates are used to bind the variables x and y. The fourth case matches any two-item tuple and binds the two values to x and y.

If you are using classes to structure your data you can use the class name followed by an argument list resembling a constructor, but with the ability to capture attributes into variables:

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

def where_is(point):
    match point:
        case Point(x=0, y=0):
            print("Origin")
        case Point(x=0, y=y):
            print(f"Y={y}")
        case Point(x=x, y=0):
            print(f"X={x}")
        case Point():
            print("Somewhere else")
        case _:
            print("Not a point")
```

You can use positional parameters with some builtin classes that provide an ordering for their attributes (e.g. dataclasses). You can also define a specific position for attributes in patterns by setting the __match_args__ special attribute in your classes. If it's set to ("x", "y"), the following patterns are all equivalent (and all bind the y attribute to the var variable):

```
Point(1, var)
Point(1, y=var)
Point(x=1, y=var)
Point(y=var, x=1)
```

A recommended way to read patterns is to look at them as an extended form of what you would put on the left of an assignment, to understand which variables would be set to what. Only the standalone names (like var above) are assigned to by a match statement. Dotted names (like foo.bar), attribute names (the x= and y= above) or class names (recognized by the "(...)" next to them like Point above) are never assigned to.

Patterns can be arbitrarily nested. For example, if we have a short list of points, with __match_args__ added, we could match it like this:

```
class Point:
    __match_args__ = ('x', 'y')
    def __init__(self, x, y):
        self.x = x
        self.y = y

match points:
    case []:
        print("No points")
    case [Point(0, 0)]:
        print("The origin")
    case [Point(x, y)]:
        print(f"Single point {x}, {y}")
    case [Point(0, y1), Point(0, y2)]:
        print(f"Two on the Y axis at {y1}, {y2}")
    case _:
        print("Something else")
```

We can add an if clause to a pattern, known as a "guard". If the guard is false, match goes on to try the next case block. Note that value capture happens before the guard is evaluated:

```
match point:
    case Point(x, y) if x == y:
        print("Y=X at", x)
    case Point(x, y):
        print("Not on the diagonal")
```

Several other key features of this statement:

- Like unpacking assignments, tuple and list patterns have exactly the same meaning and actually match arbitrary sequences. An important exception is that they don't match iterators or strings.
- Sequence patterns support extended unpacking: [x, y, *rest] and (x, y, *rest) work similar to unpacking assignments. The name after * may also be _, so (x, y, *_) matches a sequence of at least two items without binding the remaining items.
- Mapping patterns: {"bandwidth": b, "latency": l} captures the "bandwidth" and "latency" values from a dict. Unlike sequence patterns, extra keys are ignored. An unpacking like **rest is also supported. (But **_ would be redundant, so is not allowed.)
- Subpatterns may be captured using the as keyword:

```
case (Point(x1, y1), Point(x2, y2) as p2): ...
```

will capture the second element of the input as p2 (as long as the input is a sequence of two points)

- Most literals are compared by equality, however the singletons True, False and None are compared by identity.
- Named constants may be used in patterns. These named constants must be dotted names to prevent the constant from being interpreted as a capture variable:

```
from enum import Enum
class Color(Enum):
    RED = 'red'
    GREEN = 'green'
    BLUE = 'blue'

color = Color(input("Enter your choice of 'red', 'blue' or 'green': "))

match color:
    case Color.RED:
        print("I see red!")
    case Color.GREEN:
        print("Grass is green")
    case Color.BLUE:
        print("I'm feeling the blue")
```

For a more detailed explanation and additional examples, you can look into PEP 636 written in tutorial format.

Defining Functions

We can create a function that writes the Fibonacci series to an arbitrary boundary:

```
>>> def fib(n):    # write Fibonacci series up to n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while a < n:
...         print(a, end=' ')
...         a, b = b, a+b
...     print()
...
>>> # Now call the function we just defined:
>>> fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

The keyword def introduces a function definition. It must be followed by the function name and the parenthesized list of formal parameters. The statements that form the body of the function start at the next line, and must be indented.

The first statement of the function body can optionally be a string literal; this string literal is the function's documentation string, or docstring. There are tools which use docstrings to automatically produce online or printed documentation, or to let the user interactively browse through code; it's good practice to include docstrings in code that you write, so make it a habit.

The execution of a function introduces a new symbol table used for the local variables of the function. More precisely, all variable assignments in a function store the value in the local symbol table; whereas variable references first look in the local symbol table, then in the local symbol tables of enclosing functions, then in the global symbol table, and finally in the table of built-in names. Thus, global variables and variables of enclosing functions cannot be directly assigned a value within a function (unless, for global variables, named in a global statement, or, for variables of enclosing functions, named in a nonlocal statement), although they may be referenced.

The actual parameters (arguments) to a function call are introduced in the local symbol table of the called function when it is called; thus, arguments are passed using call by value (where the value is always an object reference, not the value of the object). When a function calls another function, or calls itself recursively, a new local symbol table is created for that call.

A function definition associates the function name with the function object in the current symbol table. The interpreter recognizes the object pointed to by that name as a user-defined function. Other names can also point to that same function object and can also be used to access the function:

```
>>> fib
<function fib at 10042ed0>
>>> f = fib
>>> f(100)
0 1 1 2 3 5 8 13 21 34 55 89
```

Coming from other languages, you might object that fib is not a function but a procedure since it doesn't return a value. In fact, even functions without a return statement do return a value, albeit a rather boring one. This value is called None (it's a built-in name). Writing the value None is normally suppressed by the interpreter if it would be the only value written. You can see it if you really want to using print():

```
>>> fib(0)
>>> print(fib(0))
None
```

It is simple to write a function that returns a list of the numbers of the Fibonacci series, instead of printing it:

```
>>> def fib2(n):  # return Fibonacci series up to n
...     """Return a list containing the Fibonacci series up to n."""
...     result = []
...     a, b = 0, 1
...     while a < n:
...         result.append(a)    # see below
...         a, b = b, a+b
...     return result
...
>>> f100 = fib2(100)    # call it
>>> f100                # write the result
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

This example, as usual, demonstrates some new Python features:

- The return statement returns with a value from a function. return without an expression argument returns None. Falling off the end of a function also returns None.
- The statement result.append(a) calls a method of the list object result. A method is a function that 'belongs' to an object and is named obj.methodname, where obj is some object (this may be an expression), and methodname is the name of a method that is defined by the object's type. Different types define different methods. Methods of different types may have the same name without causing ambiguity. (It is possible to define your own object types and methods, using classes, see Classes) The method append() shown in the example is defined for list objects; it adds a new element at the end of the list. In this example it is equivalent to result = result + [a], but more efficient.

More on Defining Functions

It is also possible to define functions with a variable number of arguments. There are three forms, which can be combined.

Default Argument Values

The most useful form is to specify a default value for one or more arguments. This creates a function that can be called with fewer arguments than it is defined to allow. For example:

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)
```

This function can be called in several ways:

- giving only the mandatory argument: ask_ok('Do you really want to quit?')
- giving one of the optional arguments: ask_ok('OK to overwrite the file?', 2)
- or even giving all arguments: ask_ok('OK to overwrite the file?', 2, 'Come on, only yes or no!')

This example also introduces the in keyword. This tests whether or not a sequence contains a certain value.

The default values are evaluated at the point of function definition in the defining scope, so that

```
i = 5

def f(arg=i):
    print(arg)

i = 6
f()
```

will print 5.

Important warning: The default value is evaluated only once. This makes a difference when the default is a mutable object such as a list, dictionary, or instances of most classes. For example, the following function accumulates the arguments passed to it on subsequent calls:

```
def f(a, L=[]):
    L.append(a)
    return L

print(f(1))
print(f(2))
print(f(3))
```

This will print

```
[1]
[1, 2]
[1, 2, 3]
```

If you don't want the default to be shared between subsequent calls, you can write the function like this instead:

```
def f(a, L=None):
    if L is None:
        L = []
    L.append(a)
    return L
```

Keyword Arguments

Functions can also be called using keyword arguments of the form kwarg=value. For instance, the following function:

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print("-- This parrot wouldn't", action, end=' ')
    print("if you put", voltage, "volts through it.")
    print("-- Lovely plumage, the", type)
    print("-- It's", state, "!")
```

accepts one required argument (voltage) and three optional arguments (state, action, and type). This function can be called in any of the following ways:

```
parrot(1000)
parrot(voltage=1000)
parrot(voltage=1000000, action='VOOOOOM')
parrot(action='VOOOOOM', voltage=1000000)
parrot('a million', 'bereft of life', 'jump')
parrot('a thousand', state='pushing up the daisies')
```

but all the following calls would be invalid:

```
parrot()                     # required argument missing
parrot(voltage=5.0, 'dead')  # non-keyword argument after a keyword argument
parrot(110, voltage=220)     # duplicate value for the same argument
parrot(actor='John Cleese')  # unknown keyword argument
```

In a function call, keyword arguments must follow positional arguments. All the keyword arguments passed must match one of the arguments accepted by the function (e.g. actor is not a valid argument for the parrot function), and their order is not important. This also includes non-optional arguments (e.g. parrot(voltage=1000) is valid too). No argument may receive a value more than once. Here's an example that fails due to this restriction:

```
>>> def function(a):
...     pass
...
>>> function(0, a=0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: function() got multiple values for argument 'a'
```

When a final formal parameter of the form **name is present, it receives a dictionary (see Mapping Types — dict) containing all keyword arguments except for those corresponding to a formal parameter. This may be combined with a formal parameter of the form *name (described in the next subsection) which receives a tuple containing the positional arguments beyond the formal parameter list. (*name must occur before **name.) For example, if we define a function like this:

```
def cheeseshop(kind, *arguments, **keywords):
    print("-- Do you have any", kind, "?")
    print("-- I'm sorry, we're all out of", kind)
    for arg in arguments:
        print(arg)
    print("-" * 40)
    for kw in keywords:
        print(kw, ":", keywords[kw])
```

It could be called like this:

```
cheeseshop("Limburger", "It's very runny, sir.",
           "It's really very, VERY runny, sir.",
           shopkeeper="Michael Palin",
           client="John Cleese",
           sketch="Cheese Shop Sketch")
```

and of course it would print:

```
-- Do you have any Limburger ?
-- I'm sorry, we're all out of Limburger
It's very runny, sir.
It's really very, VERY runny, sir.
----------------------------------------
shopkeeper : Michael Palin
client : John Cleese
sketch : Cheese Shop Sketch
```

Note that the order in which the keyword arguments are printed is guaranteed to match the order in which they were provided in the function call.

Special parameters

By default, arguments may be passed to a Python function either by position or explicitly by keyword. For readability and performance, it makes sense to restrict the manner in which arguments can be passed so that a developer need only look at the function definition to determine if items are passed by position, by position or keyword, or by keyword.

A function definition may look like:

```
def f(pos1, pos2, /, pos_or_kwd, *, kwd1, kwd2):
      -----------    ----------     ----------
        |             |                  |
        |        Positional or keyword   |
        |                                - Keyword only
         -- Positional only
```

where / and * are optional. If used, these symbols indicate the kind of parameter by how the arguments may be passed to the function: positional-only, positional-or-keyword, and keyword-only. Keyword parameters are also referred to as named parameters.

тЃtsdlzuj dzdz j tsвlzuO ћ cdзcfrў dzO lztetsэdg dzOucOlz dzsGts tsвћ j.Gts tsвteOLtsэOdzcw, fmffmsj dзOscLcteZк lsfrў c teOfhj cteWк lsfrў.

ч tsGdzOfmgzs lzuj.вdzsdзz тdzOdz э 5ї 6 ЂdzOfmfmOЋ cLlzuOj lsfrў cdzsj.Gteceteтsэ0dzdz? тtej.нdзj ls «Ѓ Oslj.dзOscЂO», ЂtslststcГ? э Ђdz uOj ls Oecfw dзj lscuj.fmЂcf? dзOslj.tecfOdz c dzOcdzнdzк cj.tsdзj lstecfк , O lsOЂx j тtetstfj.нj.эlscuj.fmЂcj fmj.нj.dzcfw cL Odzcj.вtcГ , ч dg.dзj dzs dzsGcfЂc c dzOucOdzOtstfcfmOslj.dz dzs? fmsOscfmlscfЂc.

l O cLlzuj.dzcfj lzuj.вdzsGts ЂlztefmO «Ѓ Oslj.dзOscЂO» tslэtsнclsfrў 340 ucOmtsэ: э 5 ЂdzOfmfmj ї 170 ucOmtsэ (5 ucOmtsэ э dg.нj.dz), э 6 ЂdzOfmfmj ї 170 ucOmtsэ (5 ucOmtsэ э dg.нj.dz).

[illegible]

5 [illegible]

[illegible]

[illegible] 0. [illegible] ([illegible]) [illegible].

[illegible].

[illegible].

[illegible].

[illegible] ([illegible]) [illegible] ([illegible]) [illegible].

[illegible].

[illegible] 2, 5, 10, 3, 9. [illegible]

[illegible].

[illegible] ([illegible]) [illegible].

[illegible]

[illegible].

[illegible].

[illegible].

ϕ tɛɑʃʷ dʒjlsʃʊjfŋʕɑʃj Hj?fŋjsɔʃʷ fŋ Hj?fŋʸlsʃʊɖʑ dʒʃ HtɛtsBʷdʒʃ. ʃ ʕtɛlzɢɑʃjɑʃj Hjfŋʸlsʃʊɖʑ ⊐ HtɛtsBj?.

t j^h j dɑʃj lsjʕfŋjstsɔ^r ⊐ LƟHƟʊ

t j^h j dɑʃj lsjʕfŋjstsɔ^r ⊐ LƟHƟʊ Ɵɑɛʃʷ dʒjlsʃʊjfŋʕɑʃdʒ fŋʃtsfŋtsBtsdʒ, t j^h j dɑʃj dɑtsɢʃʊjfŋʕɑʃ⊐ LƟHƟʊ. t j^h j dɑʃj LƟHƟʊ tʃj tɛjBtstɛtsdʒ ɔfŋj⊐ ɔtsLdʒtsʏ dʑ⊐ ɔƟɑɛʃƟɑɑtsɔ. R fŋʃtsdʑ LtsɔƟɑɑʃj tʃtɛʃ tɛj^h j dɑʃʃ LƟHƟʊ lsƟBdɑʃʊ ʃ fŋ⊐j dʒ

t j^h j dɑʃj LƟHƟʊ, fŋtsHjtɛʏƟʰ ʃ⊐ LƟɔʃfŋʃdʒtsfŋtsʃ, fŋBʷL^r ɔƟ^f ʃj ɔj dɑʃʊʃʃdʑ: fŋʕtstɛtsfŋtsᴶ, ɔtɛjdʒʷ, tɛɢfŋfŋtstsʷdɑʃj, ʊjdɑƟ ʕtsdɑʃʊjfŋtsɔts, fŋtstsʃdʒtsfŋtsᴶ. ʅ HʃdɑʃʊʳʃLdʒjtɛjdɑʃʷ: dʒƟfŋʃŋ, tsBʲʸdʒƟ ʊjdʑ, tɛɢfŋfŋtstsʷdɑʃʷ, ɔtɛjdʒjdɑʃ, fŋʕtstɛtsfŋtsʃ. ʉɔʷL^ɪ dʒjʏHlzjHʃdɑʃʊƟtʒʃ ʃLdʒjtɛjdɑʃʷ ʕƟʏHtsʔ ɔjdɑʃʊʃʃdʑ.

t j^h j dɑʃj tsfʃɑtsɔdʑ ⊐ LƟHƟʊ dɑƟHtɛtsBʃ.

ɪ tɛjHfŋtsƟɔdjɢdɑʃj HƟɑtɑʑ⊐ ɔ ɔʃHj lsƟBdɑʃʊ, fŋtstsdʑBʊƟtsʳ ⊐ HʃƟɢtɛƟtʒdʒ,

ʅ Ɵɢɑʑʷ HdɑƟʸ ɢjtstdʒjlstɛʃʷ

ʅ Ɵɢɑʑʷ HdʑjtʃtɛjHfŋtsƟɔdjɢdɑʃʷ ts ʷʃɢʃztɛƟ⊐ dɑƟ tʃdɑtsfŋʕtsfŋtsʃ: lstsʊʕƟ, tʃtɛʷdʒƟʸ, tslstɛjLtsʕ, dʑzʊ, lzɢtsdz, dɑtsdʒƟɑƟʸ, dʒɑtsɢtslzɢtsdʑ dɑʃʕ, tsʕtɛlzʏdɑtsfŋtsᴶ, ʕtɛlzɢ. ʊɢtsdz ɪ tɛʷdʒtsʔ, tsfŋtstɛʳ ʔ, lslzʃtsʔ ʃtɛƟLɔʸtɛɑʑlsʳ ʔ lzɢʑdʑ.

ʅ dɑʃʃdɑƟtslstɛjLʕƟ, dʒjlstɛʃʊjfŋʕʃj, jHʃdɑʃʊʳ Hdɑʃʃdʑ. ʅ dɑʃʃdɑƟdɑtsdʒƟɑƟtsʔ, tʃjtɛʃdʒjlstɛ dʒɑtsɢtslzɢtsdʑ dɑʃʕƟ R Ldʒjtɛjdɑʃʃj ʃ tʃtsfŋtstɛtsjdɑʃj lzɢɑtsɔ fŋjtʃtsdʒtsʰ ɹ ʁ lstɛƟɑʑfŋjtstɛtsʃtɛƟ

ʅ Ɵɢɑʑʷ HdʑjtʃtɛjHfŋtsƟɔdjɢdɑʃʷ ts ʷʃɢʃztɛƟ⊐ dɑƟ tʃdɑtsfŋʕtsfŋtsʃ: dʒɑtsɢtslzɢtsdʑ dɑʃʕ, tʃtɛʷdʒtslzɢtsdʑ dɑʃʕ, ʕɔƟHtɛƟs, lstɛjlzɢtsdʑ dɑʃʕ, ts tɛƟɔjdʑfŋtsɔj ʷʃɢʃztɛ.

R LtsBtɛƟʜjdɑʃj ʷʃɢʃztɛ, ɔ lstsdʒ ʊʃʃfɑɢjdɑƟ ʕdjlsʊƟstsʔ BlzdʒƟɢj. ɪ tsfŋstɛtsjdɑʃj ʕtsdɑʷʃɢʃztɛƟʊʃʃʔ ʃʃtɛʷdʒtsʔ, tsʕtɛlzʏdɑtsfŋtsʃ dɑƟ djɢdɑʃdɑtsɔƟdɑɑtsʔ ʃ ʕdjlsʊƟstsʔ BlzdʒƟɢj. R fŋʃtsdʑ LtsɔƟɑɑʃj fŋɔtsʔfŋtsɔ fŋststɛtsdzʃ lzɢɑtsɔ tʃtɛʷdʒtslzɢtsdʑ dɑʃʕƟ, ʕɔƟHtɛƟsƟ

ɪ dɑtsʰ ƟH^ɪ tʃtɛʷdʒtslzɢtsdʑ dɑʃʕƟ ʃ dʒɑtsɢtslzɢtsdʑ dɑʃʕtsɔ, fŋtsfŋtsƟɔdjɢdɑʃdʑ⊐ ʃL tʃtɛʷdʒtslzɢtsdʑ dɑʃʕtsɔ, ɔ lstsdʒ ʊʃʃfɑɢjdɑʃ ʷʃɢʃztɛ, ʃLtsBtɛƟʜʸdɑʃdʑ⊐ dɑƟ ʕdjlsʊƟstsʔ BlzdʒƟɢj. ʅ HʃdɑʃʊʳʃLdʒjtɛjdɑʃʷ tʃdɑtsʰ ƟHʃ.

ʅ Ɵɢɑʑʷ HdʑjtʃtɛjHfŋtsƟɔdjɢdɑʃʷ ts tʃtɛtsfŋtstɛƟɑʑfŋtsɔjdɑʃdʑ⊐ ʷʃɢʃztɛƟ: tʃtɛʷdʒtslzɢtsdʑ dʑ? tʃƟɛƟɑɑdjɢdɑʃtʃʃjH, ʕlzB, dʒɑtsɢtsɢtɛƟɑɑʃʕʃ. R LtsBtɛƟʜjdɑʃj tʃtɛtsfŋtsjʔʰ ʃ⊐ dʒɑtsɢtsɢtɛƟɑɑʃʃtsɔ. ʅ ƟLɔʸtɛʃʕʃ ʕlzBƟ ʃ tʃƟɛƟɑɑdjɢdɑʃtʃʃjHƟ ʉtsLHƟɑɑʃj dʒtsHjdjɢʔ dʒɑtsɢtsɢtɛƟɑɑʃʃtsɔ (ʃL BlzdʒƟɢʃ, tʃtɛtsɔtsdtsʕʃ, tʃdɢfŋtsʃdɑʃʃƟʃ HtɛlzɢʃʃdʒƟtsɔjtɛʃƟɑtsɔ).

ʃ Bʲʸdʒ tʃtɛʷdʒtslzɢtsdʑ dɑtsɢts tʃƟɛƟɑɑdjɢdɑʃtʃʃjHƟ ʕlzBƟ ʅ HʃdɑʃʊʳʃLdʒjtɛjdɑʃʷ tsBʲʸdʒƟ

6 ʂ ∫ ϕʉ ʉ

ʅ ƟtslzɛƟɑɑdʑ dʑ j ʊʃʃfɑɢjdƟ

ϕ tɛɑʃʷ dʒjlsʃʊjfŋʕɑʃj Hjʔfŋjsɔʃʷ fŋ dʒɑtsɢtsLdɢƟʊɑʑ dʒʃ dɢƟtslzɛƟɑɑdʑ dʑ dʒʃ ʊʃʃfɑɢjdƟdʒʃ. Yʃfɑɢjdtsɔʳ j ɔʳ tɛƟʜjdɑʃʷ, tʃtstɛʷHtsʕ Hjʔfŋjsɔʃʔ, ʃfŋjtsdʑ LtsɔƟɑɑʃj fŋʕtsBtsʕ. R fŋʃtsdʑ LtsɔƟɑɑʃj tʃtɛʃ ɔʳ ʊʃʃfɑɢjdɑʃʷ⊐ tʃjtɛjdʒjfŋjsʃlsjdʑ dɑtsɢts ʃ fŋjsʊjlsƟsjdʑ dɑtsɢts fŋBtsʔfŋjsɔ fŋɑtsʏjdɑʃʷ ʃ lzdʒɑtsʏjdɑʃʷ, tɛƟfŋjtɛjHjdɑʃlsjdʑ dɑtsɢts fŋBtsʔfŋjsƟ lzdʒɑtsʏjdɑʃʷ. ʃ ʕtɛlzɢɑʃjɑʃj dɑƟslzɛƟɑɑdʑ dʑ⊐ ʊʃʃfŋjdz.

[j dzf lsj dzf f ‡te·lsdz j uf fnfdz· dz·uf BtsdzH f? tsBfi f? Hj dzf lsj dz f dz·uf dsj dzh j j tsBfi j j ‡te·lsdzsj . [j dzf dstsfnfsI fnfzdstsdr f tftetsf Lej Hj dzfw. [j dj dzfj fnjtsfnfs·l‡tsdz

[tetsBf

f BrI ‡dzsej dzfdz·w Htetsr, tsfnfdzsedzsj fnftsI?fnfsets Htetsf, fnfs‡te·fi j dzfj Htetsj?. H te·edj dzfj f lztftstcwHtsuf e·dzfj Htetsj?. [jh j dzfj L·H·uf dz·d‡tsf Hj dzfj uf fnfsf tsls uj dzscfts f uj dzscfts tfts j cfts uf fnfsf. [tetsB dzsj uf fnfdzs ‡l‡ tej Lzdz lsls Hj dj dzfw.] tej HfnfsedjdzfjHj fnwlsfuf dzs?Htetsf e ef Hj tsBr ‡dzsej dzdzs?Htetsf f ets Ldtsxf dzsfnfsH tftej HfnfsedjdzfwtsBr ‡dzsej dzdzs?Htetsf e ef Hj Hj fnwlsfuf dzs?. [j fnwlsfuf dz j Htetsf f dtj lstcfuj fn‡fi·w fnf fnfsj dt·i dtj tc. ftecfwdtj lsfuj fn‡f j Hj?fnfsefw f uf fnfdzser j er te·Hj dzfw fnjtsBr ‡dzsej dzdz dtf f Hj fnwlsfuf dz dtf Htetsfwdtf.

f lsdzsh j dzfj. [j dj dzfj e H·dzdzsdt tslsdzsh j dzf f. f ·fnh lsf3, tftetststcuf fw.] tef dtj dj dzfj tftetststcuf? tftef tej h j dzf f L·H·u,

] tsdzwlsf j tftetsuj dzs·I r uf fnfj dzfj tftetsuj dzs·tslsej dzf uf f dz f ej dzf uf f dz tfts jy tftetsuj dzslz. Ir te·Hj dzfj tftetsuj dzstse Hj fnwlsfuf dz dtf Htetsfwdtf. [jh j dzfj L·H·u dz· tftetsuj dzsr . Ir te·Hj dzfj tslsdzsh j dzfw ej dzf uf f dzej tftetsuj dzs·.

] tsdzsxf lsj dz dz j f tslstef uf·lsj dz dz j uf fnfdz·

] tsdzsxf lsj dz dz j f tslstef uf·lsj dz dz j uf fnfdz· ^j dz j uf fnfdz· f tsHlzdz uf fnfdz· Gj tsdtj lstef uj fn‡f·w f dzsj tef te lsuf w dtstsHlzdzw uf fnfdz· R LtsBte·Hj dzfj uf fnj dz dz· ‡tststeHf dz·lsdzs? tftewdtstsI?. Y f fnfdzser j tftetsdtj xlzls‡f. H te·edj dzfj uf fnj dz f te wdtj lsfuj fn‡fj Hj?fnfsef fnjtftsdzsxf lsj dz dz dtf f tslstef uf·lsj dz dz dtf uf fnfdz·dtf.

] tewdtslzctsdz dz·w fnf fnfsj dt·i ‡tststeHf dz·ls dz·tf dzsfn‡tsfnfsf. ‡tststeHf dz·ls lstsu‡f dz· tf dzsfn‡tsfnfsf, ·Bfnuf fnf·f f tste Hf dz·ls·] tsfnfstetsj dzfj lstsuj‡ f w f cfzte dz·‡tststeHf dz·lsdzs? tf dzsfn‡tsfnfsf.

ı lz‡ ej dzdz j er te·Hj dzfw

] tef dtj dj dzfj Blz‡e Hdzw L·tf f fnf dt·lsj dt·lsf uj fn‡f·, er te·Hj dzfw? f tftej Hdzsxf j dzfw?. H ets?fnfse·O·ef wdtj lsfuj fn‡f·, Hj?fnfsef?. ı lz‡ej dzdz j er te·Hj dzfw f uf fnfdzser j tftsHfnfs·dzse‡f. ı lz‡ej dzdz j te·ej dzfnfse·O dz·tsxf Hj dzfj dj f Lej fnfsdzsdts ‡tsdtf tsdj dzs·O ^ tstedtzdz , wtstedtzdz tfj tef dtj lste·O f tf dzsh ·Hf tfte wdtslzctsdz dzf‡·O ‡e·Hte·ls·O tsBj y dt·tf·e·dzdj dj tf f tf j H·O f ‡lzB·O

[jh j dzfj lsj‡fnfstser · L·H·u,

[jh j dzfj lsj‡fnfstser · L·H·u, ·ef wdtj lsfuj fn‡f dt fnftsfnfsBtsdt [jh j dzfj dzsc fuj fn‡f · L·H·u,. [jh j dzfj L·H·u, tfj tej Btstestdte fnj · etsLdtsxf dz · e·ef wdtdzsse.

[jh j dzfj L·H·u, fnfsHj te·Hf f · L·ef fnf dtsfnfsf, fnpwLr e·Hf h f · ej dzf uf f dz : fn‡tstestsfnfsI, etej dtw, te·fnfnfstswdzfj, uj dz·) ‡tsdzf uj fnfse ts, fnfstsf dtsfnfsI, tftetsf Letsf·lsj dz dzsfnfsI, etej dtw, tsBj y dt te·Btslsr . [Hf dzf ur f Ldtj tej dzfw: dt·fnfj, fnfstsf dtsfnfsf, te·fnfnfstswdzfw, etej dtj dzf, fn‡tstestsfnfsf. H ewLI dtj xlHz j Hf dzf uf·dtf f Ldtj tej dzfw ‡·Hts? ej dzf uf f dz .

[jh j dzfj L·H·u, fnpwL·dzdz · fj tslsdzsh j dzfj dt tftetststcuf tsdz·dz dzsfnfsI ej dzf uf f dz tftetsuj dzs·dtf; tejh j dzfj tsfnfdzsedz · L·H·u dz·Htetsf f f tftetsuj dzsr .

∫ ujdkO cf tftcf[cHc) tsctdzgdjdkfj tejLzdzlsOsO utsfmsO>djdkfj BlzcejdkdzP
>r tc=ljdkf? tfts lzfmqtsocfx LO=H=Ucf.

1 tejHfmsO>djdkfj H=OkdzP fmj tftsdjsh x lsO=Bdkfucf cf Hcf[Ogtc=Odjdj, u lstsdkBucOsr j
Hcf[Ogtc=Odjdr : ulsjdkfj cf tftsfmstcstsjdkfj . Y lsjdkfj ctdzgtso> fHcf[Ogtc=Odjdj

l O=gdzw Hdk=Oy gjtsdsjlstcf[w

l O=gdzy Hdzj tftejHfmsO>djdkf[w ts W cfGlztc=OfP dk=O tfdksfmctsfmsf[: lstsuc) tftcwdj=Oy, tslstejLtsc, dkzu, lzgtsdz, dksdj=Odk=Oy, dj=dksgtslzgtsdz dkf[c, ujls> tcyflzgtsdz dkf[c, lstejlzgtsdz dkf[c, tsctdzyksfmsx , ctdzg.

.l LO=f[dj=dksj tc=OfmftsdksyjdkfjHo=lzflftcwdjfflfdk=O tfdksfmctsfmsf[, tfOtc=Odkdjdzdzj tfcwdjfj, tfjtcftjdkBHf[clzdwtcdzj tftcwdjfj. R LdsjtejdkfjtcOfmfmstswdkf?: djjyHlz Holzdjw lstsuc=Odj=f[, tsls lstsuc=cf Hts tftcwdj=sc?, Hdkf[dk=Odj=Otch tdzlsOdk=Oc=oOHtc=Osdksc? fmjlsc=j.

R Ldsjtejdkfjcftfstfmstcstsjdkfjlzgtdksofmtftsdjsh x lstcOdkfmftstdscf[tcO .l cfHr lstejlzgtsdz dkf[c=tso: tsfmstcstslzgtsdz dz ?, tftcwdj=slzgtsdz dz ?, lslzftslzgtsdz dz ?, tcOodksBjHtejdkdz ?, tcOodksfmststcstsdkdkf?. Yjlsr tcyflzgtsdz dkf[c, tftcfdsjtcr ujlsr tcyflzgtsdz dkf[c=tso. 1 tcwdj=slzgtsdz dkf[c, c=oOHtc=Ok: cffmftsdz Ltso=Odkfj fmpts?fmpo fmjststcstsdz, lzgtdkso, Hcf[Ogtcsdk=Odj?. R LtsBtcOljdkfj gjtsdsjlstcfujfmcfflfdk=O dj=dkfdksoOdkdks? Blzdj=Ogj fmj cffmftsdz Ltso=Odkfj dj=ujcftcc=lzdw, dkfdj=?c, lzgtsdz dkf[c=O) lstcOdkfmftstcslcftcO 1 tsfmstcstsjdkf[w dk=Oc=djlsuOstsP BlzdsOgj.

1 jtcfdsjlstc dsdksgtslzgtsdz dkf[c=O 1 tsdzylscfj tfdksh OHcf W cfGlztcr, jHcfdkfur cfLdsjtejdkf[w tfdksh OHcf. 1 tcfBdkfxydkdksj cfLdsjtejdkfj tfdksh OHcf W cfGlztc, o lstsdj=ufmfqj dk=Oc=oOHtc=Osdksc? fmjlsc=j. 1 tcfBdkfxydkdksj cfLdsjtejdkfj Hdkfdz tsctdzxjdksfmsf[, tfdksh OHcf c=tdzgO

H cfdjdsjlstcf[w: ujdkstcOdz dk=Oy, tsfmjoOy cf Ljtsc=Odz dk=Oy fmfdjdsjlstcfcf.

1 tsfmstcstsjdkfj fmfdjdsjlstcfudz W cfGlztc.

l O=gdzyHdzj tftejHfmsO>djdkf[w ts tftcstsfmstcOdkfmsojdkdz W cfGlztcO: tfOtcOdkdjdkjtfcftfjH, c=lzB, tftcfLdj=O) tfcftcOdj=fHO) c=tsdkzfm ufcfdkfdkHtc, h Ok cf fmyjtcO R LtsBtcOljdkfj tftcstsfmstcOdkfmsojdkdz W cfGlztc. 1 tcfdsjtcr tcOLytcstsc=dksgtsGtsGtcOdkdkf[c=tso, ufcfdkfdkHtcO cf c=tsdkzfmO H tsLHOdkf[j dj=sHjdj? tftcstsfmstcOdkfmsojdkdz W cfGlztc (cfL BlzdsOgcf, tftcstsotsdksc=cf, tfdkfmsfcfdkfdk=Ocf Htcdzgcf dj=Oksjtcf[Odkso).

1 tsdzylscfj tsBjydj=O) jHcfdkfur cfLdsjtejdkf[w tsBjydj=O ∫ Bjydj tfcwdj=slzgtsdz dksgts tfOtcOdkdjdkjtfcftfjHO) c=lzBO

[illegible]

[illegible]

[illegible]:

1) [illegible]:

[illegible];

2) [illegible]:

[illegible];

3) [illegible]:

[illegible];

4) [illegible]:

[illegible];

5) [illegible]:

[illegible];

6) Ẅ ɋʟɔɋʋɥfɧʂtʂɥ ɘtʂfɳɟɋlʂOɒɯʃɥ, Ẅ tʂtɔɒʂɟɋtɑtʂɘOɒɯʃɥ ʕʟzɒźlʂɮtɔʳ ʟHtʂtɑtʂɘʲ ᵂ ɋ ʲɒʂʊɋɟtʂɒɒOɒźɒʃ ɒɯʂɑtʂʁɒɒOɑtʂtʂɟtʂɒɯzʊɋᵂ:

ɑtʂlʂtʂɘɒɯʂfɳʂᴵᴿ tʃtɔɋɒʒɥɒźʷlʂᴵ ɒʒOʂʒɥɒʒOʂɟɋʊɥfɧɋɟɥ ʟɒźOɒźɋᵂ ɘ ɋɒɒʂɥtɔɥfɳOᴱ fɳɘtʂɥɑtʂ ʟHtʂtɑtʂɘʲ ᵂ, ɘɥHɥɒźɋᵂ ʟHtʂtɑtʂɘtʂɑtʂ tʂʁtɔOʟO ʏɋɟʟɒźɋ (ʟHtʂtɑtʂɘtʂɥ tʃɋɟlʂOɒźɋɥ, fɳʁOɒźOɒźɑɳʃtɑtʂɘOɒɯźɒ ʕ tɔɥʏɋɟɒʒ ʟOɒźʷlʂɋʔ ɋ tʂlʂʜʳ ᴱO tɔɥɑźzɒźʷtɔɒźOʷ Ẅ ɋʟɔɋʋɥfɧʂOʷ Oʕlʂɋɘɒʂfɳʂᴵ), fɳʷtɔɒʒɋtɑtʂɘOɒɯʂfɳʂᴵᴿ ɒźOɘʳ ʕO tɔɥẄɒɥʂɋfɳʃɋɟ, tʃtɔɋʟɒźOɒźɥɒʒ ɑtʂtʂɟtɔɘOɒźOtʂʰ ɋʁʕzɋ ʟʂOʕtʂɑtʂ ʏɋɟɒʒ ɑtʂɘOHtɔzɑtʂɑtʂ ʋɥɒʂɘɥʕOᴱ:

7) ᴵʕtʂɒʂɑɑɋʋɥfɧʂtʂɥ ɘtʂfɳɟɋlʂOɒźɥɒʒ:

tʂtɔɋɥɒʂOʋɋɟɥʔ ɒźOtʃtɔɋɒʒɥɒźɥɒʒɋɟɥ ɒʒOʂʒɥɒʒOʂɟɋʊɥfɧʂɋᴱ ʟɒźOɒźɋʔ Hɒźʷtɥɟʰ ɥɒźɋᵂ ʟOHOʋ ɘ tʂʁɒźɒźfɳʂɋ fɳɑᴱtɔOɒźɒʂfɳʂɋ tʂʕtɔzʏOᵞʰ ɥʔ fɳɑɥHʳ, tʃɒźOɒźɋtɑtʂɘOɒźɋᵂ tʃtʂfɳʂzʃʕtʂɘ ɋ tʂʋɥɒźɋʕɋ ɋᴱ ɘtʂLɒźʂʏɋɒźʷᴱ ɥʔ fɳɑɥHʳ, tʂfɳɑLɒźOɒźɑɳɥɒʒ ɑɒźʂʁOɒźɒźɑtʂɑtʂɘOᴱ ɒźOʕtʂɑtʂɋɟɥɒźOᴵʕtʂɒʂɑɑɋʋɥfɧʂɋᴱ tʃtɔɑʂʁɒɥɒʒɋ tʃʃzʂɥʔ ɋᴱ tɔɥʰ ɥɒźɋᵂ;

8) OHOʃʂOʋɋᵂ ʕ ɋʟɔɒʒɥɒźʷᴿ ʰ ɋɒʒʃɳ ʂzfɑɒźɘɋᵂ ɒʒ fɳʂʋɋOɒźɒʂɘʔ ɋ tʃtɔɋtɑtʂHɒźʂɘʔ fɳɑɥHʳ:

ɑtʂlʂtʂɘɒɯʂfɳʂᴵᴿ ʕ Hɥʔfɳɘɋᵂɒʒ ɘ ʂzfɑɒźɘɋᵂᴱ ɒźɥtʂʃtɔɥHɥɒźɒʂɑɑɋʋɥfɧʂɋ, tʃtɘʳʰ ɥɒźɋᴿ ʂztɑtʂɘɒźʷ fɳɘtʂɥʔ ʕtʂɒʒʃɥʂɘźɒʂɑɑɋʋɥfɧʂɋ ʋɥtɔɥʟ tʃtɔOʕlʂɋʋɥfɧʕᴿ Hɥᵂlʂɥɒźɒʂɑɑɋʋɥfɧʂᴵ, ɘ lʂtʂɒʒ ʋɋʃɳɑɑɥɥźɒʂɒʂɑɑɋʋɥfɧʂɋ ʂzɒʒɥɒźɋ ʂzʋɋʃɳlʂᴵ fɳʸ ʂz Hɒtɔzɑɋʃɳᴱ ɒźɒʂ Hɥʔ, tʃtɔɋtʂʁɒtɔɥlʂOʂᴵ ɘ fɳɘɒʂɒźɥɒʒ fɳɘʂɒʂɑɑɋʋɥfɧʂɋ ɒʂɘʔ Hɥᵂlʂɥɒźɒʂɑɑɋʋɥfɧʂɋ ɒʂɘʳɥɒźɋᵂ, ɒźOɘʳ ʕɋ ɋ ʕtʂɒʒʃɥʂɘźɒʂɑɑɋʋɥfɧʂɋ ɒʂɘʳɥɒźɋᵂ;

ɒźɥtʂʁɳᴱtʂHɋɒʒʂɑɑɋʋɥfɧʂɋᴵᴿ ɘ Ẅ tʂtɔɒʒɋtɑtʂɘOɒźɋɒʂɘʳᴱ ɒźɒʂ ʟɒźOɒźɋʔ, ɘ lʂtʂɒʒ ʋɋʃɳɑɑɥɥźɒʂɒʂɑɑɋʋɥfɧʂɋ Ẅ tʂtɔɒʒɥźɒźɋtɑtʂɘOɒʂᴵ ɋHɥɒźɋ, tʃtʂɒźᵂlʂɋᵂ, ɑɋʃɳʃtʂlʂɥʟʳ tʂʁ tʂʁʲ ɥʔʕlʂOᴱ ɋ ᵂɘɒźɥɒźɋᵂᴱ, ɘ lʂtʂɒʒ ʋɋʃɳɑɑɥɥźɒʂɒʂɑɑɋʋɥfɧʂɋ tɔOɒźɥɒźɥ ɒźɥɋʃɳLɘɥɒźɒʂɑɑɋʋɥfɧʂɋ ɒʂɘʳɥɒźɋᵂ Hɥᵂlʂɥɒźɒʂɑɑɋʋɥfɧʂɋ fɳʁʂʁfɳɥɒźɘɥɒźɒʂɑɑɋʋɥfɧʂɋ ɋ tʃtɔɋʟɒźOɒźɋʔ, tʃɒźOɒźɋtɑtʂɘOɒʂᴵ fɳʁɥʸ tɔOɘɋʃɳlʂɋɥɒźɥ;

fɳɑtʂfɳʂʁɒźɒʂɑɑɋʋɥfɧʂᴵᴿ tʂfɳʂʟɒźOɒźɘOɒʂᴵ fɳɘtʂɥɟfɳɑtʂɘɒźʷᴿ fɳɑʃlʂzOɥɋᵂᴿ, ɘtʂfɳɑtʃtɔɋɒźɒʂɋɟɒʒɒʂᴵ fɳɘtʂɥɟfɳɑtʂɘɒźʷᴿ fɳɑʃlʂzOɥɋᵂᴿ ʕOʕ ɘʳ ʟtʂɘ, lʂtɥɟʁɒźʷᴿ ʰ ɋʔ ʕtʂɒźɒʂɋtɔɒʒɥtɔ, ʕtʂtɔɥɟʕlʂɋtɑtʂɘOɒʂᴵ tʃtɔɋɒźɒʂɋɟɒʒɒʂᴵ tɔɋʃɳɑɑɥɥźɒʂɒʂɑɑɋʋɥfɧʂɋ tʂʁtɔɋɒźɒʂɋɟɒʒɒʂᴵ tʂfɳʂɋ ɋ tʃtʂfɳɑtʃɋɟ ɋ tʃtʂfɳɑtʃɋɟɥHɥɒźɘɥɒźɒʂɑɑɋʋɥfɧʂɋᵂ, Ẅ tʂtɔɒʒɋtɑtʂɘOɒʂᴵ tʂtʃʳ ʟʂ.

ʃ ʅ ʊɸʅ ʇ ʅ ɾ ʃ ʅ ʊɾ ʅ ʒ ɾ ʇ ʅ ɾ ʅ ʋ ʃ ʔ ʊɸʊʒ

ɿ tʂʟɒźɒźOɘOʂɟɋɥɒźɒʂ ɒźɒʂ ɟ ɥɒźɒʂɑɑɋʋɥfɧʂɟɥɋɑɑɋʋɥfɧʂɋ ɒźɒʂ ɟ ɥʋɥʁʁɒźɒʂ ɟ ɥɘɥʔfɳɥɒźɒʂɑɑɋʋɥfɧʂɋᵂ

ɹ Oʇtɔɘʳ ɟ ɒźɒʂɑɑɋʋɥfɧʂɋɟɥɘɥʔfɳɥɒźɒʂɑɑɋʋɥfɧʂɋᵂ:

- ɘʳ ᵂɘɒźᵂlʂᴵ ɋ ᴱOɑɋʕlʂɟɥtɔɋʟtʂɘOʂᴵ fɳɥźʰ ɟ fɳɑɘɟɥɒźɒʂɑɑɋʋɥfɧʂɋ ɒʂɘʳɥɒźɋᵂᴱ tʂʁᴵʲ ɟ ʕlʂtɔɘ, tʃtʂɒźᵂlʂɋʔ, tʂlʂɒźɒʂʰ ɟ ɒźɒʂɋʔ ɒźɟɥʕ ʏɋɟʟ tʃtʂɒźᵂlʂɋᵂɒʒɋ, Ẅ tʂtɔɒʒɥźɒźɋtɑtʂɘOɒʂᴵ tʂtʃɋɟɥɒźɥɒʒɋɟɒʒɒʂᴵ tʃtʂɒźᵂlʂɋᵂ, ɥźfɳɑʂOɒźɒʂɘɒʂɋᵂ ʰ ɋʔ tʃtɔɋʟɒźOɒźɋʔ, ɒźɒʂɋʔ ɒʒɟɥʕ ʏɋɟʟ tʃtʂɒźᵂlʂɋᵂɒʒɋ, Ẅ tʂtɔɒʒɥźɒźɋtɑtʂɘOɒʂᴵ ɋ tʂʋɥɒźɋɘɒʂɋᵂ tɔɋʃɳɑɑɥɥźɒʂɒʂɑɑɋʋɥfɧʂɋ ɋ tʃtʂfɳɑtʃɋɟɥHɥɒźɘɥɒźɒʂɑɑɋʋɥfɧʂɋᵂ, Ẅ tʂtɔɒʒɋtɑtʂɘOɒʂᴵ tʂtʃʳ ʟʂ.
- ɘtʂfɳʃtɔɋɒźɒʂɋɟɒʒɒʂᴵ, Ẅ tʂtɔɒʒɥźɒźɋtɑtʂɘOɒʂᴵ ɋ tʃtɔɋɟɥtʂʁtɔOʟtʂɘʳ ɘOɒʂᴵ fɳɥźʸɥHɥɒźɒʂɑɑɋʋɥfɧʂɋᵂ: ɥźlʂɘɟɥtɔɥHɋʃɳlʂɋɥɒźɒʂ ɟ ɋ tʂlʂtɔɋʃɳOɒʂɋɟɒʒɒʂ ɟ, ɟɥHɋɒźɒʂɑɑɋʋɥfɧʂɋɟɒʒɒʂ ɟ, ʋɋOɑɳʂɒźɒʂ ɟ ɋ tʂʁʰ ɋɟɒʂ, ɥźlʂɘɟɥtɔɥHɋʃɳlʂɋɥɒźɒʂɑɑɋʋɥfɧʂɋ ɘɟɥtɔɥHɋʃɳlʂɋɥɒźɒʂ ɟ;

- ər wədzʷlsᴵ dʒɑ̈lsjɪdʒɑ̈lsʃuɻɪfŋʂʃɻɪ ʟɑ̈ʂtsɑ̈ʦsɑ̈ʦɪjɪtɑ̈ɑ̈ʦsfɪŋʦʃ, əʟɑ̈ʊʃɑ̈ʦsfɪŋʦʷʟɑʃ ʃ ʃʃtɑ̈ʦlsʃəʦstɑ̈jɪuʃʃʷ ə ʷʷɑ̈ʂlsɑ̈ɔ̈, ʜɑ̈ʊɑ̈ɑ̈ʑ ɔ̈, ɑ̈ɑ̈ʒɜɑ̈ʑ ʜjɪɑ̈ɑ̈ʃʷɔ̈ ʃ ɮlsəjɪtɑ̈ɤɪʜjɪɑ̈ɑ̈ʃʷɔ̈, ʃʃtɑ̈jɪʜɑ̈ʊɑ̈ɑ̈ʊɑ̈ʊɑ̈ʊᴵ ʂɪtɑ̈ʃlsjɪtɑ̈ʃʃɪɑ̈ʑʷ əʳ ʷəɑ̈jɪɑ̈ɑ̈ʃʷ ʟɑ̈ʂtsɑ̈ʦsɑ̈ʦɪjɪtɑ̈ɑ̈ʦsfɪŋʦʃʔ ʃ ʃʃtɑ̈ʦlsʃəʦstɑ̈jɪuʃʃʔ;
- ʜjɪɑ̈ʊɑ̈ʊɑ̈ʊᴵ əʳ əʦsʜʳ fŋ ʃfɪŋʃʦsɑ̈ʑʟʦsəʊɑ̈ɑ̈ʃjɪɑ̈ʒ ʟɑ̈ʂtsɑ̈ʦsə ɑ̈ʦsɑ̈ʊʂʃʂɪʃ, ʜjɪʜlʑʂlsʃəɑ̈ʑ ɔ̈ ʃ ʃɑ̈ɑ̈ʜlʑʂlsʃəɑ̈ʑ ɔ̈ lɑ̈ʒɤsʟɑ̈ʂɪɑ̈ʑ uɻɪɑ̈ɑ̈ʃʔ, lɑ̈ʒɤsʟɑ̈ʂɪɑ̈ʑ uɻɪɑ̈ɑ̈ʃʔ ʃʃʦ ʊɑ̈ʊɑ̈ʦsɑ̈ʊʂʃʃʃ;
- tɑ̈ʊʟʙʃʃtɑ̈ʊɑ̈ʊᴵ ʜʦsʂʊɑ̈ʟɑ̈ʊsjɪɑ̈ʑ fŋʦəʊɑ̈ʒɑ̈ʊɑ̈ʊɑ̈ʒɑ̈ʊsʃuɻɪfŋʂʃʃɔ̈ lɑ̈ɮlsəjɪtɑ̈ɤɪʜjɪɑ̈ɑ̈ʃʔ (ʃʃtɑ̈ʷɑ̈ʒʳ jɪ ʃ tsls ʃʃtɑ̈ʦlsʃəɑ̈ʦsɑ̈ʦʦs), ʃʃtɑ̈ʦəʦsʜʃʃlsᴵ fŋʊɑ̈ʒɑ̈ʦfŋʦʦsʷlsjɪɑ̈ʑ ɑ̈ʦs ɑ̈ɪjɪfŋʃɑ̈ʦsɤɪɑ̈ʑ jɪ ʜʦsʂʊɑ̈ʟɑ̈ʊsjɪɑ̈ʑ fŋʦəʊɑ̈ʒɑ̈ʊɑ̈ʊɑ̈ʒɑ̈ʊsʃuɻɪfŋʂʃʃɔ̈ ʷʷɑ̈ʂlsʦsə, əʳ fŋʦtɑ̈ʊʃəɑ̈ʊsᴵ ʊɑ̈ɑ̈ʒʃɑ̈ʑjɪɑ̈ɑ̈sɑ̈ʊʃʃɑ̈ʃʃʁ, ʃʃtɑ̈ʃəʦsʜʃʃlsᴵ ʃʃtɑ̈ʃɑ̈ʒjɪtɑ̈ʳ ʃ ʂʦsɑ̈ʦtɑ̈ʃɑ̈ʃʃtɑ̈ʃɑ̈ʒjɪtɑ̈ʳ, tsʙtsfɪɑ̈ʦsəʳ əɑ̈ʊsᴵ fŋʦʙfɪŋʦəjɪɑ̈ɑ̈ʑjɪtɑ̈ʊɑ̈fɪŋʃʑɤɪʜjɪɑ̈ɑ̈ʃʷ;
- əʳ ʙʃʃtɑ̈ʊɑ̈ʊᴵ fŋʃʦsfɪŋʦʙ tɑ̈jɪʰ jɪɑ̈ɑ̈ʃʷ lʑuɻɪʙɑ̈ʦsʔ lʊʜʊuʃ (fŋʃɑ̈ʊɑ̈ɑ̈ʃəɑ̈ʊsᴵ ɑ̈jɪfŋʂʦsɑ̈ʑʂʦs əɑ̈ʊɑ̈ʃɑ̈ʊɑ̈ʦsɑ̈ʦsə tɑ̈jɪʰ jɪɑ̈ɑ̈ʃʷ, əʳ ʙʃʃtɑ̈ʊɑ̈ʊᴵ ɑ̈ɑ̈ʊʃʃʙʦsɑ̈jɪjɪ ʃʃʦʜɔ̈ʦsʜʷʰ ʃʔ fŋ lʑuʸlsʦsɑ̈ʒ fŋʊɑ̈ʒɑ̈ʦfŋʦʦsʷlsjɪɑ̈ʑ ɑ̈ʦs əʳ ʜjɪɑ̈jɪɑ̈ɑ̈ʑ ɔ̈ ʂɪtɑ̈ʃlsjɪtɑ̈ʃjɪə).

ʁ ʊɑ̈ʦsəʳ jɪ ʃʃfɪŋʃɑ̈jɪɑ̈ʜʦsəɑ̈ʊsjɪɑ̈ʑ fŋʂʃʃɪjɪ ʜjɪʔfɪŋʦəʃʷ:

- ʃʃfɪŋʃʦsɑ̈ʑʟʦsəɑ̈ʊsᴵ əʦsʃʃtɑ̈ʦsfɪŋ ʂʊɑ̈ʂ ʃʃfɪŋʃɑ̈ʃɑ̈jɪʜʦsəɑ̈ʊsjɪɑ̈ʑ fŋʂʃʔ ʃʃɑ̈ɑ̈fɪŋʦʦɑ̈ʑɑ̈ʒjɪɑ̈ɑ̈s ʃʃʦsʟɑ̈ɑ̈ʊɑ̈ɑ̈ʃʷ, ʷʷʦtɑ̈ʦɑ̈ʒɑ̈ʑɑ̈ʊɑ̈ʃtɑ̈ʦsəɑ̈ʊsᴵ əʦsʃʃtɑ̈ʦsfɪŋ, ʷʷɑ̈ʃʂɪfɪŋʃtɑ̈ʑʁʰ ɑ̈ʃjɪ ʃʃtɑ̈ʦlsʃəʦstɑ̈jɪuʃʃjɪ, ʃʃtɑ̈ʦsʙɑ̈jɪɑ̈ʒʑ, fŋʊɑ̈ʒɑ̈ʦfŋʦʦsʷlsjɪɑ̈ʑ ɑ̈ʦs ɮfɪŋʦʊɑ̈ʊɑ̈ɑ̈ʊɑ̈ʃəɑ̈ʊsᴵ ʃʃfɪŋʂʦsɑ̈ʒʦsjɪ ʃ ʜʊɑ̈ʟɑ̈ʦsjɪ, ʷʷʦtɑ̈ʦɑ̈ʒʃɑ̈ʃʦsəɑ̈ʊsᴵ ɑ̈ʃʃɑ̈ʦsɑ̈ʦsɑ̈ʦsʃʃɑ̈ʃʃʁ ʃʃʦsʟʃuɻɪʁ, ɑ̈ʒɑ̈jɪɑ̈ɑ̈ʃjɪ;
- ʃʃtɑ̈ʦəʦsʜʃʃlsᴵ ʃʃʦ fŋʊɑ̈ʒɑ̈ʦfŋʦʦsʷlsjɪɑ̈ʑ ɑ̈ʦs fŋʦfŋʦəɑ̈jɪɑ̈ɑ̈ʦsɑ̈ʒʑ ʃʃɑ̈ʊɑ̈ʟɑ̈ʊɑ̈ʑ ɑ̈jɪfŋʃɑ̈ʦsɤɪɑ̈ʑ ʔ ᵢʂfŋʃjɪtɑ̈ʃɑ̈jɪɑ̈ʦs, ɑ̈jɪʙʦsɑ̈ʑʰ tsjɪ ʃʃfɪŋʃɑ̈jɪʜʦsəɑ̈ʊɑ̈ʃjɪ ʃʃʦ lɑ̈ɮlsəɑ̈ʊsəɑ̈jɪɑ̈ɑ̈ʃʁ tsfŋʦʙjɪɑ̈ɑ̈ʦsfɪŋʃjɪʔ ɑ̈ʒɑ̈ʊɑ̈ʊɑ̈ʃʃtɑ̈ʦsʃuɻɪfŋʂʃʦsɑ̈ʦ tsʙʲjɪʂlsʊ ʟɑ̈əɑ̈ʃfŋʃɑ̈ʃʃɑ̈ʦsfɪŋʃjɪʔ tsʙʲjɪʂlsʦsə ɑ̈jɪɤɪʜlʑ fŋʦʙʦsʔ;
- fŋʊɑ̈ʒɑ̈ʦsfɪŋʦʦsʷlsjɪɑ̈ʑ ɑ̈ʦs ʷʷʦtɑ̈ʦɑ̈ʒɑ̈ʑɑ̈ʊɑ̈ʃtɑ̈ʦsəɑ̈ʊsᴵ tsʙtsʙʰ jɪɑ̈ɑ̈ʃʷ ʃ əʳ əʦsʜʳ ʃʃʦ tɑ̈jɪʟɑ̈ʑɑ̈ʒlsɑ̈ʊsɑ̈ʊɑ̈ʒ ʃʃtɑ̈ʦsəjɪʜʸɑ̈ɑ̈ʦsɑ̈ʦʦs ɑ̈ʊɑ̈ʙɑ̈ʑ ʜjɪɑ̈ɑ̈ʃʷ, ʃʃfɪŋʃɑ̈jɪʜʦsəɑ̈ʊɑ̈ʃʷ, tsuɻɪɑ̈ʃəɑ̈ʊsᴵ ʜʦsfɪŋʦʦsəjɪtɑ̈ɑ̈ʦsfɪŋʃᴵ ʃʃtsɑ̈ɑ̈ʒuɻɪɑ̈ɑ̈ʑ ɔ̈ tɑ̈jɪʟɑ̈ʑɑ̈ʒlsɑ̈ʊsɑ̈ʊ, əʳ əʦsʜʦsə ʃ tsʙtsʙʰ jɪɑ̈ɑ̈ʃʔ;
- ʃʃtɑ̈ʦsɑ̈ʦɑ̈ʦsʟʃtɑ̈ʦsəɑ̈ʊsᴵ əʦsʟɑ̈ʒʦsɤɪɑ̈ʦsjɪ tɑ̈ʊʟəʃlsʃjɪ ʃʃtɑ̈ʦsuɻɪfŋʃʃ; Ɵ lsʊʂɤɪjɪ əʳ ʜəʃɡɑ̈ʊsᴵ ʃʃtɑ̈jɪʜʃʃtsɑ̈ɑ̈ʦsɤɪjɪɑ̈ɑ̈ʃʷ tsjɪɡʦs tɑ̈ʊʟəʃlsʃʃ əɑ̈ʦsəʳ ɔ̈ lɮfɪŋɑ̈ʦsəʃʷɔ̈.

ʈ ʊʙʦsɬsʊfɪŋʃʃɑ̈ʑʷ tsɑ̈ʒʊɑ̈ʃjɪʔ:

- əʳ ʷəɑ̈ʑʷlsᴵ ɑ̈jɪʜʦsfɪŋʂʊʦsɑ̈ʦsfɪŋʃᴵ ʃ ʃʃʟʙʳ lʦsɑ̈ʦsfɪŋʃᴵ ʃʃɑ̈ʑʷtsɑ̈ʒʊɑ̈ʃʃʃ, ʜʊɑ̈ʟɑ̈ʑ ɔ̈, ɑ̈jɪʙʦsɔ̈ʃʦsʜʃʃɑ̈ʑʑʷ ʃ ʟʊʜʊuʃʃ;
- əʳ ʙʃʃtɑ̈ʊɑ̈ʊᴵ, ʊɑ̈ʟʊɑ̈ʃʟʃtɑ̈ʦsəɑ̈ʊsᴵ, fŋʃʃfɪŋʂjɪɑ̈ʒʂʊʃʟʃtɑ̈ʦsəɑ̈ʊsᴵ ʃ ʃʃɑ̈ɑ̈ʦsjɪtɑ̈ʃjɪtɑ̈ʃjɪlsʃtɑ̈ʦsəɑ̈ʊsᴵ ʃʃɑ̈ʑʷtsɑ̈ʒʊɑ̈ʃʃʁ tɑ̈ʊʟɑ̈ʑʃuɻɪɑ̈ʑ ɔ̈, əʃʜʦsə ʃ ʷʷʦtɑ̈ʒʃʃtɑ̈jɪʜfɪŋʂʊɑ̈ʑjɪɑ̈ʃʷ;
- əʳ ʙʃʃtɑ̈ʊɑ̈ʊᴵ ʷʷʦtɑ̈ʒʑ ʃʃtɑ̈jɪʜfɪŋʂʊɑ̈ʑjɪɑ̈ʃʷ ʃʃɑ̈ʑʷtsɑ̈ʒʊɑ̈ʃʃʃ ʃ ʃʃɑ̈ʑʑ fŋʂʦtɑ̈ʃtsəɑ̈ʊsᴵ tɑ̈jɪʰ Ɡjɪʒʳ jɪ ʟʊʜʊuʃ fŋʃjɪʒʊʒʃ, ʜʃʊɡtɑ̈ʊʒʊʒʃ, ʃɑ̈ʦsʔ ɡtɑ̈ʊʷʃʃʂʦsʔ ʃ ʃʃʃ ʂʦsʒʙʃʃtɑ̈ʃɑ̈ʒʊɑ̈ʃʃʷ ʒʃʃ;
- tsuɻɪɑ̈ʃəɑ̈ʊsᴵ ɑ̈ʃʜʸɤɪɑ̈ʦsfɪŋʃᴵ ʃʃɑ̈ʑʷtsɑ̈ʒʊɑ̈ʃʃʃ ʃʃʦ ʂtɑ̈ʃlsjɪtɑ̈ʃʷʒ ʃʃtɑ̈jɪʜɑ̈ʦsɤɪjɪɑ̈ɑ̈ʑ ʒ lzuʃʃlsjɪɑ̈jɪʒʃɑ̈ʃʃ fŋʷtsɑ̈ʒʊɑ̈ʃtsəɑ̈ʊsᴵ ɑ̈ʒfŋʊʒsfɪŋʦʦsʷlsjɪɑ̈ʑ ɑ̈ʦs.

ʂ tsɑ̈ʒɑ̈ʒʊɑ̈ʃʂʊsɑ̈ʊsᴵ ɑ̈ʑ jɪ lɑ̈ɑ̈ʃəjɪɑ̈ʃfŋʊɑ̈ʑ ɑ̈ʑ jɪ lzuɻɪʙɑ̈ʑ jɪ ʜjɪʔfɪŋʦəʃʷ:

- əʦsfŋʃʃtɑ̈ʃɑ̈ʃɑ̈ʒʊɑ̈ʃʃʃ ʃ ʷʷʦtɑ̈ʒʊɑ̈ʃʃʃ fŋɮɤɪʜjɪɑ̈ɑ̈ʃʷ ə fŋʦʦsʦlsəjɪsfɪŋʦəʃʃ fŋlzfɪŋɑ̈ʦsəʃʷɔ̈ʒʃ ʃ ɻuɻɪɑ̈ʑʷɔ̈ʒʃ tsʙʰ jɪɑ̈ɑ̈ʃʷ, ʷfɪŋɑ̈ʦs, lʦsɻɑ̈ʦs, ɡtɑ̈ʊɑ̈ʒʦsɬsɑ̈ʦs əʳ tɑ̈ʊɤɪʊʃʃ fŋʙʦsʁ lʦsɻʂlz ʟtɑ̈jɪɑ̈ɑ̈ʃʷ

ə lzfnjsdz¬ cf tfcfnjdzjdzdz¬ lsjçfnjsO¬, HO⊕Osl tftswfqjqdzfw tfts¬tsHlz tqjh jdzfw LOHOqcf, çtsdzdzjdzsftətsəOsl tftsdlzqjdzdz¿ tqjLlzdzlsOls;

- ə¬tsHj tsBfnjzçHjdzfw LOHO⊕Osl ətstftətsfnj tftsfnjzh jfnjsəlz tsBfnjzçHGjdzs¿ lsjdz , tftətsBdjdz , tqjh Gjdzs¿ LOHOqcf, ərfnçOLr əOsl cfHjcf, dzOqjdjdzdz¿j dzO⊕tftscfnç tqjh jdzfw, fntstftsfnjsə⊕dzwls fntscf fnjzçHjdzfw fnj fnjzçHjdzfwdzcf Htdzgcf¬ lzqOfnjsdzfçtsə HcfOdzsgO tsBdzOdzçcfəOsl tdOLdzqcfj cf fn¬tsHfnjsəts tftsLcfqcf¿, ə çtstətqjçlsdzs¿ wwtstədzj wwtstədzdzdzftətsəOsl tdOLdzsgdzOfnfw, fntscf ətsLtdO⊕jdzfw;
- tftqjHfnjsO⊕dzwls tqjLlzdzlsOlsr tqjh jdzfw LOHOqcf, lçfnjjtəcfdzjdzsO cffnfnjqjHtsəOdzfw, tftətsjçlsO fnO⊕dzsfnjstswlsjdzdzs ər BcftdOsl wwtstədzOls ər fnjslztfdjdzfw fnjlzqylstsdzLOHOqtfqjLjdzsOqcfcf cftsfntsBjdzdzsfnjsj¿ Olzhcflststəcf;
- tftsdzfdzOsl cf cffntfstdzLtsəOsl tftqjcfdzzfh jfnjsəO çtsdzO⊕dzdzs¿ cf cfdzHcfəcfHlzO⊕dzdzs¿ tdO⊕Btslsr tftəcftqjh jdzfw cf lzqjBdz¬dzOsjdzOslcfqjfnçf¬LOHOq;
- tftəcfdzfdzOsl qjdz fntsədzjfnjsdzs¿ Hjwlsjdzdzsfnjscf, tfdzO⊕dzfçtətsəOsl tstəgO⊕dzfLOqcfg fntsədzjfnjsdzs¿ tdO⊕Btslsr , tdOfntftqjHjdzwls əcfHr tdO⊕Btsls, HtsgtsəO⊕dcfəOsl fny, tsBfnjzçHOsl tftətsqjfnfncf tqjLlzdzlsOls tdO⊕Btslsr , tsBtsBh Osl dzdjdzfw djfnçtsdzçf¬ dz Hj¿;
- lzqOfnjsətsəOsl ə gtdztftsər¬ wwtstədzO¬ tdO⊕Btslsr (tsBfnjzçHjdzfw, tsBdjdz dzdjdzfwdzcf, dzstLgtsər j h lszdtədzr cf Htdzgcfj), ər tftsdzdzwls fntsg qOfnjs tdO⊕Btslsr cf çtststəHcfdzftətsəOsl fntscf Hj¿fnjsəcfw fnj Htdzgcfdzcf qdjdzO⊕dzcf çtsdzO⊕dzHr , tsqjdzfəOsl çOqjfnjsəts fntsjgts əçdzO⊕HO ə tsBh cf¿ tftətsHlzçls tfts çtəcflsjtəcfwdz, fnywtstədzdzdzftətsəOdzdz dzlzqOfnjsdzfçtOdzf əLOcfdzstHj¿fnjsəcfw.

tjdlzdzwlscfədz¿ jlzdzfəjtəfnjOdz dz¿ jlzqjBdz¿ jHj¿fnjsəcfw

qOdzststəgOdzfLOqcfw:

- fnjOdzstfnjstswlsjdzdzs fntsfnjsO⊕dzwls tfjdzOdz, Odzgtstəcflscdzftqjhjdzfw LOHOqcf (cfdzf jgts qOfnjs), ərBcftdOsl fntftsfntsB tqjhjdzfw fnjlzqylstsdzcfdzjqfh cf¬fny tqjfnjztəfntsə cf fntsBfnjsəjdzdz¬ ətsLdzsyçHdzsfnjsj¿, OtəgHzdzjdzsftətsəOsl cf çtstətqjçlscftətsəOsl əOtəcfOdzsr tqjhjdzf¿ fnjlzqylstsdzdzsətsOdzf¿ cfdzwtstədzfOqcfcf.

qOdzsçtsdzstətsdz, ldzsqcftsdzOdz dz¿ cfdzsjdzdzçls:

- ədzOHjls fntftsfntsBO⊕dzfcf fnjOdzstftətsəjtəçcf, fnjOdzsçtsdzstətsdzw tftətsqjfnfnjOcf tqjLlzdzlsOlsO tqjh jdzfw dzOlsjdzOlsfcfqjfnçts¿ LOHOqcf;
- tftqjHəcfHjls lstdzHdzsfnjscf, çtslststər j dzsgHzls ətsLdzfçtdzzls tftəcf tqjh jdzfcf LOHOqcf, ədzsfnjls çtstətqjçlscfər ə Hjwlsjdzdzsfnjs dzO tsfnjdzsəj dzsər¬ tsBfnjstswlsjdzfnjsə, dzO¿Hjdzdz¬ tsh cfBtsç, ərwədjdzdz¬ lstdzHdzsfnjsj¿;
- tsqjdzfəOsl fntstslsəjlsfnjsəcfj tqjLlzdzlsOlsOHjwlsjdzdzsfnjscf tftsfnjsO⊕djdzdzs¿ qjdzf cf lzfnjdzsəcfwdz tsBjwfnjdzwls tftəcfqcfdz Htsfnjscfxjdzfwcfdzf djHtsfnjscfxjdzfw qjdzf, dzO¬tsHcflsh tsh cfBçlz, HO⊕Osl tsqjdzçlz tftəcftsBtqjlsydzdzsdzz tstfr lslz.

ɨ tɾrʃ ɾʊɭ ʒ ɾ tɾɭʊʃ ʔʊɸʊʒ

ङ ङtscdzulz tsBlzuijdzkfw ə 5 ङdzGfnfgj tsBlzuGy h cf?fny tftscdlzucfls fnqjqHlzg h cfj tftejHcdzjlscdz j tejLlzcdzlsGlsr :

Y cfnqdzOcf ər ucffnqjqdzkfw

1 tscdzkfcdzGlsI cf tftaGəcfcdz dzks lztftslstejBdzwlsI lsjtacdzgfcdz, fnpwLOdzkdz j fn dzkslztaOdzdz cdzgf ucffnqdzOdzgf, tsBr ङdzksəjdzkdz cdzgf cf Hjfnywlscfucdz cdzgf HtatsBwcdzgf.

u tGəcdzkfəGlsI cf lztftstcwHtsucfəGlsI dzkslztaOdzdz j ucffnqdzO fnqaGəcdzkfəGlsI ə tftatsfnjsj?h cf=fnqldzuGy=tsBr ङdzksəjdzkdz j HtatsBcf, Hjfnywlscfucdz j HtatsBcf.

u tstslscdzsfnqflsI lstsuङlz dzkO ङtststaHcfdzkslscdzks? (ucffnqdzstəts?) tftcwdzgs? fn fnjstslsəjlsfnjsəlzg h cfdzj? ucffnqdzstcdzj cf cfLtsBtaOyGlsI dzkslztaOdzdz j ucffnqdzO lstsuङGdzgf dzkO ङtststaHcfdzkslscdzks? (ucffnqdzstəts?) tftcwdzgs?.

Jr tftscdzdzwlsI Gtacfwdzjlscfujfnjcfj Hj?fnjsəcfw fn dzkslztaOdzdz cdzgf ucffnqdzOdzgf, fn tsBr ङdzksəjdzkdz cdzgf HtatsBwdzgf ə tftatsfnjsj?h cf=fnqldzuGy=.

Jr tftscdzdzwlsI tftatsəjtcङlz, tftacfङcfHङlztejLlzcdzlsGlsOər ucffnqdzjdzkf?.

ʃ ङtalzGdzwlsI dzkslztaOdzdz j ucffnqdzO

tjh jdzkfjlsjङfnjstsər = LGHGuq

tjh Gls lsjङfnjstsər j LGHGucf Gtacfwdzjlscfujfnjcfdz fnjftsfnjsBtscdz cf fn tftscdzgtsh ɹ ʁ tstaGOdzkfLtsəOdzkdzksGts ङtscdjqucdzksGts tfjtejBtstaOəfj=ətsLdzgsxdz=əOtacfOdzkstsə.

tjh Gls LGHGucf, fnjsHjtcxOy cfj LGəcffnqfdzgtsfnjscf, fnpwLr əOy h cfj əjdzkfucfdz: fnjङtstatsfnjsI, ətejdzgw, taGfnfnjstswdzkfj, ujdzkO ङtscdzkfujfnjsəts, fnjstscfdzgtsfnjsI.

R fnjftscdz Ltsəcls ङtaGsङcfj LGyfcffnqf, fn=jdzg, lsOədzkfur, tsBtsLdzkOujdzkfw tftacf tejh jdzkfcf LGHGuq.

1 tscdz Ltsəcls fny tsfnqdzsədzkdzgf cfLdzjtejdzkfw: ujdz, dzgGfnfnj, taGfnfnjstswdzkfw, ətejdzjdzkf, fnjङtstatsfnjscf, ər taOxGls tsHdzkf jHcfdzkfur əjdzkfucfdz ujtejL HtalzGcfj.

R Lədzjङls, GdzkOdzkfLcftatsəcls, tsujdzkfəcls cfdzwtstacdzgOucfx, tftejHfnjsOədzjdzkdzx ə lsOədzkfujdz, dzkO fnjstsdzəucfstsङ HcfOGtaOdzgdzj, cfdzksjtaftejlscftatsəcls tftejHfnjsOədzjdzkdzx j HOdzkdzx j, cffnjftscdz Ltsəcls HOdzkdzx j tftacf tejh jdzkfcf LGHGuq.

l OGdzw HdzkOy cjjtscdzjlstacfw

1 tscdz Ltsəcls fny cjjtscdzjlstacfujfnjङcfdzgf tftscdzwlscfwdzgf: lstsuङG, tftcwdzgOy, tslstejLtsङ, dzzuq, lzGtscdz, dzgdzksGtslzGtscdz dzkfङ, tsङtalzxdzksfnjs, ङtalzG.

1 tacfətsHcfls tftacfdzjtcr tsBj jङGlstsə tsङtalzxOy h jGts dzgftaO, cfdzjx h cf= wtstacdzjz cfLlzujdzkdz = cjjtscdzjlstacfujfnjङcf= wcfGlztc.

R fnjftscdz Ltsəcls lsjtacdzgfdzksdzksGcfx, fnpwLOdzkdzx fnjlzGdzkOdzgf: əjtch cfdzkO, fnjststatsdzkO, fn dzgdzksGtslzGtscdz dzkfङOdzgf: lzGtscdz, əjtch cfdzkO, fnjststatsdzkO, HcfOGtscdzkOdz, fn tsङtalzxdzksfnjs ɹ ʁ : taOHcflzfnj, HcfOdzjlstc, ujdzkstc.

R LtsBtaOyGls cfLlzujdzkdz j cjjtscdzjlstacfujfnjङcfj wcfGlztcr dzkO dzjdzkfdzksəOdzkdzks? cf ङdzjlsuGstsङ Blzdzgcjfnjftscdzgsh ɹ ʁ ucftaङlzdzw cf dzkfdzjङcf.

ا ⊙⊐tsнqlsᴵ нdzqdz tslstqjL$tsə djqtstsfmqjнfmsəjdzdz dʒ qLdʒjtqjdzqjdʒ fmtstsdzsʰ ɹʁ dzqdqj?$q, fmststsqlsᴵ tslstqjL$q LO⊙нO⊙dzdzs? нdzqdz ; fmststsqlsᴵ ts$tdzxdzsfmsᴵ LO⊙нO⊙dzdzsGts tɛO⊙нqlzfmO

ʀ fmtstsdzLtsəO⊙sᴵ fmtsfmsəO⊙fmstststsdzqlzGdzsə tstʷdʒslzGtsdzdzq$O, $əO⊙нtɛO⊙sO⊙нdzʷ q⊐ tstsfmststsjdzqʷ, əʳ uqfmqjdzqʷ tsdzsʰ O⊙нq qtjjtɛqdʒjlstɛO

ɹʳ uqfmqzʷlsᴵ tjjtɛqdʒjlstɛ q tsdzsʰ O⊙нᴶ $əO⊙нtɛO⊙sO, tstʷdʒslzGtsdzdzq$O, ʷqGzts, fmsfmsO⊙djqdzdz ⊐ qL tstʷdʒslzGtsdzdzq$tsə, ə lstsdʒ uqfmqjᵩ ʷqGzts, qLtsBtɛO⊙ʸdzdz ⊐ dzO $djqlsuO⊙stsʔ Blzdʒ⊙Gj .

ɻ tsdzLtsəO⊙sᴵ fmʸ tsfmqzsədz dʒq djqlstɛquqfmqjq$qdʒq jнqdzquO⊙dʒq qLdʒjtqjdzqʷ нdzqdz , tsdzsʰ O⊙нq; əʳ tɛO⊙ʸO⊙sᴵ tsнdzqjнqdzquʳ əjdzquqdz uqjtqjL нtɛzGqj .

ʇ O⊙fmtstsLdzO⊙sᴵ tsO⊙tɛO⊙dzqjdqjtsqtsjн, $lzB, qfmtstsdzLtsəO⊙sᴵ lsjtɛdʒqdzsdzsGqᵍ : əjtɛʰ qdzO tqjBtɛts, GtɛO⊙dz, qLdʒjtqjdzqʷ, dzO⊐tsнqlsᴵ qLdʒjtqjdzqʷ tsO⊙tɛO⊙dzqjdqjtsqtsjнO $lzBO

ɹʳ uqfmqzʷlsᴵ tsBʲʸdʒ $lzBO tsO⊙tɛO⊙dzqjdqjtsqtsjнO tsts LO⊙нO⊙dzdz dʒ qLdʒjtqjdzqʷdʒ tstsdzLtsəO⊙sᴵ fmʸ jнqdzquO⊙dʒq qLdʒjtqjdzqʷ tsBʲʸdʒO

ʇjʰ O⊙sᴵ djqfmqtsxdz j LO⊙нO⊙uq dzO qLdʒjtqjdzqj Gjtsdzjlstɛquqfmq⊐ əjdzquqdz ə tsO⊙tɛO$lsquqfmq⊐ fmqlslzO⊙uqdʷ⊐.

ş $tsdzqlz tsBlzuqjdzqʷ ə **6 $tdzO⊙fmqjj** tsBlzuO⊙ʸ ʰ q?fmʸ tstsdzzuqlls fmqjqнlzᵍ ʰ qj tstqjнdʒjlsdz j tqjLlzdzlsO⊙sʳ :

ʏ qfmqtzO⊙q əʳ uqfmqjqjdzqʷ

ɪ dzO⊙sᴵ q tstsdzqdʒ⊙sᴵ lsjtɛdʒqdz , fmqʷLO⊙dzdz j fmtɛOLdzquqdz dʒq əqHO⊙dʒq uqfmjdz q fmtstsfmtsB⊙dʒq q⊐ LOtqfmq, tjjtqj⊐tsнqlsᴵ (jfmqtzq ɻlsts ətsLdʒsxdzs) tsls tsнdzs? ʷtstɛdʒʳ LOtqfmq uqfmqtzO$ нtɛzGts? .

ʉtɛO⊙dzqjəO⊙sᴵ q lztjtstɛʷHtsuqjəO⊙sᴵ ujdz j uqfmqtzO tsBʳ $dzsəjdzdz j q нjfmʸlsquqdz j нtɛtsBq, fmqO⊙dzqjəO⊙sᴵ uqfmqtzO⊙tsнdzsGts q tɛOLdz ⊐ LdzO$tsə.

ɹʳ tjtsdzdzʷlsᴵ , fmtsujlsOʸ lzfmtsdz j q tjqfmjdzjdzdz j tjtɛqʸdʒʳ , O⊙tɛqʷdzjlsquqfmjqjj нjʔfmtsəqʷ fmj dzOslztɛO⊙dzdz dʒq q uqjdz dʒq uqfmqtzO⊙dʒq, tsBʳ $dzsəjdzdz dʒq q нjfmʸlsquqdz dʒq нtɛtsBʷdʒq, tjtsdzsxqlsjdzdz dʒq q tslstɛquO⊙sjdzdz dʒq uqfmqtzO⊙dʒq.

ɹʳ uqfmqzʷlsᴵ LdzOujdzqʷ uqfmqtsəʳ ⊐ əʳ tɛO⊙ʸjdzq?, əʳ tjtsdzdzʷlsᴵ tjtɛqqHlz q tsujdz$lz tqjLlzdzlsO⊙sO əʳ uqfmqjqjdzq?, əʳ tjtsdzdzʷlsᴵ tjtqjtsBtɛOLtsəO⊙dzqʷ uqfmqtsəʳ ⊐ əʳ tɛO⊙ʸjdzq? dzO⊙tsfmqtsəjfmtstsʔfmtsəO⊙tɛqʷdzjlsquqfmjq⊐нjʔfmtsəqʔ.

ʉtststsdzsfmqjlsᴵ ststuq$lz dzO $tststɛHqdzOstsdzsʔ tjtɛʷdʒtsʔ fmfmtststsəjlsfmtsəlzᵍ ʰ qdʒjʔ uqfmqtzstsdʒ q qLtsBtɛO⊙ʸO⊙sᴵ uqfmqtzO ststuq$O⊙dʒq dzO $tststɛHqdzOstsdzsʔ tjtɛʷdʒtsʔ, dzO⊐tsнqlsᴵ dʒtsнlzdz uqfmqtzO

ʉtststsdzsfmqjlsᴵ ststuq$q ə tjtɛʷdʒtslzGtsdzdzsʔ fmqjfmtsjdzj $tststɛHqdzOstsfmj$tststɛHqdzOstsO⊙dʒq ɻlstsʔ ststuq$q.

ʃ $tɛzGdzʷlsᴵ ujdz j uqfmqtzO q нjfmʸlsquqdz j нtɛtsBq, dzO⊐tsнqlsᴵ tjtɛqBdzqʞjdzqʷ uqfmjdz

ʏ qfmqtzstsəʳ j q Blz$əjdzdz j əʳ tɛO⊙ʸjdzqʷ

ɪ tsdzʃtsəʊksᴵ ʃ lzʧtslstɕjʁbdzʷlsᴵ lsjtɛdʒʃdzᶻ , fɲpʷLʊdzdzᶻ j fɲLʊʃʃfɲᴶʁ fɲljtjdzʃ ɯʃfɲdzʊ dzʊ⊐tsHʃlsᴵ ʕəʊHtɛʊks ʃ ʕlzʁ ɯʃfɲdzʊ əʳ ɯʃfɲdzʷlsᴵ ldzʊujdzʃʷ ɯʃfɲdzʦəʳ ⊐ əʳ tɛʊ|jdzʃʔ, fɲʦHjtɛ|ʊᴶ ʃ⊐ fɲljtjdzʃ.

ɪ tsdzʃtsəʊksᴵ fɲʸ ʧtɛʃldzʊʕʊdʒʃ Hjdzʃdʒʦfɲksʃ, tɛʊfɲʕdzʊHʳ əʊksᴵ dzʊkslztɛʊdzᶻ dzᶻ j ɯʃfɲdzʊ dzʊʧtɛʦfɲksʳ j dʒʦdzʦ|ʃlsjdzʃ.

ɪ tsdzʃtsəʊksᴵ fɲʸ dʒʊfɲʰ lsʊʁtsdʒ fɲʦfɲksʊ⊐dzʷlsᴵ ʧtɛʦstʃtstɛɯʃʃʃ ʃ tslsdzʦʰ jdzʃʷ.

R fɲʧtsdzʃtsəʊksᴵ ʁlzʕəʳ HdzʷtsʁtsLdzʊujdzʃʷ ɯʃfɲjdzʧtɛʃ Lʊʧʃfɲʃdʒʊksjdʒʊksʃujfɲʃ⊐ əʳ tɛʊ|jdzʃʔ, fɲʦfɲksʊ⊐dzʷlsᴵ ʁlzʕəjdzdzᶻ j əʳ tɛʊ|jdzʃʷ ʃ ʷtstɛdʒzdzᶻ , dzʊ⊐tsHʃlsᴵ ldzʊujdzʃʷ ʁlzʕəjdzdzᶻ⊐ əʳ tɛʊ|jdzʃʔ, tsfɲzʰ j fɲksʊdzʷʷ dzɸtsʁ⊐tsHʃdʒʳ j ʧtsHfɲksʊdzʦəʕʃ ʃ ʧtɛjtsʁtɛʊltsəʊdzʃʷ.

l ʊ⊐tsHʃlsᴵ dzɸʃLəjfɲksdzᶻ ʔ ʕtsdʒʧtsdzɸjdzʦtɛʊ⊐jdzfɲksəʊ

t j ʰ j dzʃj lsjʕfɲkstsəʳ ⊐ lʊHʊu

t j ʰ ʊksᴵ dʒdzʦgtsʰ ʊgtsəʳ j lsjʕfɲkstsəʳ j lʊHʊuʃ ʊtɛʃʷdzjlsʃujfɲʃʃdʒfɲʧtsfɲʁtsdʒ

t j ʰ ʊksᴵ lʊHʊuʃ, fɲpʷLʊdzdzᶻ j fɲ tslsdzʦʰ jdzʃjdʒ, ʧtɛtsʧtstɛɯʃtsdzʊdzdzʦfɲksᴵʁ əjdzʃuʃdzᶻ, ʧtɛtsujdzʦʊdʒʃ, tɛjʰ ʊksᴵ lstɛʃ tsfɲdzʦədzᶻ j lʊHʊuʃ dzʊHtɛtsʁʃ ʃ ʧtɛtsujdzʦᴶ .

t j ʰ ʊksᴵ lʊHʊuʃ, fɲʦHjtɛ|ʊᴶ ʃj lʊəʃfɲʃdʒtsfɲksʃ, fɲpʷLᴶ əʊᶠʰ ʃj əjdzʃuʃdzᶻ : fʕtstɛtsfɲksᴵ, ətɛjdʒʷ, tɛʊfɲfɲksts^wdzʃj, ujdzʊ ʕtsdzʃujfɲksəts, fɲkstsʃdʒtsfɲksᴵ, ʧtɛtsʃLətsHʃlsjdzdzʦfɲksᴵ, ətɛjdʒʷ, tsʁʲʸdʒ tɛʊʁtslsᴶ , ʃfɲʧtsdzʃtszʷ ʊɛdzʷʷdzjlsʃujfʕʃj Hjʔfɲksəʃʷ, tsujdzʁz, ʧtɛʃʕʃHʕz, ʧtsdzʃtsəʊksᴵ fɲʸ jHʃdzʃuʊdʒʃ ʃLdzjtɛjdzʃʷ fɲkstslsəjlsfɲksəzᴶʁ ʰ ʃ⊐ əjdzʃuʃdzᶻ.

ʜtsfɲksʊdzʷlsᴵ ʁlzʕəjdzdzᶻ j əʳ tɛʊ|jdzʃʷ ʧtsɛzfɲdzʦəʃᴶʁ lʊHʊuʃ.

R lədjʕʊksᴵ ʃdzʷtstɛdʒʊuʃᴶʁ , ʧtɛjHfɲksʊdjdzdzᶻ ə lsʊʁdzʃuʊ⊐, dzʊ dzʃdjʔdzʦʔ, fɲkstsdzʁuʊkstsʔ ʃdzʃ ʕtɛzʗtsətsʔ Hʃʊʗtɛʊdʒʃʒʊ⊐, ʃdzɸjtɛʧtɛjlsʃtɛtsəʊksᴵ ʧtɛjHfɲksʊdjdzdzᶻ j Hʊdzdzᶻ j, ʃfɲʧtsdzʃtsəʊksᴵ Hʊdzdzᶻ j ʧtɛʃtɛjʰ jdzʃʃ lʊHʊu.

ɪ tɛjHfɲksʊdzʷlsᴵ ʃdzʷtstɛdʒʊuʃᴶʁ fɲʧtsdzʦʰ ɹᴶʁ lsʊʁdzʃuʃ, dzʃdjʔdzʦʔ ʃ fɲkstsdzʁuʊkstsʔ Hʃʊʗtɛʊdʒʃʒ,

l ʊgdzʷHdzʊʷ gjtsdzjlstɛʃʷ

ɪ tɛʃətsHʃlsᴵ ʧtɛʃtɛʃdzjtɛʳ tsʁʲjʕlstsə tsʕtɛzʏʊᴶʰ jgtsdʒʃtɛʊ, ʃdzjᴶʁ ʰ ʃ⊐ ʷtstɛdʒzʃlzujdzdzᶻ⊐ gjtsdzjlstɛʃujfɲʃʃ⊐ ʧdzʦfɲʃʃ⊐ ʃ ʧtɛtsfɲkstɛʊdzfɲksəjdzdzᶻ⊐ ʷʃgʒztɛ, ʧtɛʃdzjtɛʳ tɛʊədzᶻ⊐ ʃ fɲʃjdʒdzjlstɛʃujdzᶻ⊐ ʷʃgʒztɛ.

R ltsʁtɛʊ|ʊksᴵ fɲ ʧtsdzʦʰ ɹᴶʁ ɯʃtɛʕlzdzʷ, dzʃdjʔʕʃ, lstɛʊdzfɲʧtstdsʃtɛʊ dzʊ dzjdzʃdzʦəʊdzdzʦʔ ʃ ʕdzjlsɯʊkstsʔ ʁlzdʒʊgj ʃLzujdzdzᶻ j ʧdzʦfɲʕʃj gjtsdzjlstɛʃujfɲʕʃj ʷʃgʒztɛʳ ʃ ʕtsdzʷʃgʒztɛʊujʃ, fɲʃdʒdzjlstɛʃujdzᶻ j ʷʃgʒztɛʳ .

ɪ tsdzʃtsəʊksᴵ fɲʸ gjtsdzjlstɛʃujfɲʕʃdʒʃ ʧtsdzᶻʷlsʃʷdzʃ: tɛʊjdzfɲksəts ʷʃgʒztɛ, fɲʃdʒdzjlstɛʃʷ, ʃfɲʧtsdzʃtsəʊksᴵ lsjtɛdʒʃdzʦdzʦgʃᴶʁ , fɲpʷLʊdzdzᴶʁ fɲ fɲʃdʒdzjlstɛʃjʔ: tsfɲ fɲʃdʒdzjlstɛʃʃ, ujdzʦtɛfɲʃdʒdzjlstɛʃʃ.

l ʊ⊐tsHʃlsᴵ əjdzʃuʃdzᶻ lzgdzʦəʃʃdʒtɛjdzʃjdʒ fɲ ʧtsdʒʦʰ ɹᴶʁ lstɛʊdzfɲʧtstɛdsʃtɛʊ fɲkstɛtsʃlsᴵ lzgdzᶻ lʊHʊdzdzʦʔ əjdzʃuʃdzᶻ , ʧtsdzʃtsəʊksᴵ fɲʸ ʧtɛʃ tɛjʰ jdzʃʃ lʊHʊuʃgtɛʊHlzfɲdzʦʔ

dzjtɑtsʔ lzɢdɑtsɔ, tɑƟfɳʃtsLdɑƵɔƟsᴵ dɑƵƟujtɑtsjɤlƟ⊐tsfɳstɕʳ ʔ, tʃtɕʷdʒsʔ, tɑƟLɔʸtɑdɑzlsʳ ʔ cʃlszʃtsʔ lzɢdz .

Jᴵʳ ʯcʃfɳgzʷlsᴵ нdɑcʃdɑz dɑtsdʒƟdɑtsʔ, tʃjtɑcʃdʒjlstɕ dʒdɑtsɢʈtslzɢʈtsdz dɑʃʕƟ, tʃtsdzʟtsɔƟsᴵ fɳʸ jнcʃdɑʃʯƟdʒʃ cʃʟdʒjtɑjdɑʃʷ нdɑcʃdz , ɔʳ tɑƟɤƟsᴵ tsнdɑʃ jнcʃdɑʃʯʳ cʃʟdʒjtɑjdɑʃʷ нdɑcʃdz ʯjtɑjʟ нtɑdzɢʈjcʃj .

lƟ⊐tsнcʃlsᴵ, cʃfɳʃtsdzʟlzʷ ʯjtɑlsʸɤlƟdz jcʃdɑɳstɑzdʒjdɑsʳ , tɑƟfɳɳstsʷdɑʃʷ: dʒjɤнlz нɔlzdʒʷ lstsʯʕƟdʒʃ, tslslstsʯʕcʃ нts tʃtɕʷdʒtsʔ, нdɑcʃdɑz tʃlzlscʃ dɑƵƟʕɔƟнtɑƟsdɑtsʔ fɳjlsʕj .

Jᴵʳ ʯcʃfɳgzʷlsᴵ tʃdɑtsʰ Ɵнᴵ ᵂʷcʃɢʈzʈtɕ, fɳtsfɳsƟɔdjɑdɑzʷ⊐ cʃʟ tʃtɕʷdʒtslzɢʈtsdz dɑʃʕtsɔ, cʃfɳʃtsdzʟtsɔƟsᴵ tɑƟɔʟʙcʃjdɑʃj dɑƵƟ tʃtɕʷdʒtslzɢʈtsdz dɑʃʕcʃ, dɑƵƟ tɑƟɔdzʷj ᵂʷcʃɢʈzʈtɕʳ , нtsfɳstɑƟɑʃɔƟdɑʃj нts tʃtɕʷdʒtslzɢʈtsdz dɑʃʕƟ, tʃtsdzʟtsɔƟsᴵ fɳʸ tsfɳɑtsɔdzʷ dʒʃ jнcʃdɑʃʯƟdʒʃ cʃʟdʒjtɑjdɑʃʷ tʃdɑtsʰ Ɵнcʃ, ɔʳ tɑƟɤƟsᴵ tsнdɑʃ jнcʃdɑʃʯʳ cʃʟdʒjtɑjdɑʃʷ tʃdɑtsʰ Ɵнcʃ ʯjtɑjʟ нtɑdzɢʈjcʃj .

tƟfɳʃtsLdɑƵɔƟsᴵ dɑƵƟ dʒtsнjdzʷ⊐ cʃ cʃʟtsʙtɑƟɤjdɑʃʷ⊐ tʃcʃtɑƟdʒʃнlz, ʕtsdɑzfɳ ʯcʃdɑcʃdɑʙtɕ, cʃfɳʃtsdzʟtsɔƟsᴵ lsjtɑcʃdʒʃdɑtsdɑtsɢʈcʃᵡ : ɔjtɕʰ cʃdɑƵƟ tɑjʙtɑts, ɢʈtɑƟdz, tsfɳɑtsɔƟdɑʃj, tɑƟLɔʸtɑlsʕƟ)

ʀ ʟtsʙtɑƟɤƟsᴵ dɑƵƟʕdjlsʯƟstsʔ ʙlzdʒƟɢj tʃtɕʷdʒtslzɢʈtsdz dzʔ tʃƟɑƟdɑdjdjtʃcʃtʃjн.

Jᴵʳ ʯcʃfɳgzʷlsᴵ tsʙʲʸdʒtʃtɕʷdʒtslzɢʈtsdz dɑtsɢʈts tʃƟɑƟdɑdjdjtʃcʃtʃjнƟ, ʕlzʙƟ, tʃtsdzʟtsɔƟsᴵ fɳʸ tsfɳɑtsɔdzʷ dʒʃ jнcʃdɑʃʯƟdʒʃ cʃʟdʒjtɑjdɑʃʷ tsʙʲʸdʒƟ)

tjʰƟsᴵ djfɳgtsɤdzʷ j ʟƟнƟʯcʃ dɑƵƟ dɑƵ⊐tsɤнjdɑʃj ɢjtsdjlstɕcʃʯjfɳʕcʃ⊐ ɔjdɑcʃʯcʃdzɔ tʃtɑƟʕlscʃʯjfɳʕcʃ⊐ fɳgʃlslzƟʯcʃʷ⊐.

Отчет о результатах самообследования
5 класс

№ п/п	Наименование образовательной программы	Количество участников	Количество победителей	Количество призеров	Ссылка на размещение информации (публичный адрес)
1	Олимпиада по математике	43	1		https://m.edsoo.ru/7f4131ce
2	Олимпиада по русскому языку	12		2	https://m.edsoo.ru/7f4131ce
3	Конкурс чтецов	48	1		https://m.edsoo.ru/7f4131ce
4	Олимпиада по окружающему миру	10		1	https://m.edsoo.ru/7f4131ce
5	Спортивные соревнования	38	1		https://m.edsoo.ru/7f4131ce
6	Олимпиада по английскому языку	9		1	https://m.edsoo.ru/7f4131ce
7	Творческий конкурс	10	1		https://m.edsoo.ru/7f4131ce
Итого		170	4	4	

6 ş ∫ ф ч ч

- tƒ/tƒ	ı Ot∫dşjdʌtsɔ Otʌ∫j teO.ч jdʌtsɔ d∫ lsjdʒ tƒteтsc?teOdşdʒ	ş tsdʌ∫ıjfhşɔ ts ıtO?hşɔ			Ş dşjŞlsteтsdʌdž j (ıtd∫w teтsɔr j) tsвteO.tsɔ Olsjdž dž j teşjfhşteфrj
		J fıjc?ts	ş tsdʌlsteтsdž dž j teO.вtslsr	ı teO?lsd∫ıjfhş d∫j teO.вtslsr	
1	ı Otslžteдdž dž j ıtd∫fıdžO	30	1		ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
2	ı Oc?dwнdʌOv cj tsdʒjlsteд∫w. ı tєwdʒr j dʌO tƒdʌsfhş tsfhjsd∫	7			ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
3	∫ teтsвd∫	32	1	1	ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
4	ı Oc?dwнdʌOv cj tsdʒjlsteд∫w. ч d∫dʒdʒjlsteд∫w	6		1	ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
5	J r teO.ıjdʌ∫w fıjвlž.ɔOdʒ∫	6			ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
6	ı Oc?dwнdʌOv cj tsdʒjlsteд∫w. ʌ d∫c?ždтєr dʌO tƒdʌsfhş tsfhjsd∫	14	1	1	ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
7	ı tsdʌsɤ d∫lsjdždž j d∫ tslsteд∫ıtOsjdždž j ıtd∫fıdžO	40	1		ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
8	ı tejнfhjsOɔdjdʌ∫j нOdʌdž ⊐	6		1	ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
9	ı Oc?dwнdʌOv cj tsdʒjlsteд∫w. ʌ d∫c?ždтєr ɔ tƒteтsfhjsteOdʌfhşɔj	9		1	ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
10	ı tsɔlsteтejdʌ∫j , tsвtsвh j dʌ∫j , fıj∫fhjsjdʒOsd∫ɹ.Oıd∫w	20	1		ɹ d∫вdʌ∫tslsj Ş.OʌʃŞ https://m.edsoo.ru/7f414736
∫ ɹ ž ∫ ∫ Ş ∫ ∫ R Y ∫ ч ?J ∫ Y ф ч ∫ J ı ∫ ı t ∫ ɹ t ф ∫ ∫ ∫		170	5	5	

ɨ ʃ ʊ ʈ ʃ ʏ ɭ ʃ ɾ ɨ ʃ ɸ ɻ ʁ ʈ ʃ ɹ ɸ ɻ ʁ ɾ

5 ʂ ʃ ɸ ʉ ʉ

- ʧ/ʧ	ʊ j d ʒ O l z t ɑ t s ʕ Θ	ʂ t s d ʞ ʃ ʯ j f ɧ ʂ ɔ t s ʯ C ɧ ɧ ʂ ɔ			ɾ C l s O ɕ ɻ l z ʯ j d ʞ ʃ ʷ	ʕ d j ʕ ɬ s t ɑ t s d ʞ ʑ j ʯ ɕ ʃ ʷ t ɑ t s ɔ ʳ j t s ʁ t ɑ O ɻ t s ɔ O ɬ s j d z d ʑ j t ɕ j f ɧ ɣ t ɑ f ɲ
		ɹ l f ɲ j c ɑ t s	ʂ t s d ɬ s t ɑ t s d z d ʑ j t ɑ O ʁ t s l s ʳ	ɨ t ɑ C ʕ ɬ s ʃ ʯ j f ɧ ʂ ɕ j t ɑ O ʁ t s l s ʳ		
1	ɾ j f ʷ ɣ l s ʃ ʯ ʞ z C ʸ f ɯ ʃ f ɬ ʂ j d ʒ O f ɲ ʯ ɕ ʃ f ɯ ɕ j d ʞ ʃ ʷ . ʈ ʷ H d ʞ C ɬ s l z t ɑ O ɻ z d ʑ ⊐ ʯ ɕ ʃ f ɲ j d z	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0cc0c
2	ɾ j f ʷ ɣ l s ʃ ʯ ʞ z C ʸ f ɯ ʃ f ɬ ʂ j d ʒ O f ɲ ʯ ɕ ʃ f ɯ ɕ j d ʞ ʃ ʷ . ʈ ʷ H d ʞ C ɬ s l z t ɑ O ɻ z d ʑ ⊐ ʯ ɕ ʃ f ɲ j d z	1				
3	ɭ C ɬ s l z t ɑ O ɻ z d ʑ ʔ t ɕ ʷ H . ʏ ɕ ʃ f ɯ ɣ t ʂ 0	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0cafe
4	ɭ C ɬ s l z t ɑ O ɻ z d ʑ ʔ t ɕ ʷ H . ʏ ɕ ʃ f ɯ ɣ t ʂ 0	1				
5	ɭ C ɬ s l z t ɑ O ɻ z d ʑ j ʯ ɕ ʃ f ɯ ɣ t O ɔ d ʞ O ʕ ɬ s t s t ɑ ɨ ɕ ʃ d ʞ C ɬ s ɔ d ʂ ʔ ʧ t ɕ ʷ d ʒ ʂ ʔ	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0e0fc
6	ɭ C ɬ s l z t ɑ O ɻ z d ʑ j ʯ ɕ ʃ f ɯ ɣ t O ɔ d ʞ O ʕ ɬ s t s t ɑ ɨ ɕ ʃ d ʞ C ɬ s ɔ d ʂ ʔ ʧ t ɕ ʷ d ʒ ʂ ʔ	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0e2a0
7	ɭ C ɬ s l z t ɑ O ɻ z d ʑ j ʯ ɕ ʃ f ɯ ɣ t O ɔ d ʞ O ʕ ɬ s t s t ɑ ɨ ɕ ʃ d ʞ C ɬ s ɔ d ʂ ʔ ʧ t ɕ ʷ d ʒ ʂ ʔ	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0e426
8	ʉ t ɑ ɔ d ʑ j d ʞ ʃ ɟ , t s ʕ t ɑ z ɔ d ʑ j d ʞ ʃ ɟ d ʞ C ɬ s l z t ɑ O ɻ z d ʑ ⊐ ʯ ɕ ʃ f ɲ j d z	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0ce32
9	ʉ t ɑ ɔ d ʑ j d ʞ ʃ ɟ , t s ʕ t ɑ z ɔ d ʑ j d ʞ ʃ ɟ d ʞ C ɬ s l z t ɑ O ɻ z d ʑ ⊐ ʯ ɕ ʃ f ɲ j d z	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0cf54
10	ʉ t ɑ ɔ d ʑ j d ʞ ʃ ɟ , t s ʕ t ɑ z ɔ d ʑ j d ʞ ʃ ɟ d ʞ C ɬ s l z t ɑ O ɻ z d ʑ ⊐ ʯ ɕ ʃ f ɲ j d z	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0d300
11	ʉ t ɑ ɔ d ʑ j d ʞ ʃ ɟ , t s ʕ t ɑ z ɔ d ʑ j d ʞ ʃ ɟ d ʞ C ɬ s l z t ɑ O ɻ z d ʑ ⊐ ʯ ɕ ʃ f ɲ j d z	1				ɹ ɕ ʃ ʁ ɔ d ʞ ʃ t s l ʂ j ʕ Θ ʎ ∫ ʂ https://m.edsoo.ru/f2a0d440

12	ʉ tɑɔ∋ɖʑiɖʑʃj , ts∑tɑʒɔɖʑiɖʑʃj ɖʑɔlsʒtɑɔɒtʒ ɖʒ ⊐ ʋʃʃfɲjɖʒ	1				
13	ɸ tɑɔʃʷ ɖʒjlsɔʃʋjfɲʃɔʃj ʜjʔfɲʃɔɔʃʷ fŋ ɖʑɔlsʒtɑɔɒtʒ ɖʒ. ɖʒʃ ʋʃʃfɲɟɔɒɔɖʒʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a0eaca
14	ɸ tɑɔʃʷ ɖʒjlsɔʃʋjfɲʃɔʃj ʜjʔfɲʃɔɔʃʷ fŋ ɖʑɔlsʒtɑɔɒtʒ ɖʒ. ɖʒʃ ʋʃʃfɲɟɔɒɔɖʒʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a0f5ba
15	ɸ tɑɔʃʷ ɖʒjlsɔʃʋjfɲʃɔʃj ʜjʔfɲʃɔɔʃʷ fŋ ɖʑɔlsʒtɑɔɒtʒ ɖʒ. ɖʒʃ ʋʃʃfɲɟɔɒɔɖʒʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a0f704
16	ɸ tɑɔʃʷ ɖʒjlsɔʃʋjfɲʃɔʃj ʜjʔfɲʃɔɔʃʷ fŋ ɖʑɔlsʒtɑɔɒtʒ ɖʒ. ɖʒʃ ʋʃʃfɲɟɔɒɔɖʒʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a0fd8a
17	ɸ tɑɔʃʷ ɖʒjlsɔʃʋjfɲʃɔʃj ʜjʔfɲʃɔɔʃʷ fŋ ɖʑɔlsʒtɑɔɒtʒ ɖʒ. ɖʒʃ ʋʃʃfɲɟɔɒɔɖʒʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a1015e
18	ɸ tɑɔʃʷ ɖʒjlsɔʃʋjfɲʃɔʃj ʜjʔfɲʃɔɔʃʷ fŋ ɖʑɔlsʒtɑɔɒtʒ ɖʒ. ɖʒʃ ʋʃʃfɲɟɔɒɔɖʒʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a10c3a
19	ɸ tɑɔʃʷ ɖʒjlsɔʃʋjfɲʃɔʃj ʜjʔfɲʃɔɔʃʷ fŋ ɖʑɔlsʒtɑɔɒtʒ ɖʒ. ɖʒʃ ʋʃʃfɲɟɔɒɔɖʒʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a10da2
20	ʉ ɘtsʔfɲʃɔƟɔɖʑɖʑʷ ʧʃtɑɔʃ fɲɟtsʏɻjɖʑʃɔʃ ɔʃ ʅɔɖʒɔɖʑtsʏɻjɖʑʃɔʃ, fɲɟtsʔfɲʃɔƟjʜɔʃɖʑʃʋʳ ʧʃtɑɔʃ ʅɔɖʒɔɖʑtsʏɻjɖʑʃɔʃ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a104ec
21	ʉ ɘtsʔfɲʃɔƟɔɖʑɖʑʷ ʧʃtɑɔʃ fɲɟtsʏɻjɖʑʃɔʃ ɔʃ ʅɔɖʒɔɖʑtsʏɻjɖʑʃɔʃ, fɲɟtsʔfɲʃɔƟjʜɔʃɖʑʃʋʳ ʧʃtɑɔʃ ʅɔɖʒɔɖʑtsʏɻjɖʑʃɔʃ	1				
22	ɻ jʇtɕjɖʒjfɲʃɔʃlsjɖʒɖʑsjʇɔʃ fɲʂʋjlsƟsjɖʒɖʑsj fɲɟtsʔfɲʃɔƟfɲɟtsʏɻjɖʑʃʷ ɔʃ ʅɔɖʒɔɖʑtsʏɻjɖʑʃʷ, tɑɔƟfɲɟʇtɕjʜjɖʑʃlsjɖʒɖʑsj fɲɟtsʔfɲʃɔts ʅɔɖʒɔɖʑtsʏɻjɖʑʃʷ	1				ɹ ɔʃʙɔɖʑʃtslsjʕƟʎ∫ ʂ https://m.edsoo.ru/f2a0ef3e
23	ɻ jʇtɕjɖʒjfɲʃɔʃlsjɖʒɖʑsjʇɔʃ fɲʂʋjlsƟsjɖʒɖʑsj fɲɟtsʔfɲʃɔƟfɲɟtsʏɻjɖʑʃʷ ɔʃ ʅɔɖʒɔɖʑtsʏɻjɖʑʃʷ,	1				

	tɶϴfɳʃtɕjʜjdzɭlsjdžʤsjfɱɒts?fɳʃɘɒts lzdʒʤsɤɭjdzɬʷ					
24	ɻjtɕjdʒjfɳʃsɭlsjdžʤsjɭʃfɳʃʂɥjʃlsϴʃsjdžʤsj fɱɒts?fɳʃɘϴfɳʤʤsɤɭjdzɬʷɭʃlzdʒʤsɤɭjdzɬʷ, tɶϴfɳʃtɕjʜjdzɭlsjdžʤsjfɱɒts?fɳʃɘɒts lzdʒʤsɤɭjdzɬʷ	1				
25	ɾjdzɭlsjdzɭɭʃʕtɶϴsdžjɥɥʃfɳʤϴ tɶϴʟʤsɤɭjdzɬjɥɥʃfɳʤϴdzϴdʒʤsɤɭʃlsjdzɭ	1				
26	ɾjdzɭlsjdzɭɭʃʕtɶϴsdžjɥɥʃfɳʤϴ tɶϴʟʤsɤɭjdzɬjɥɥʃfɳʤϴdzϴdʒʤsɤɭʃlsjdzɭ	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a116b2
27	ɾjdzɭlsjdzɭɭʃʕtɶϴsdžjɥɥʃfɳʤϴ tɶϴʟʤsɤɭjdzɬjɥɥʃfɳʤϴdzϴdʒʤsɤɭʃlsjdzɭ	1				
28	ɾjdʑdzɬjɥɥʃfɲtsfɳʃsϴsʕtsdʒ	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a1116c
29	ɾjdʑdzɬjɥɥʃfɲtsfɳʃsϴsʕtsdʒ	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a114fa
30	ɻtɶsfɳʃʳjʃɥʃfɱsfɳʃsϴɘdžjɥɥʃfɳʤϴ	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a11a90
31	ɻtɶsfɳʃʳjʃɥʃfɱsfɳʃsϴɘdžjɥɥʃfɳʤϴ	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a11bb2
32	ɻtɶʃʟdzϴʕʃɥʃʜjdzɭʃdʒʦsfɳʃsʃdzϴ2, 5, 10, 3, 9	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a11806
33	ɻtɶʃʟdzϴʕʃɥʃʜjdzɭʃdʒʦsfɳʃsʃdzϴ2, 5, 10, 3, 9	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a1196e
34	ʏʃfɳʤʦɘʳjɘʳtɶϴɬjdzɬʷ;tʃtstɕʷʜtsʕ ʜjʔfɳʃɘʃʔ	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a11f18
35	ʏʃfɳʤʦɘʳjɘʳtɶϴɬjdzɬʷ;tʃtstɕʷʜtsʕ ʜjʔfɳʃɘʃʔ	1				ɹʃʁdzɭtslsjʕϴʎʃʂ https://m.edsoo.ru/f2a12080

36	ʏ ɕʃfɳɡƶtsɔʳ ʝ ɔʳ tɷʘɤɨ ʝ dʑɑʃʷ; tʃtstɕʷʜtsʕ ʜʝ ʔfɳʂɔɕʃʔ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a123fa
37	ʈ ʝʰ ʝ dʑɑʃʝ lsʝ ʕfɳʂtsɔʳ ⊐ʟϴʜʘɥ dʑʘɔfɳʝ ϴɷɑʃʷ dʒʝ lsɕʃɥʝ fɳʕɕʃʝ ʜʝ ʔfɳʂɔɕʃʷ, dʑʘ ʜɔɕʃɤɨ ʝ dʑɑʃʝ ɕ tʃtsʕɬzʃʕɕʃ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a0f894
38	ʈ ʝʰ ʝ dʑɑʃʝ lsʝ ʕfɳʂtsɔʳ ⊐ʟϴʜʘɥ dʑʘɔfɳʝ ϴɷɑʃʷ dʒʝ lsɕʃɥʝ fɳʕɕʃʝ ʜʝ ʔfɳʂɔɕʃʷ, dʑʘ ʜɔɕʃɤɨ ʝ dʑɑʃʝ ɕ tʃtsʕɬzʃʕɕʃ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a0f9fc
39	ʈ ʝʰ ʝ dʑɑʃʝ lsʝ ʕfɳʂtsɔʳ ⊐ʟϴʜʘɥ dʑʘɔfɳʝ ϴɷɑʃʷ dʒʝ lsɕʃɥʝ fɳʕɕʃʝ ʜʝ ʔfɳʂɔɕʃʷ, dʑʘ ʜɔɕʃɤɨ ʝ dʑɑʃʝ ɕ tʃtsʕɬzʃʕɕʃ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a121a2
40	ʈ ʝʰ ʝ dʑɑʃʝ lsʝ ʕfɳʂtsɔʳ ⊐ʟϴʜʘɥ dʑʘɔfɳʝ ϴɷɑʃʷ dʒʝ lsɕʃɥʝ fɳʕɕʃʝ ʜʝ ʔfɳʂɔɕʃʷ, dʑʘ ʜɔɕʃɤɨ ʝ dʑɑʃʝ ɕ tʃtsʕɬzʃʕɕʃ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a12558
41	ʈ ʝʰ ʝ dʑɑʃʝ lsʝ ʕfɳʂtsɔʳ ⊐ʟϴʜʘɥ dʑʘɔfɳʝ ϴɷɑʃʷ dʒʝ lsɕʃɥʝ fɳʕɕʃʝ ʜʝ ʔfɳʂɔɕʃʷ, dʑʘ ʜɔɕʃɤɨ ʝ dʑɑʃʝ ɕ tʃtsʕɬzʃʕɕʃ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a12832
42	ʈ ʝʰ ʝ dʑɑʃʝ lsʝ ʕfɳʂtsɔʳ ⊐ʟϴʜʘɥ dʑʘɔfɳʝ ϴɷɑʃʷ dʒʝ lsɕʃɥʝ fɳʕɕʃʝ ʜʝ ʔfɳʂɔɕʃʷ, dʑʘ ʜɔɕʃɤɨ ʝ dʑɑʃʝ ɕ tʃtsʕɬzʃʕɕʃ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a12990
43	ʂ tsɕƶtɕtsɕƶ dʑϴʏ tɷʘɜtslsϴtʃts lsʝ dʒʝ "ɭ ϴlsƶtɷʘɥƶ dƶ ʝ ɥɕʃɳɡƶϴɕʃ dƶɕdƶ"	1	1			ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a12cba
44	ʊ tsɕɬʕϴ, tʃtɕʷ dʒϴʏ, tslstɕʝ ʟtsʕ, dƶzɥ ʃ tsɕʒϴɥʑʏ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a0d54e
45	ʀ ʟdʒʝ tɷʝ dʑɑʃʝ ʜdʑɑʃƶ tslstɕʝ ʟʕϴ, dʒʝ lstɕʃɥʝ fɳʕɕʃʝ ʝ ʜɕʃɕdʑɑʃɥʳ ɕʃʟdʒʝ tɷʝ dʑɑʃʷ ʜdʑɑʃƶ	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ https://m.edsoo.ru/f2a0daee
46	ʀ ʟdʒʝ tɷʝ dʑɑʃʝ ʜdʑɑʃƶ tslstɕʝ ʟʕϴ,	1				ɻ ɕʃʙɕʃɕtslsʝ ʕϴʎ ʃ ʂ

	dʒ lstɑʃʰuj fɧʃqj j нʃdʑʃʰuʳ qʟdʒ tɕj dʑʃʷ нdʑʃdʑ					https://m.edsoo.ru/f2a0df3a
47	ʃ ʕtɑlzʏ dʑsfɧjsᵍ qʃ ʕtɑlzɢᶜ	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a0d684
48	ʃ ʕtɑlzʏ dʑsfɧjsᵍ qʃ ʕtɑlzɢᶜ	1				
49	ɬ tɑƟʕlsʃʰuj fɧʃƟʸ tɑƟʙtslsƟʧts lsj dʒj "ɬ tsfɧjstɑtsj dʑʃj ʟzʟtstɑƟʃqʟ tsʕtɑlzʏ dʑsfɧjsj ʔ"	1		1		ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a0d7e2
50	ʊ ɢtsdzɬ tɕʷ dʒtsʔ, tsfɧjstɕʳ ʔ, lsʐʧtsʔ qʃ tɑƟʟəʸ tɑdʑlsʳ ʔ ʟzɢdʑ	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a1302a
51	ʊ ɢtsdzɬ tɕʷ dʒtsʔ, tsfɧjstɕʳ ʔ, lsʐʧtsʔ qʃ tɑƟʟəʸ tɑdʑlsʳ ʔ ʟzɢdʑ	1				
52	ʀ ʟdʒ tɕj dʑʃj ʟzɢdʑtsə	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a1319c
53	ʀ ʟdʒ tɕj dʑʃj ʟzɢdʑtsə	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a132fa
54	ʀ ʟdʒ tɕj dʑʃj ʟzɢdʑtsə	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a13476
55	ɬ tɑƟʕlsʃʰuj fɧʃƟʸ tɑƟʙtslsƟʧts lsj dʒj "ɬ tsfɧjstɑtsj dʑʃj ʟzɢdʑtsə"	1		1		ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a13606
56	ɾ tɑtsʙᴵ . ɬ tɑƟəqʃdʑ dʑ j qʃ dʑj tʃ tɑƟəqʃdʑ dʑ j нtɑtsʙqʃ	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a13764
57	ɾ tɑtsʙᴵ . ɬ tɑƟəqʃdʑ dʑ j qʃ dʑj tʃ tɑƟəqʃdʑ dʑ j нtɑtsʙqʃ	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a13c8c
58	ɾ tɑtsʙᴵ . ɬ tɑƟəqʃdʑ dʑ j qʃ dʑj tʃ tɑƟəqʃdʑ dʑ j нtɑtsʙqʃ	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a14146
59	ɾ tɑtsʙᴵ . ɬ tɑƟəqʃdʑ dʑ j qʃ dʑj tʃ tɑƟəqʃdʑ dʑ j нtɑtsʙqʃ	1				ɹ qʃʙdʑʃtslsj ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a153f2

60	[tɑsʙᴵ . ɻ tɑ⊙ɔɑ[ɑz ɑz ʝ ɑ[ɑɸʝ[tɑ⊙ɔɑ[ɑz ɑz ʝ ʜtɑsʙɑ[	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a15582
61	∫ fɑɑ[ɑsɔɑɑsʝ fɑɔts÷ fɑʝɔts ʜtɑsʙɑ[	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a143e4
62	∫ fɑɑ[ɑsɔɑɑsʝ fɑɔts÷ fɑʝɔts ʜtɑsʙɑ[	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a1451a
63	∫ fɑɑ[ɑsɔɑɑsʝ fɑɔts÷ fɑʝɔts ʜtɑsʙɑ[	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a1463c
64	∫ fɑɑ[ɑsɔɑɑsʝ fɑɔts÷ fɑʝɔts ʜtɑsʙɑ[	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a1475e
65	∫ fɑɑ[ɑsɔɑɑsʝ fɑɔts÷ fɑʝɔts ʜtɑsʙɑ[	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a14c90
66	∫ fɑɑ[ɑsɔɑɑsʝ fɑɔts÷ fɑʝɔts ʜtɑsʙɑ[	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a14de4
67	∫ fɑɑ[ɑsɔɑɑsʝ fɑɔts÷ fɑʝɔts ʜtɑsʙɑ[	1				
68	ʜ tɑ⊙ɔɑɸʝɑɑ[ʝ ʜtɑsʙʝ÷	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a14f74
69	ʜ tɑ⊙ɔɑɸʝɑɑ[ʝ ʜtɑsʙʝ÷	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a151f4
70	ʜ tɑ⊙ɔɑɸʝɑɑ[ʝ ʜtɑsʙʝ÷	1				
71	ʜ tɑ⊙ɔɑɸʝɑɑ[ʝ ʜtɑsʙʝ÷	1				
72	ʜ ɑɑsʏʝɑɑ[ʝ ɑ[ɔʳ ʋɑ[lsɔɑɑ[ʝ tsʙʳ ʕɑɑsɔʝɑɑzʂ ⊐ ʜtɑsʙʝ÷	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a17cc4
73	ʜ ɑɑsʏʝɑɑ[ʝ ɑ[ɔʳ ʋɑ[lsɔɑɑ[ʝ tsʙʳ ʕɑɑsɔʝɑɑzʂ ⊐ ʜtɑsʙʝ÷	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a17e54
74	ʜ ɑɑsʏʝɑɑ[ʝ ɑ[ɔʳ ʋɑ[lsɔɑɑ[ʝ tsʙʳ ʕɑɑsɔʝɑɑzʂ ⊐ ʜtɑsʙʝ÷	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ https://m.edsoo.ru/f2a1802a
75	ʜ ɑɑsʏʝɑɑ[ʝ ɑ[ɔʳ ʋɑ[lsɔɑɑ[ʝ	1				ɻ ɑ[ʙɑɑ[tslsʝ ʕ⊖ʎ ∫ ʂ

	tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ					https://m.edsoo.ru/f2a181ce
76	ʉ ʤsʏʝ ʤʣʃʝ ɕʃ ɔʳ ʋɕʃlsʘʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1835e
77	ʉ ʤsʏʝ ʤʣʃʝ ɕʃ ɔʳ ʋɕʃlsʘʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ	1				
78	ʉ ʤsʏʝ ʤʣʃʝ ɕʃ ɔʳ ʋɕʃlsʘʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ	1				
79	ʉ ʤsʏʝ ʤʣʃʝ ɕʃ ɔʳ ʋɕʃlsʘʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ	1				
80	ʉ ʤʒʲʰ ʘʤʣʘʸ нtɕtsʙᴶ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1592e
81	ʉ ʤʒʲʰ ʘʤʣʘʸ нtɕtsʙᴶ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a15a5a
82	ʉ ʤʒʲʰ ʘʤʣʘʸ нtɕtsʙᴶ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a15b68
83	ʉ ʤʒʲʰ ʘʤʣʘʸ нtɕtsʙᴶ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a15e2e
84	ʊ ʤʒʤsʏʝ ʤʣʃʝ ɕʃ нʝ ʤʲʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ; ɔʟʘɕʃʤʒʤʣtsʙtɕʘsʤʣ ʝ нtɕtsʙɕʃ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a184e4
85	ʊ ʤʒʤsʏʝ ʤʣʃʝ ɕʃ нʝ ʤʲʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ; ɔʟʘɕʃʤʒʤʣtsʙtɕʘsʤʣ ʝ нtɕtsʙɕʃ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a18692
86	ʊ ʤʒʤsʏʝ ʤʣʃʝ ɕʃ нʝ ʤʲʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ; ɔʟʘɕʃʤʒʤʣtsʙtɕʘsʤʣ ʝ нtɕtsʙɕʃ	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a18a20
87	ʊ ʤʒʤsʏʝ ʤʣʃʝ ɕʃ нʝ ʤʲʤʣʃʝ tsʙʳ ʕɕʤsɔʝ ʤʣʒ ⊐ нtɕtsʙʝʔ;	1				ɹ ɕʃʙɕʤʣʃtslsʝ ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a18b56

	ɘLʘɑ[dʒʤtsʁtɕʘsdʑ ʝ нtɑtsʁɑ[					
88	ʊ dʒʤtsʏ ʝ dʑ[ʝ ɑ[нʝ dʑʝ dʑ[ʝ tsʁʳ ʕ dʤsɔʝ dʑdʑ ⊐ нtɑtsʁʝ ʔ; ɘLʘɑ[dʒʤtsʁtɕʘsdʑ ʝ нtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a19088
89	ʊ dʒʤtsʏ ʝ dʑ[ʝ ɑ[нʝ dʑʝ dʑ[ʝ tsʁʳ ʕ dʤsɔʝ dʑdʑ ⊐ нtɑtsʁʝ ʔ; ɘLʘɑ[dʒʤtsʁtɕʘsdʑ ʝ нtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a19560
90	ʊ dʒʤtsʏ ʝ dʑ[ʝ ɑ[нʝ dʑʝ dʑ[ʝ tsʁʳ ʕ dʤsɔʝ dʑdʑ ⊐ нtɑtsʁʝ ʔ; ɘLʘɑ[dʒʤtsʁtɕʘsdʑ ʝ нtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a196a0
91	ʊ dʒʤtsʏ ʝ dʑ[ʝ ɑ[нʝ dʑʝ dʑ[ʝ tsʁʳ ʕ dʤsɔʝ dʑdʑ ⊐ нtɑtsʁʝ ʔ; ɘLʘɑ[dʒʤtsʁtɕʘsdʑ ʝ нtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a198da
92	ʈ ʝʰ ʝ dʑ[ʝ lsʝ ʕfɲʂtsɔʳ ⊐ lʘнʘuʂ fɲʂнʝ tɕʏ ʘʲ ɑ[⊐ нtɑtsʁɑ[. ʃ fɲʤtsɔdʑ ʝ lʘнʘuɑ[dʑʘнtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a181ce
93	ʈ ʝʰ ʝ dʑ[ʝ lsʝ ʕfɲʂtsɔʳ ⊐ lʘнʘuʂ fɲʂнʝ tɕʏ ʘʲ ɑ[⊐ нtɑtsʁɑ[. ʃ fɲʤtsɔdʑ ʝ lʘнʘuɑ[dʑʘнtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a1835e
94	ʈ ʝʰ ʝ dʑ[ʝ lsʝ ʕfɲʂtsɔʳ ⊐ lʘнʘuʂ fɲʂнʝ tɕʏ ʘʲ ɑ[⊐ нtɑtsʁɑ[. ʃ fɲʤtsɔdʑ ʝ lʘнʘuɑ[dʑʘнtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a18c5a
95	ʈ ʝʰ ʝ dʑ[ʝ lsʝ ʕfɲʂtsɔʳ ⊐ lʘнʘuʂ fɲʂнʝ tɕʏ ʘʲ ɑ[⊐ нtɑtsʁɑ[. ʃ fɲʤtsɔdʑ ʝ lʘнʘuɑ[dʑʘнtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a18e76
96	ʈ ʝʰ ʝ dʑ[ʝ lsʝ ʕfɲʂtsɔʳ ⊐ lʘнʘuʂ fɲʂнʝ tɕʏ ʘʲ ɑ[⊐ нtɑtsʁɑ[. ʃ fɲʤtsɔdʑ ʝ lʘнʘuɑ[dʑʘнtɑtsʁɑ[	1				ɹ ɑ[ʁdʑ[tslsʝ ʕƟʎ ʃ ʂ https://m.edsoo.ru/f2a18f7a

97	t j h j dz ʃj lsj ʂ fŋʂtsəʳ ⊐ LƟHƟu ę fŋʂHj tɕʏ Ɵʲ ɕ⊐ Htɑtsʙɕ. ʃ fŋʤzsəʤ j LƟHƟuɕ dzƟHtɑtsʙɕ	1					ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a199f2
98	t j h j dz ʃj lsj ʂ fŋʂtsəʳ ⊐ LƟHƟu ę fŋʂHj tɕʏ Ɵʲ ɕ⊐ Htɑtsʙɕ. ʃ fŋʤzsəʤ j LƟHƟuɕ dzƟHtɑtsʙɕ	1					ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a19c2c
99	t j h j dz ʃj lsj ʂ fŋʂtsəʳ ⊐ LƟHƟu ę fŋʂHj tɕʏ Ɵʲ ɕ⊐ Htɑtsʙɕ. ʃ fŋʤzsəʤ j LƟHƟuɕ dzƟHtɑtsʙɕ	1					ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a1a1d6
100	t j h j dz ʃj lsj ʂ fŋʂtsəʳ ⊐ LƟHƟu ę fŋʂHj tɕʏ Ɵʲ ɕ⊐ Htɑtsʙɕ. ʃ fŋʤzsəʤ j LƟHƟuɕ dzƟHtɑtsʙɕ	1					ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a1a2ee
101	ɻ tɑɕʃdʒj dʑj dzɑʃj ʙlzʂə нdzʷ LƟʃɕʃfŋʃ dʒƟsj dʒƟsɕʃuj fŋʂɕ⊐ əʳ tɑƟʏ j dzɑʃʔ ɕʃ tʃtɕj нdʤsʏ j dzɑʃʔ	1					ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a1a3fc
102	ɻ tɑɕʃdʒj dʑj dzɑʃj ʙlzʂə нdzʷ LƟʃɕʃfŋʃ dʒƟsj dʒƟsɕʃuj fŋʂɕ⊐ əʳ tɑƟʏ j dzɑʃʔ ɕʃ tʃtɕj нdʤsʏ j dzɑʃʔ	1					
103	ʂ tsdʤtɑtsdz dzƟʸ tɑƟʙtslsƟtʃts lsj dʒj "ʃ ʙʳ ʂ dʤsəj dzdz j нtɑtsʙɕ"	1	1				ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a1a51e
104	ʃ dʤsɑtslzɑtsdz dzɑʃʂɕ. ʏ j lsʳ tɕʸ⊐lzɑtsdz dzɑʃʂ, tʃ tɕʷdʒtslzɑtsdz dzɑʃʂ, ʂəƟHtɑƟs	1					ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a16ae0
105	ʃ dʤsɑtslzɑtsdz dzɑʃʂɕ. ʏ j lsʳ tɕʸ⊐lzɑtsdz dzɑʃʂ, tʃ tɕʷdʒtslzɑtsdz dzɑʃʂ, ʂəƟHtɑƟs	1					ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a16c7a
106	ɻ tɑƟʂlsɕʃuj fŋʂƟʸ tɑƟʙtslsƟtʃts lsj dʒj "ɻ tsfŋʂtɑtsj dzɑʃj tʃ tɕʷdʒtslzɑtsdz dzɑʃʂƟfŋ	1		1			ɹ ɕʃʙdzɕtslsj ʂƟʎ ʃ ʂ https://m.edsoo.ru/f2a16e1e

	ʟʘʜʘʁʁʐ ʤʃ ɦɧʂtʁʂʤʘʁʃ ʁʘ ʤɻ ʁʃʤʂəʘʁʁʂʔ ʙʅʐʤʃʘɕɻ"					
107	ʊtɻʅʐɕtsʤ ʁʃʔ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a16194
108	ʊtɻʅʐɕtsʤ ʁʃʔ	1				
109	ɪ ʤʂʰ ʘʜᴵ ʃ ʧ ʝ tɶʃʤɻlstɕ ʧtɕʷʤʃtsʅʐɕtsʤ ʁʃʔʘʃ ʤʃʤʂɕtsʅʐɕtsʤ ʁʃʔtsə, ɦɧʃɦʃʘəʤɻʤʁʤʻ ⊐ ʃʟ ʧtɕʷʤʃtsʅʐɕtsʤ ʁʃʔtsə, ʝ ʜʃʤʁʃʰʳ ʃʟʤʃɻ tɻʤ ʁʃʷ ʧʃʤʂʰ ʘʜʃ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a16fe0
110	ɪ ʤʂʰ ʘʜᴵ ʃ ʧ ʝ tɶʃʤɻlstɕ ʧtɕʷʤʃtsʅʐɕtsʤ ʁʃʔʘʃ ʤʃʤʂɕtsʅʐɕtsʤ ʁʃʔtsə, ɦɧʃɦʃʘəʤɻʤʁʤʻ ⊐ ʃʟ ʧtɕʷʤʃtsʅʐɕtsʤ ʁʃʔtsə, ʝ ʜʃʤʁʃʰʳ ʃʟʤʃɻ tɻʤ ʁʃʷ ʧʃʤʂʰ ʘʜʃ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a17184
111	ɪ ʤʂʰ ʘʜᴵ ʃ ʧ ʝ tɶʃʤɻlstɕ ʧtɕʷʤʃtsʅʐɕtsʤ ʁʃʔʘʃ ʤʃʤʂɕtsʅʐɕtsʤ ʁʃʔtsə, ɦɧʃɦʃʘəʤɻʤʁʤʻ ⊐ ʃʟ ʧtɕʷʤʃtsʅʐɕtsʤ ʁʃʔtsə, ʝ ʜʃʤʁʃʰʳ ʃʟʤʃɻ tɻʤ ʁʃʷ ʧʃʤʂʰ ʘʜʃ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a17328
112	ɪ ʝ tɶʃʤɻlstɕ ʤʃʤʂɕtsʅʐɕtsʤ ʁʃʔʘ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1691e
113	ɪ ʝ tɶʃʤɻlstɕ ʤʃʤʂɕtsʅʐɕtsʤ ʁʃʔʘ	1				
114	ʃ ʝ ɦʲlsʃʏʧʁʘʷ ʟʘʃʃʃɦʰ ʜtɶsʙʝʔ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1b55e
115	ʃ ʝ ɦʲlsʃʏʧʁʘʷ ʟʘʃʃʃɦʰ ʜtɶsʙʝʔ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1b87e
116	ʃ ʝ ɦʲlsʃʏʧʁʘʷ ʟʘʃʃʃɦʰ ʜtɶsʙʝʔ	1				ɹ ʃʙʁʃtslɻʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1bcfc

117	ʜ tɑ⊃ɔdʑ dʑʃj ʜj fŋʸlsʃ ɥʦz ⊐ ʜtɑtsʙj ʔ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1c49a
118	ʜ tɑ⊃ɔdʑ dʑʃj ʜj fŋʸlsʃ ɥʦz ⊐ ʜtɑtsʙj ʔ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1c63e
119	ʜ tɑ⊃ɔdʑ dʑʃj ʜj fŋʸlsʃ ɥʦz ⊐ ʜtɑtsʙj ʔ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1cb02
120	ʜ tɑ⊃ɔdʑ dʑʃj ʜj fŋʸlsʃ ɥʦz ⊐ ʜtɑtsʙj ʔ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1cc2e
121	ʜ tɑ⊃ɔdʑ dʑʃj ʜj fŋʸlsʃ ɥʦz ⊐ ʜtɑtsʙj ʔ	1				
122	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1ce4a
123	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1cf62
124	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1d174
125	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1d516
126	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1d64c
127	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1d750
128	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1d85e
129	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1d962
130	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1da7a
131	ɾ j ʔ fŋʃɔʃʷ fŋʜj fŋʸlsʃ ɥʦz ʤʃ ʜtɑtsʙʷʤʃ	1				ɹ ʃʦʙɔdʑʃtslsj ʕΘʎ ʃ ʂ https://m.edsoo.ru/f2a1db88

132	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e01a
133	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e150
134	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e268
135	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e3da
136	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e4f2
137	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e4f2
138	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e5f6
139	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e704
140	ɾ ʝ ʔ fɳʂɔɕʷ fŋʜʝ fɳʸlsɕ∫ɥɕƶ ɕʒɕ∫ нtɑtsʙʷɕʒɕ∫	1				
141	∫ ʕtɑʒɕɕɖʝ ɑƙ[ʝ ʜʝ fɳʸlsɕ∫ɥɕƶ ⊐ нtɑtsʙ[ʔ	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1e826
142	∫ ʕtɑʒɕɕɖʝ ɑƙ[ʝ ʜʝ fɳʸlsɕ∫ɥɕƶ ⊐ нtɑtsʙ[ʔ	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1eb50
143	∫ ʕtɑʒɕɕɖʝ ɑƙ[ʝ ʜʝ fɳʸlsɕ∫ɥɕƶ ⊐ нtɑtsʙ[ʔ	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1ec68
144	∫ ʕtɑʒɕɕɖʝ ɑƙ[ʝ ʜʝ fɳʸlsɕ∫ɥɕƶ ⊐ нtɑtsʙ[ʔ	1				
145	t ʝʰ ʝ ɑƙ[ʝ lsʝ ʕfɳʂtsɔʳ ⊐ LƟнƟʊ ɳ fɳʂнʝ tɔɤ Ɵᶠ ɕ⊐ нtɑtsʙɕ. ∫ fɳʂtsɔɖƶ ʝ LƟнƟʊ[ɕ ɑƙƟнtɑtsʙɕ∫	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1ed8a
146	t ʝʰ ʝ ɑƙ[ʝ lsʝ ʕfɳʂtsɔʳ ⊐ LƟнƟʊ ɳ	1				ɹ ɕʙɕƙɕ[tslsʝ ʕƟʎ ∫ ʂ https://m.edsoo.ru/f2a1ef10

	fɱsʜjˌtɕʏ[ʘʱ ɖ[⊐ ʜtɑtsʙɖ[. ʃ fɳɖʑsɔdƶ ˌj ʟʘʜʘɯ[ɖ ɖʑʘʜtɑtsʙɖ[					
147	ʈ ˌjʰ ˌj dɑɖ[ˌj lsˌjˌʢfɱʂtsɔʳ ⊐ ʟʘʜʘɯˌ fɱsʜjˌtɕʏ[ʘʱ ɖ[⊐ ʜtɑtsʙɖ[. ʃ fɳɖʑsɔdƶ ˌj ʟʘʜʘɯ[ɖ ɖʑʘʜtɑtsʙɖ[	1				ɹ ɖ[ʙɖɑ[tslsˌjˌʢʘʎ ʃ ʂ https://m.edsoo.ru/f2a1f028
148	ʈ ˌjʰ ˌj dɑɖ[ˌj lsˌjˌʢfɱʂtsɔʳ ⊐ ʟʘʜʘɯˌ fɱsʜjˌtɕʏ[ʘʱ ɖ[⊐ ʜtɑtsʙɖ[. ʃ fɳɖʑsɔdƶ ˌj ʟʘʜʘɯ[ɖ ɖʑʘʜtɑtsʙɖ[	1				ɹ ɖ[ʙɖɑ[tslsˌjˌʢʘʎ ʃ ʂ https://m.edsoo.ru/f2a1f136
149	ʈ ˌjʰ ˌj dɑɖ[ˌj lsˌjˌʢfɱʂtsɔʳ ⊐ ʟʘʜʘɯˌ fɱsʜjˌtɕʏ[ʘʱ ɖ[⊐ ʜtɑtsʙɖ[. ʃ fɳɖʑsɔdƶ ˌj ʟʘʜʘɯ[ɖ ɖʑʘʜtɑtsʙɖ[	1				
150	ʈ ˌjʰ ˌj dɑɖ[ˌj lsˌjˌʢfɱʂtsɔʳ ⊐ ʟʘʜʘɯˌ fɱsʜjˌtɕʏ[ʘʱ ɖ[⊐ ʜtɑtsʙɖ[. ʃ fɳɖʑsɔdƶ ˌj ʟʘʜʘɯ[ɖ ɖʑʘʜtɑtsʙɖ[	1				
151	ʂ tscɖʑtɑtsdƶ ɖɑʘʸ tɑʘʁtslsʘtʃtsˌlsˌjˌdʒˌj "[ˌj fɳʸlsɑ[ʏ[ɖƶ ˌj ʜtɑtsʙɖ["	1	1			ɹ ɖ[ʙɖɑ[tslsˌjˌʢʘʎ ʃ ʂ https://m.edsoo.ru/f2a1f23a
152	ʃ ɖʑsɑʈsɑʈɑʘɯɖɑ[ʢɖ. ʀ ʟtsʙtɑʘ[ˌj ɖɑɖ[ˌj dʒɖʑsɑʈsɑʈɑʘɯɖɑ[ʢtsɔ. ʃ tsʜˌj ɖɑ[tʃ tɑtsfɱʂtɑʘɯɖfɱʂɔˌj ɖɑɖƶ ⊐ lsˌj dƶ	1				ɹ ɖ[ʙɖɑ[tslsˌjˌʢʘʎ ʃ ʂ https://m.edsoo.ru/f2a1a69a
153	ʃ ɖʑsɑʈsɑʈɑʘɯɖɑ[ʢɖ. ʀ ʟtsʙtɑʘ[ˌj ɖɑɖ[ˌj dʒɖʑsɑʈsɑʈɑʘɯɖɑ[ʢtsɔ. ʃ tsʜˌj ɖɑ[tʃ tɑtsfɱʂtɑʘɯɖfɱʂɔˌj ɖɑɖƶ ⊐ lsˌj dƶ	1				ɹ ɖ[ʙɖɑ[tslsˌjˌʢʘʎ ʃ ʂ https://m.edsoo.ru/f2a1ad2a
154	ɻ tɕʷdʒtslzɑʈsdƶ dƶ ʔ tʃʘɯɑʘɯɖɑdˌjˌtʃ ɖ[tʃˌj ʜ, ʢlzʙ. ʈ ʘɻɔʸ tɑɖsʢɖ[ʢlzʙʘɖ[tʃ ʘɯɑʘɯɖɑdˌjˌtʃ ɖ[tʃˌj ʜʘ	1				ɹ ɖ[ʙɖɑ[tslsˌjˌʢʘʎ ʃ ʂ https://m.edsoo.ru/f2a1a802
155	ɻ tɕʷdʒtslzɑʈsdƶ dƶ ʔ tʃʘɯɑʘɯɖɑdˌjˌtʃ ɖ[tʃˌj ʜ, ʢlzʙ. ʈ ʘɻɔʸ tɑɖsʢɖ[ʢlzʙʘɖ[tʃ ʘɯɑʘɯɖɑdˌjˌtʃ ɖ[tʃˌj ʜʘ	1				ɹ ɖ[ʙɖɑ[tslsˌjˌʢʘʎ ʃ ʂ https://m.edsoo.ru/f2a1a924

156	ɪ tɛʘʕlsʃˤuˌ fɧʕʘʸ tɛʘʁtslsʘtʃts lsˌdʒˌ "t ʘʟɵʸ tɛlsʕʘʕɬzʁʘ'	1		1		ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1aef6
157	ʃ ʁʲ ʸ dʒʕɬzʁʘ, tʃtɛʷdʒʂslzɕtsdʓ dʓsɕts tʃʘɹɛɕɵɖzˌdˌtʃɖʃtʃˌ ʜʘ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1b09a
158	ʃ ʁʲ ʸ dʒʕɬzʁʘ, tʃtɛʷdʒʂslzɕtsdʓ dʓsɕts tʃʘɹɛɕɵɖzˌdˌtʃɖʃtʃˌ ʜʘ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1b248
159	ʃ ʁʲ ʸ dʒʕɬzʁʘ, tʃtɛʷdʒʂslzɕtsdʓ dʓsɕts tʃʘɹɛɕɵɖzˌdˌtʃɖʃtʃˌ ʜʘ	1				
160	ʃ ʁʲ ʸ dʒʕɬzʁʘ, tʃtɛʷdʒʂslzɕtsdʓ dʓsɕts tʃʘɹɛɕɵɖzˌdˌtʃɖʃtʃˌ ʜʘ	1				
161	ɪ tsɵlststɛˌdʓʃˌ tsfɳdʓsɵdʓ ⊐ tʃtsdʓˤlsʃʔ ʃ dʒˌlstsʜtsɵ ʕlztɛfɳ·5 ʕdɕʃfɳʃˌ tsʁtsʁʰ ˌdʓʃˌ ʟdɕ·dʓʃʔ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1f76c
162	ɪ tsɵlststɛˌdʓʃˌ tsfɳdʓsɵdʓ ⊐ tʃtsdʓˤlsʃʔ ʃ dʒˌlstsʜtsɵ ʕlztɛfɳ·5 ʕdɕʃfɳʃˌ tsʁtsʁʰ ˌdʓʃˌ ʟdɕ·dʓʃʔ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1f924
163	ɪ tsɵlststɛˌdʓʃˌ tsfɳdʓsɵdʓ ⊐ tʃtsdʓˤlsʃʔ ʃ dʒˌlstsʜtsɵ ʕlztɛfɳ·5 ʕdɕʃfɳʃˌ tsʁtsʁʰ ˌdʓʃˌ ʟdɕ·dʓʃʔ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1faaa
164	ɪ tsɵlststɛˌdʓʃˌ tsfɳdʓsɵdʓ ⊐ tʃtsdʓˤlsʃʔ ʃ dʒˌlstsʜtsɵ ʕlztɛfɳ·5 ʕdɕʃfɳʃˌ tsʁtsʁʰ ˌdʓʃˌ ʟdɕ·dʓʃʔ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1fc08
165	ɪ tsɵlststɛˌdʓʃˌ tsfɳdʓsɵdʓ ⊐ tʃtsdʓˤlsʃʔ ʃ dʒˌlstsʜtsɵ ʕlztɛfɳ·5 ʕdɕʃfɳʃˌ tsʁtsʁʰ ˌdʓʃˌ ʟdɕ·dʓʃʔ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a1feec
166	ɪ tsɵlststɛˌdʓʃˌ tsfɳdʓsɵdʓ ⊐ tʃtsdʓˤlsʃʔ ʃ dʒˌlstsʜtsɵ ʕlztɛfɳ·5 ʕdɕʃfɳʃˌ tsʁtsʁʰ ˌdʓʃˌ ʟdɕ·dʓʃʔ	1				ɹ ɖʃʁɖʓʃtslsˌʕʘʎ ʃ ʂ https://m.edsoo.ru/f2a200a4

167	ʀ ls ts ɕ ts ə ʘ ʏ ʕ ts d ʑ s t ɕ ts d ʑ d ʑ ʘ ʏ t ɕ ʘ ʁ ts ls ʘ	1	1			
168	ɬ ts ə ls ts t ɕ j d ʑ ʃ j ts f q ʑ s ə ʑ ⊐ t ʃ ts d ʑ ʏ ls q ʔ q ʃ d ʒ j ls ts ʜ ts ə ʕ ʐ t ɕ f ɳ ʘ 5 ʕ d ʑ ʃ f ɳ ɳ ʘ ts ʁ ts ʁ ʰ j d ʑ ʃ j ʟ d ʑ ʘ d ʑ q ʔ	1				ɹ q ʃ ʁ d ʑ q ts ls j ʕ ʘ ʎ ʃ ʂ https://m.edsoo.ru/f2a201f8
169	ɬ ts ə ls ts t ɕ j d ʑ ʃ j ts f q ʑ s ə ʑ ⊐ t ʃ ts d ʑ ʏ ls q ʔ q ʃ d ʒ j ls ts ʜ ts ə ʕ ʐ t ɕ f ɳ ʘ 5 ʕ d ʑ ʃ f ɳ ɳ ʘ ts ʁ ts ʁ ʰ j d ʑ ʃ j ʟ d ʑ ʘ d ʑ q ʔ	1				ɹ q ʃ ʁ d ʑ q ts ls j ʕ ʘ ʎ ʃ ʂ https://m.edsoo.ru/f2a20388
170	ɬ ts ə ls ts t ɕ j d ʑ ʃ j ts f q ʑ s ə ʑ ⊐ t ʃ ts d ʑ ʏ ls q ʔ q ʃ d ʒ j ls ts ʜ ts ə ʕ ʐ t ɕ f ɳ ʘ 5 ʕ d ʑ ʃ f ɳ ɳ ʘ ts ʁ ts ʁ ʰ j d ʑ ʃ j ʟ d ʑ ʘ d ʑ q ʔ	1				ɹ q ʃ ʁ d ʑ q ts ls j ʕ ʘ ʎ ʃ ʂ https://m.edsoo.ru/f2a2069e
ʃ ɹ ʐ ɻ ɻ ʂ ʃ ʃ ʀ ʏ ɻ ʜ ʊ ɹ ʃ ʏ ʧ ʜ ʃ ɹ ɬ ʃ ɬ ʃ ɻ t ʃ ɻ t ʧ ʃ ʃ ɻ		170	4	4		

6 ş ∫ ф ч ш

- tƒ/tƒ	ʊ j dʒΘ lztœs£Θ	ş tsdzſujfhjs ɔ ts ʋCfhjs ɔ				
		⊿I fgj cŵts	ş tsdzlstœsdz dz℅ r j tœΘɜtslsr	ɻ tœΘ£lsſujfhj cſj tœΘɜtslsr	⌈r Θlsθ cſuLzujdz℅ cſw	⌈r tsſjtsdzdzſlsj dz dz℅ Θv cſdzẄ tsœdʒΘuſcſ w
1	ɻ ts ɔ lststœj dzſj £ lztœfhjΘ5 £ dzCfhjhjΘ\ Θlslztœ Θdz dz℅ j ʋſcſfhjdzΘ	1			4.09	
2	ɻ ts ɔ lststœj dzſj £ lztœfhjΘ5 £ dzCfhjhjΘ⌋ ſ в r £ dzts ɔ j dzdz℅ j нtœtsвcſ	1			5.09	
3	ɻ ts ɔ lststœj dzſj £ lztœfhjΘ5 £ dzCfhjhjΘ ⌈ j fhjvlscſuſdz℅ j нtœtsвcſ	1			6.09	
4	ɻ ts ɔ lststœj dzſj £ lztœfhjΘ5 £ dzCfhjhjΘ.ɻ j tscdʒj lstœcſujfhj£cſj wẄ cſcɔlztœr	1			7.09	
5	ш tœj нdzj ɻ Θtœcſẅ dʒj lscſujfhj£tsj	1			8.09	
6	ɻ tœtsujdzts r	1			11.09	
7	ɻ j tœj ɔtsн ʋcſfhjdzΘɔ tƒtœtsujdzts r	1			12.09	
8	ɻ j tœj ɔtsн tƒtœtsujdztsɔ ɔ ʋcſfhjdzts	1			13.09	
9	ţ j h j dzſj LΘнΘɔɻdzΘlsj dʒz «ɻ tœtsujdzts r »	1			14.09	
10	ş tœlzcɔtsɔΘv нcſΘœtœΘdʒcʒΘ	1			15.09	
11	ɻ tœj нfhjsΘɔ dzj dzſj ʋcſfhjdzts ɔ ts? cſdzẄ tsœdʒΘuſcſ ɔ £ tœlzcɔtsɔ r ⊐ нcſΘœtœΘdʒcʒΘ⊐	1			18.09	
12	⊿I cſн r lstœj lzcɔtsdz dzſ£ tsɔ	1			19.09	
13	υ tœts£ tsвtsв h j dzſw cſ fhjſfhjsj dʒΘlscſLΘɔuſcſɻLdzΘdzſ?	1			20.09	
14	**ş tsdzlstœsdz dzΘv tœΘɜtslsΘ¯ 1**	**1**	**1**		21.09	
15	ɻ tœtsfhjs r j cſ fhjsfhjsΘɔ dz j ʋcſfhjdzΘ	1			22.09	
16	ţ Θldzts›ɻ j dzſj ʋcſfhjdzΘdzΘ tƒtœtsfhjs r j dʒdzts›ɻ cſlsj dzſ	1			25.09	

17	t ɻ ʰ ɻ ʤ ɻ lstɕ ʤ ʧtʂs ɔ tsʏʤ ⊐ ʟ Θ ʜ Θ ʏ ʤ Θ lsɻ ʤ ʒz «t Θ ɻ ʤ ʂ ɻ ɻ ʤ ɻ ʏ ʧ fɳ ʤ Θ ʤ Θ tʃ ʧ ʂ fɳ ʒ ɻ ʤ ʒ ʤ ʂ ɻ ʧ lsɻ ʤ ɻ»	1			26.09	
18	l Θ ʧ ʙ ʦ ʂ ʤ ʰ ʧ ʔ ʦ ʙ ʱ ʧ ʔ ʜ ɻ ʤ ʧ lsɻ ʤ z	1			27.09	
19	ɸ ʤ ʔ ʦ ʦ ɕ ʧ lsʧ ʒ ʤ Ɛ ʔ ʦ ʂ ɻ ʜ ɻ ʤ ɻ ʷ l ʃ ɾ	1			28.09	
20	ɻ ʟ Θ ʧ ʤ ʒ ʤ ʂ ʦ ʃ ʧ ʂ fɳ ʒ ɻ ʏ ʧ fɳ ʤ Θ	1			29.09	
21	t ɻ ʰ ɻ ʤ ɻ ʟ Θ ʜ Θ ʏ ʤ Θ ʤ Ɛ ʔ ʦ ʂ ɻ ʜ ɻ ʤ ɻ ɻ l ʃ ɾ	1			02.10	

22	[illegible]	1			03.10	
23	[illegible]	1			04.10	
24	[illegible]	1			05.10	
25	[illegible]	1			06.10	
26	[illegible]	1			09.10	
27	**[illegible] 2**	**1**	**1**		10.10	
28	[illegible]	1			12.10	
29	[illegible]	1			13.10	
30	[illegible]	1			16.10	
31	[illegible]	1			17.10	
32	[illegible]	1			18.10	
33	[illegible]	1			19.10	
34	[illegible]	1			20.10	
35	[illegible]	1			23.10	
36	[illegible]	1			24.10	
37	[illegible]	1			25.10	
38	[illegible]	1			26.10	
39	[illegible]	1			27.10	
40	[illegible]	1			07.11	
41	[illegible]	1			08.11	
42	**[illegible] 3**	**1**	**1**		09.11	

43	[j ʔ fhjs o[j lzdʒdtssʏ j dz[w fnhzj h Qdzdz ⊐ ʊ[fhjdz	1			10.11	
44	t j h j dz[j tʃtɑ[dʒjtɑtss o dzQ ʜj ʔ fhjs o[j lzdʒdtssʏ j dz[w fnhzj h Qdzdz ⊐ ʊ[fhjdz	1			13.11	
45	l QTtsʏ ʜj dz[j ʜtɑtsʙo[tsls ʊ[fnhzQ	1			14.11	

46	Урок-путешествие Проект «Наш класс – наша семья»	1			15.11	
47	Проект «Наш класс – наша семья»	1			16.11	
48	Контрольный диктант по теме «Звуки и буквы» Работа над ошибками	1			17.11	
49	Контрольный диктант по теме «Звуки и буквы» Работа над ошибками	1			20.11	
50	Повторение изученного материала Контрольный диктант по теме «Звуки и буквы» Работа над ошибками	1			21.11	
51	Текст. Работа с текстом	1			22.11	
52	Проект «Рассказ о слове» Работа с текстом	1			22.11	
53	Проект «Наш класс – наша семья»	1			23.11	
54	Урок-путешествие Работа с текстом	1			24.11	
55	Проект «Наш класс – наша семья»	1			27.11	
56	Текст. Работа с текстом	1			28.11	
57	Повторение изученного материала Урок-путешествие	1			29.11	
58	**Контрольная работа по теме «Звуки и буквы»**	**1**	**1**		30.11	
59	Текст. Работа с текстом	1			01.12	
60	Урок-путешествие Работа над ошибками	1			04.12	
61	Предложение. Работа с предложением	1			05.12	
62	Урок-путешествие Работа с текстом	1			06.12	
63	Повторение изученного материала Проект «Рассказ о слове»	1			07.12	
64	Повторение изученного материала	1			08.12	
65	Текст. Работа с текстом	1			11.12	
66	Повторение изученного материала	1			12.12	
67	Проект «Наш класс – наша семья» Работа над ошибками	1			13.12	
68	Повторение изученного материала Работа над ошибками	1			14.12	
69	Текст. Виды текста Работа над ошибками	1			15.12	

70	∫ Gfh] lsO3	1			18.12	

71	[j h j dzfj LOHOUdzO tslsdzsh j dzfw,tftstsftstcuqfqf dgGfhj lsOs	1			19.12	
72	U tstsF tssstsh j dzfw qf fgffhsj dgOsqfLOUqfqfLdzOdzf?	1			20.12	
73	**S tsdlststsdz dzOy tcOstslsO¯ 5**	**1**	**1**		21.12	
74	f fjs sOy, uj dzstcOdz dzOy qf Lj tsFOdz dzOy fgfdgdgj lstcfqf	1			22.12	
75	t tsfhstcsj dzfj fgfdgdgj lstcfudz = ww qfdzts	1			25.12	
76	H qfdgdgj lstcfw s tftstsfhstcOdzhsj	1			26.12	
77	t tcOlsfujfhjOy tcOstslsO «f fjs sOy fgfdgdgj lstcfw»	1		1	27.12	
78	[dzfdzOtsFtdz>q dzsfhsf	1			28.12	
79	t tcOlsfujfhjOy tcOstslsO «f lsdzsh j dzfj Hdzfdz tsFtdz>q dzsfhsf Fj j Hf Odgj lstdz»	1		1	09.01	
80	t dzsh OHI FtdzgO	1			10.01	
81	t tcOlsfujfhjOy tcOstslsO«t dzsh OHI FtdzgO»	1		1	11.01	
82	t tsdzs>q flsj dz dz j f tslstcfuOsj dz dz j uf fgdzO	1			12.01	
83	t tsdzs>q flsj dz dz j f tslstcfuOsj dz dz j uf fgdzOdzOFtststcHfdzOsdzs? tftcwdgs?	1			15.01	
84	t tstslsfo tsfsdzs>q dz j uf fgdzO	1			16.01	
85	v j dz j uf fgdzO	1			17.01	
86	f tsHlzdz uf fgdzO	1			18.01	
87	,j tsdgj lstcfujfhjOy qfdlsj tefjtej lsOuqfw dgsHlzdzy uf fgdzO	1			19.01	
88	t tcfdgj tcr tcO1oj tdstsF dgdlscstscOdldzqfFtso,uqfdzfdzHtcOqf Ftsdlzfh	1			22.01	
89	U tstsF tssstsh j dzfw qf fgffhsj dgOsqfLOUqfqfLdzOdzf?	1			23.01	
90	**S tsdlststsdz dzOy tcOstslsO¯ 6**	**1**	**1**		24.01	
91	H tcOo dgj dzfj tftsdzs>q flsj dz dz = qf tslstcfuOsj dz dz = uf fjdz	1			25.01	

92	ʉ tɕɵɔ dʑ dʑʃʝ ʧ tsdʐsɤ ɕlsʲ dz dz ⊐ ɕ tslstɑʃ ʋɥɵsʲ dz dz ⊐ ʋɕʃfŋ dzfŋ ʧ tsdʒʂʰ ɹ ʁ ʕ tststɑ↔ɕ dʑɵsdʐs? ʧ tɕʷ dʒʂ?	1			26.01	
93	t ʝ ʰ ʝ dʑʃʝ ʟɵ↔ɵ ɻ dʑɵ fŋɕɔ dʑ dʑʃʝ ʧ tsdʐsɤ ɕlsʲ dz dz ⊐ ɕ tslstɑʃ ʋɥɵsʲ dz dz ⊐ ʋɕʃfŋ dz	1			29.01	

94	ʁ ʟdʒ dʑ dz[j ɘj dz[ˆɥ[dz.	1			30.01	
95	ʉ dzsʏ j dz[j ɘɕ[нΘ-a + b fŋ ʧtsdʒs^{h} ɹ ʁ ʕtststɑ+ɕ[dzƟsdzs? ʧtɕ^{w}dʒs?	1			31.01	
96	ʉ dzsʏ j dz[j ɘɕ[нΘ-a + (-b) fŋ ʧtsdʒs^{h} ɹ ʁ ʕtststɑ+ɕ[dzƟsdzs? ʧtɕ^{w}dʒs?	1			01.02	
97	ʉ dzsʏ j dz[j ɘɕ[нΘ-a + a fŋ ʧtsdʒs^{h} ɹ ʁ ʕtststɑ+ɕ[dzƟsdzs? ʧtɕ^{w}dʒs?	1			02.02	
98	ɪ Ɵʕtɑj ʧ dʑ dz[j dzƟɘ^{r} ʕtsɔ fɥdzsʏ j dz[^{w} ʧtsdzsʏ ɕ[lsj dz dz. ⊐ ɕ[tslstɑ[ˆɥƟsj dz dz. ⊐ ˆɥ[fŋj dzfŋʧtsdʒs^{h} ɹ ʁ ʕtststɑ+ɕ[dzƟsdzs? ʧtɕ^{w}dʒs?	1			05.02	
99	ʉ dzsʏ j dz[j tslstɑ[ˆɥƟsj dz dz. ⊐ ˆɥ[fŋj dz	1			06.02	
100	t ɻ^{h} j dz[j ʟƟнΘˆɥʧts lsj dʒj «ʉ dzsʏ j dz[j tslstɑ[ˆɥƟsj dz dz. ⊐ ˆɥ[fŋj dz»	1			07.02	
101	ʉ dzsʏ j dz[j ˆɥ[fŋj dzfŋtɑΘɻdz. dʒ[ʟdzƟʕΘdʒ[	1			08.02	
102	ɸ dzɪtstɑ[lsdʒ fɥdzsʏ j dz[^{w} ˆɥ[fŋj dzfŋ tɑΘɻdz. dʒ[ʟdzƟʕΘdʒ[	1			09.02	
103	t ɻ^{h} j dz[j ʟƟнΘˆɥʧts lsj dʒj «ʉ dzsʏ j dz[j ˆɥ[fŋj dzfŋtɑΘɻdz. dʒ[ʟdzƟʕΘdʒ[»	1			12.02	
104	ɾ ɻʔfŋɘɔ[j ɘ^{r} ˆɥ[lsΘdz[^{w}	1			13.02	
105	ɭ Ɵɪtsʏ нj dz[j нdz[dz. tslstɑj ʟʕΟ dzƟʕtststɑ+ɕ[dzƟsdzs? ʧtɕ^{w}dʒs?	1			14.02	
106	t ɻ^{h} j dz[j ʟƟнΘˆɥʧts lsj dʒj «ɾ ɻʔfŋɘɔ[j ɘ^{r} ˆɥ[lsΘdz[^{w}»	1			15.02	
107	ɾ ɻʔfŋɘɔ[j ɮdʒdzsʏ j dz[^{w}. ʊ dʒdzsʏ j dz[j нɘɮɪˆɥ[fŋj dzfŋtɑΘɻdz. dʒ[ʟdzƟʕΘdʒ[	1			16.02	
108	ʊ dʒdzsʏ j dz[j нɘɮɪtslstɑ[ˆɥƟsj dz dz. ⊐ ˆɥ[fŋj dz	1			19.02	
109	t ɻ^{h} j dz[j ʟƟнΘˆɥʧts lsj dʒj «ɾ ɻʔfŋɘɔ[j ɮdʒdzsʏ j dz[^{w}»	1			20.02	
110	ɾ ɻʔfŋɘɔ[j нj dʑ dz[^{w}. ɾ ɻ dʑ dz[j нɘɮɪ ˆɥ[fŋj dzfŋtɑΘɻdz. dʒ[ʟdzƟʕΘdʒ[	1			21.02	
111	ɾ ɻ dʑ dz[j нɘɮɪtslstɑ[ˆɥƟsj dz dz. ⊐ ˆɥ[fŋj dz	1			22.02	
112	t ɻ^{h} j dz[j ʟƟнΘˆɥʧts lsj dʒj «ɾ ɻʔfŋɘɔ[j нj dʑ dz[^{w}»	1			26.02	

113	υ tɛtsʕ tsʙtsʙʰ ʝ dʑqʷ ɕ fɳʃfɳʰʝ dʒʘsɕ[ʟʘ.ɻɕ[ɕ[ʟdʑʘdʑ[ʔ	1			27.02	
114	**ʂ tɕdʑstɛtsdz dʑʘʸ tɛʘʒtsʦʘ⁻ 7**	**1**	**1**		28.02	

115	ʎ ɕʃʤɕʃʤʁʦɕ, ʰ ʘʦɕɕʃ fɲʷ ɟ ʦɐʘ	1			29.02	
116	ɻ ʦᵂʤɟʂʐɕʦsʤ ʤ. ʔ ʧʃʘʦɐʘʤʤʑʤɟ ɟ ʧʃɕʃʧɟ ʜ, ʕʂʐʙ	1			01.03	
117	ʀ ʟʦsʙ ʦɐʘɦ ɟ ʤɕʃɟ ʧʃʦɐʦsfɲʂʦɐʘʤʑfɲʂɐɔ ɟ ʤʤʤ. ⊐ ʷʷ ɕʃɕʟʤʦɕ	1			04.03	
118	ɻ ʦsʤʷʂsɕʃɟ ʦsʙʲ ɟ ɕʒʘ ɟ ʜɕʃʤɕʃʯʳ ɕʃʟɕʤɟ ʦɕɟ ʤɕʃʷʦsʙʲ ɟ ɕʒʘ	1			05.03	
119	ʃ ʙʲ ɟ ɕʒ ʧʃʦᵂɕʒʂʐɕʃʦsʤ ʤʑsɕʃʦ ʧʃʘʦɐʘʤʤʑʤɟ ɟ ʧʃɕʃʧɟ ʜʘ ʕʂʐʙʘ	1			06.03	
120	ɻ ʦɐʘʕʂsɕʃʯɟ fɲʃʘʸ ʦɐʘɔʦʂsʂʘ«ʉ ʦsʟʜʘʤʑfɲ ɕʒʂʦsʜɟ ɕʤɟ ʔ ʧʃʦɐʦsfɲʂʦɐʘʤʑfɲʂɐɔ ɟ ʤʤʤ. ⊐ ʷʷ ɕʃɕʟʤʦɐɔ»	1		1	07.03	
121	ʈ ʘʯɕʃʦsʤʘʤʑ ʤʑsɟ ʯɕʃfɲʤʑʦ	1			11.03	
122	ɻ ɟ ʦɐʃʦsʜɕʃʯɟ fɲʃʘʸ ʜʦɐʦsɔʲ	1			12.03	
123	ɻ ɟ ʦɟ ɕʒɟ fɲʂɕʃʂɟ ɕʤ ʤʑsɟ fɲɔʦs? fɲʂɔʦs fɲʤʑsʏ ɟ ɕʤɕʃʷ ɕʃʤɕʒʤʑsʏ ɟ ɕʤɕʃʷ	1			14.03	
124	ʉ ʦsʯɟ ʂsʘʂɟ ɕʤ ʤʑsɟ fɲɔʦs? fɲʂɔʦs fɲʤʑsʏ ɟ ɕʤɕʃʷ ɕʃʤɕʒʤʑsʏ ɟ ɕʤɕʃʷ	1			15.03	
125	ʈ ɟ ʰ ɟ ɕʤɕʃɟ ʟʘʜʘʯʟ ɕʤʘ ʧʃɟ ɟ ɕʤɟ ɟ ɕʒɟ fɲʂɕʃʂɟ ɕʤ ʤʑsɟ ɕʃfɲʦsʯɟ ʂsʘʂɟ ɕʤ ʤʑsɟ fɲɔʦs? fɲʂɔʦs ʘ	1			18.03	
126	ʈ ʘfɲʃɟ ɟ ʜɟ ɕʤɕʃʂɟ ɕʤ ʤʑsɟ fɲɔʦs? fɲʂɔʦsʂʤɕʒʤʑsʏ ɟ ɕʤɕʃʷ	1			19.03	
127	ʈ ɟ ʰ ɟ ɕʤɕʃɟ ʟʘʜʘʯʟ ɕʤʘ ʘfɲʃɟ ɟ ʜɟ ɕʤɕʃʂɟ ɕʤ ʤʑsɟ fɲɔʦs? fɲʂɔʦs ʂʤɕʒʤʑsʏ ɟ ɕʤɕʃʷ	1			20.03	

128	ʉ ɵtsʔfŋʂɵΘʜjʔfŋʂɵɕʔ fŋ tɛΘɻɥɕ[tsɕʑʊʑ ɖʐ ɕʒʃˈɥɕ[fŋʐʊɕʒɕʃ: ʟGʕtɛʝtʃɕʲɕʑʃj	1			21.03	
129	ɻ tɛΘʕlsɕˈɥʲfŋʃΘʸ tɛΘɜtslsΘ «ɻ tsɕʑsɤɻɕˈlsʲɕʐ ɖʐ ʲ ɕˈ tslstɛɕʃɥΘlsʲɕʐ ɖʐ ʲ ɥɕ[fŋʐʊΘ»	1		1	22.03	
130	ʊ tɛtsʕ tsʙtsʙʰ ʲ ɕʑ[ʷ ɕ[fŋɕ[fŋʃʲ ɕʒΘlsɕʃʟΘɥɕ[ɕ[ʟɕʑʊʑ[ʔ	1			01.04	
131	**ʂ tsɕʑlstɛtsɕʑ ɕʑΘʸ tɛΘɜtslsΘ¯ 8**	**1**	**1**		02.04	
132	ʈ Ɛfŋʃtɛʳ lsɕʃʲ fŋʃtsʙtsʕ fŋʂ ʟɕʑʕtsɕʒ «+» tʃʲ tɕʲ ʜfŋʃtsʙʕΘɕʒɕ[	1			03.04	
133	ʈ Ɛfŋʃtɛʳ lsɕʃʲ fŋʃtsʙtsʕ fŋʂ ʟɕʑʕtsɕʒ «-» tʃʲ tɕʲ ʜfŋʃtsʙʕΘɕʒɕ[	1			04.04	
134	ʂ tsᵘ ʷ ʷ ɕˈɥɕ[ʲ ɕʑs	1			05.04	
135	ʊ tʃˈtɛtsʰ ʲ ɕʑ[ʲ ɘʳ tɛΘɻʲ ɕʑ[ʔ	1			08.04	
136	ɻ tsʜtsʙɕʐ ʲ fŋɥΘɛʕʲ ɕʒʳ ʲ	1			09.04	
137	ɻ tɛ[ɘʲ ʜʲ ɕʑ[ʲ tʃtsʜtsʙɕʐ ⊐ fŋɥΘɛʕʲ ɕʒʳ ⊐	1			11.04	

12.04

138	t j h j dzfj lztcGbdjdzf?	1			12.04	
139	ʃ cfdj?dzsj lztcGbdjdzfj	1			15.04	
140	υ tɛtsʔ ï tʃtcGʔlsfʔlzdʒtʃts tɕj h j dzfʁ lztcGbdjdzf?	1			16.04	
141	υ tʃtɛtsʰ j dzfj ɘr tcG|j dzf? cf tɕj h j dzfj lztcGbdjdzf?	1			17.04	
142	υ tɛtsʔ ï tʃtcGʔlsfʔlzdʒtʃts lztʃtɛtsʰ j dzfʁ ɘr tcG|j dzf? cf tɕj h j dzfʁ lztcGbdjdzf?	1			18.04	
143	υ tɛtsʔ ʟGʔtɕjtʃdjdzfw tɕj h j dzf? lztcGbdjdzf?	1			19.04	
144	t j h j dzfj lsjʔfhjstsɔr ⊐ ʟGHGuf	1			22.04	
145	υ tɛtsʔ ï tʃtcGʔlsfʔlzdʒtʃts tɕj h j dzfʁ lsjʔfhjstsɔr ⊐ ʟGHGuf fŋ tʃtscdʒtsʰ ɹ ʁ fŋsfhjGbdjdzfw lztcGbdjdzf?	1			23.04	
146	υ tɛtsʔ ʟGʔtɕjtʃdjdzfw tʃts tɕj h j dzfʁ lsjʔfhjstsɔr ⊐ ʟGHGuf fŋ tʃtscdʒtsʰ ɹ ʁ fŋsfhjGbdjdzfw lztcGbdjdzf?	1			24.04	
147	ɻ tcGʔlsfʰujfhjGw tcGɜtslsG «t j h j dzfj lztcGbdjdzf?»	1		1	25.04	
148	υ tɛtsʔ tsʁtsʁʰ j dzfw cf fŋffhjdʒGlsfʟGuffffʟcdzGdzf?	1			26.04	
149	**ʂ tsdlstɛtsdz dzGw tcGɜtslsG¯ 9**	**1**	**1**		29.04	
150	ɻ j tɛfj dzʜcfʔlzdzʸtɛdz j tʃtɛwdʒr j	1			30.04	

151	ɪ ɟ tɕʃ ɟ dʑʜɾɕʃʂ lzdzʷ tɕdʑ ɟ tslstɕ ɟ ʟʂɾɕʃ	1			02.05	
152	ɪ ɕɪɕɕɪdzdʑ dz dʑ ɟ ʧ tɕʷ dʒʳ ɟ	1			03.05	
153	ɪ ɕɪɕɕɪdzdʑ dz dʑ ɟ tslstɕ ɟ ʟʂɾɕʃ	1			06.05	
154	ʂ tststɕʜɾɕʃdʑɕɪsdʑɕʏ ʧ ʃdʑsfnʂ tsfnʃsʰ	1			07.05	
155	ʂ tststɕʜɾɕʃdʑɕɪsʳ lstsʋʂɾɕʃ dʑɕ ʧ ʃdʑsfnʂ tsfnʃsɕʃ,ɕɪɜ fnɻɕʃfnɲɕʃ	1			08.05	
156	ɻ tɕɕʷ ɾɕʃʂ	1			10.05	
157	ɪ tɕʝ ʜfnʃsɕɜ dʝ dʑʃj ʋɕʃfɲʃdzsɜ tsɾʔ ɾʃdʑʷ tstɕdʒɕʋɕʃɾʃdʑɕ ɕtɕɕʷ ɾʂʂɕʷ ɾʃʂɕʃ	1			13.05	
158	ɪ tɕɕʂʳ lsɾʃʋj fnʂɕʏ tɕɕɜtslsɕ «ɪ tsfnʃstɕtsʝ dʑʃj lstsʋj ʂɾ ɾʃʷ ɾʃɕɾʃdzɾtɕ dʑɕ ʂɾtststɕʜɾɕʃdʑɕɪsdʑsɾʔ ʧ ʃdʑsfnʂ tsfnʃsɕʃ»	1		1	14.05	
159	ʏ ɟ lsʳ tɕʝ ⊐ lzɕɾtsdz dʑʃʂ. ɪ tɕʷ dʒʃslzɕɾtsdz dʑʃʂ.ʂ ɘɕɜtɕɕɪs	1			15.05	

160	ɪ ʝ tɑʃdʒʝ lstɕ dʒʤsɕtslzɕtsdz dɑʃʕʘ ɪ dʤsʰ ʘʜᴶ ᵂ ɕʃɑ[ɕlzɑɕ . ɪ ʝ tɑʃdʒʝ lstɕ ɕʃ tʃ dʤsʰ ʘʜᴶ tʃ tɕʷ dʒʤslzɕtsdz dɑʃʕʘ	1			16.05	
161	ɪ tsɔ lststɕʝ dɑʃʝ ʕlzɑɕfɳʘ6 ʕ dɑɕfɳɲʘ [tɑtsʁɕʃ.ʃ lsdʤsʰ ʝ dɑʃʷ ɕʃ tʃ tɑtsʃ tstɑɕʃɕʃ	1			17.05	
162	ɪ tsɔ lststɕʝ dɑʃʝ ʕlzɑɕfɳʘ6 ʕ dɑɕfɳɲʘ t ʝʰ ʝ dɑʃʝ lzɑɕʘɔ dʒʝ dɑʃʔ	1			20.05	
163	ɪ tsɔ lststɕʝ dɑʃʝ ʕlzɑɕfɳʘ6 ʕ dɑɕfɳɲʘt ʘ ʋɕʃtsdɑʘdz dʑ ʝ ʋɕʃfɳʤʘ	1			21.05	
164	ʊ tɑtsʕ tsʁtsʁʰ ʝ dɑʃʷ ɕʃ fɳʃfhʂʝ dʒʘsɕʃʟʘʋɕʃɕʃʟdɑʘdɑʃʔ	1			22.05	
165	**ʂ tsdʤstɑtsdz dɑʘʸ tɑʘʁtslsʘ¯ 10**	**1**	**1**		23.05	
166	t ʝ ʟʝ tɑɔ	1			24.05	
167	t ʝ ʟʝ tɑɔ	1				
168	t ʝ ʟʝ tɑɔ	1				
169	t ʝ ʟʝ tɑɔ	1				
170	t ʝ ʟʝ tɑɔ	1				
ʃ ɪ ʒ ʃ ʃ ʂ ʃ ʃ ʀ ʏ ʃ ʜ ʊ ʃ ʃ ʏ ɸ ʜ ʃ ʃ ɪ ʃ ɪ t ʃ ɻ ʃ ɸ ʃ ʃ ʃ	170	10	7			

ʊ ʏ ɽ ɹ ɿ ʆ -ʄ ɽ ʊʆ ɼ ʀ ʏ ɽ ʉ ʂ ʆ ɽ ʆ ɹ ɽ ʉ ɟ ɽ ʏ ɽ ɿ ʀ ɽ
ʆ ɹ ɭ ɸ ɿʆ ɺ ɸʊɽ ʃ ʔ ɿ ʆ ɺʆ ɟ ɭʆ ʎ ɽ ʉ ʉ ɸ
ʆ ɹ ʗ ɿ ɸʊɽ ʃ ʔ ɿ ʒ ɽ ʊ ʏ ɽ ɹ ɿ ʒ ɽ ʄ ɸʊɽ ɭ ʀ ɸʃ ʒ ɼ ʃ ʗ ʊ ʏ ɽ ɿ ʀ ʂ ɸ

ʄ ɽ ʊʆ ɼ ʀ ʏ ɽ ʉ ʂ ʀ ɽ ʄ ɸʊɽ ɭ ʀ ɸʃ ʒ ɼ ʃ ʗ ʊ ʏ ʀ ʊɽ ʃ ʗ

ʎ ʀ ʌ ɭʆ ɺ ʒ ɽ ʆ ɹ ɭ ɸ ɿʆ ɺ ɸʊɽ ʃ ʔ ɿ ʒ ɽ ɭ ɽ ʉ ʊ ɭ ʉ ʒ ʀ ɭ ɽ ʉ ʊ ɭ ʉ ʒ ʉ ɽ ʊ ʀ

ʀ ɿ ʊɽ ɭ ɿ ɽ ʊ