MA Computational Linguistics, University of Wolverhampton
7LN004 Computational Linguistics (2020-21)
Program for Assessment
Christopher Vidler

Computational Linguistics Module Assessment, Program Section: A Basic Program for Transliterating (Russian) Cyrillic Characters to the Latin Alphabet

Program Description

The code I have written below is an attempt to create a basic program using the Python programming language for transliterating Cyrillic characters (specifically, only those that are used in the Russian alphabet) to equivalent characters from the Latin alphabet.

As touched upon in the essay that this program accompanies, there is effectively no single system for transcribing Cyrillic that is commonly used and agreed upon, but rather a number of different systems that exist instead, some of which are officially recognised and used for specific purposes. In order to provide some standardised character mappings for the transliteration program to work with, three of the most prominent of these systems have been selected for inclusion in the program, which can then be selected by the user as a basis for transliterating the entered string:

- The **International Scholarly System** (also known as the scientific system) for transliterating Cyrillic, which has been used in linguistics publications since the 19th century, as it is described in Wikipedia's article on the scientific translation of Cyrillic (2021a)
- The **ISO 9:1995** standard for transliterating Cyrillic characters into Latin characters (International Organization for Standardization, 1995), which is identical to the GOST 7.79-2000(A) standard system used within Russia and other countries of the CIS, and is notable for its use of one-to-one character mappings that also allow for reliable reverse transliteration
- The **International Civil Aviation Organization (ICAO)** guidelines for transliteration used in the production of machine readable travel documents, e.g. passports (International Civil Aviation Organization, 2015)

These different systems are introduced in more detail in the printed instructions for the program itself, along with links to relevant sources of information on these systems, which are also summarised in a comprehensive transliteration table in Wikipedia's article on the romanisation of Russian (2021b).

As highlighted in the essay section, this ability to choose from a selection of different transliteration systems is an additional feature that has been added to the program in order to distinguish it in some way and expand the options that are available to the user. Another additional feature is the opportunity to convert multiple input strings in a single run of the program, to avoid having to re-run the program each time. To incorporate these features, the program has been augmented with elements to control program flow, such as while loops with sentry variables to allow the repetition of certain steps where required, and a custom function to restrict user input to certain strings using if/else conditions in conjunction with break/continue statements within a while loop.

Setting aside these additional features, however, the way that the core principle of the program works is what I believe to be the simplest approach to automatic transliteration that involves pure, consistently applied character-for-character(s) substitution. Further details are provided in the comments in the program code below, but a basic summary is outlined here:

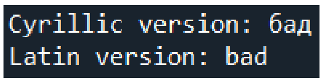
- A string is entered by the user for transliteration.
- Since strings are immutable objects in Python, this string is converted into a (mutable) list with each character of the string comprising a separate item in the list, which facilitates processing by allowing each character to be modified on an individual basis.
- A new list is then created that will comprise the transliterated string: to do this, the program uses Python's get() method in conjunction with a *for* loop to iterate over each character in the list of the original string and check these characters against a dictionary of character mappings for the selected transliteration system. If the character in question is a Cyrillic character that corresponds to a key in the dictionary, the value for that key, which is its Latin equivalent in the given transliteration system, is returned using get() and added to the new list; if the character in question is any other character, get() returns this character unmodified and adds it to the list as well.
- Once the new list has been created (which essentially consists of the same characters in the same order as the original string, but with any Russian Cyrillic characters modified), this list of characters is then converted back into a complete string using Python's join() method.
- This transliterated string is then printed for the user.

The advantage of this approach (and the use of the get() method in particular) is that the program is able to handle virtually any input from the user without throwing an exception, even if no Russian Cyrillic characters for transliteration are actually identified. It is important for the program to be able to return the original string with its characters in the same order with only the necessary characters modified and with the same punctuation and spacing elements anyway, and so this approach is favourable for ensuring consistency in this regard.

One other approach that was considered and explored in my early experimentations with Python is to use Python's maketrans() and translate() methods to create a mapping table (using strings rather than a dictionary) that would then be used to convert the necessary characters of the string to their Latin equivalents. A basic example of this approach (using only the first five characters of the Russian alphabet) is demonstrated very briefly below:

```
# Mapping Table Method Example:

# A sample string for testing is assigned to a variable and printed.
test_string = "бад"
print("Cyrillic version: " + test_string)

# Source characters for mapping are included in a string
# that is assigned to a variable; in this case, these characters
# are the first five letters of the Russian alphabet.
translit_table_source_chars = "абвгд"

# The target characters to be mapped are assigned to another variable;
# these must be positioned within the string to match the position
# of the corresponding source character in the string assigned
# to the previous variable.  The two strings must be of equal length,
# and only one-to-one character mappings can be made with this method.
translit_table_target_chars = "abvgd"

# The two variables are passed to the maketrans() method,
# which is called on the test_string to create a mapping table.
# This mapping table is then assigned to the variable translit_table.
translit_table = test_string.maketrans(translit_table_source_chars,
                                       translit_table_target_chars)

# The transliterated version of the test_string is then printed
# by using the translit_table variable as a parameter
# for the translate() method on the test_string.
print("Latin version: " + test_string.translate(translit_table))
```

Output:



```
Cyrillic version: бад
Latin version: bad
```

While this bypasses the issue of string immutability and eliminates the need to break the string down into a list of characters altogether, the main issue with the above approach is that the strings used for character mapping must be of equal length and only allow for one-to-one character mappings. Although this approach would still technically be feasible for the ISO 9 transliteration system, which uses one-to-one character mappings anyway, this would present an issue for most other transliteration systems that render certain Russian characters with two or more letters in a romanised script (e.g. "ж" > "zh", "ц" > "ts", "щ" > "shch"). (The ability to pass a dictionary directly as a parameter to the translate() method without creating a string mapping table with maketrans() was also considered as an alternative, but further research and testing indicated that a dictionary passed in this way could only contain ASCII values belonging to the basic 127-character ASCII set, and so could not be used to map Cyrillic characters.)

The string > list > modified list > modified string approach using the get() method in conjunction with a corresponding dictionary was therefore considered to be next best alternative, as it can easily allow for one-to-multiple character mappings. It also perhaps allows for a greater level of flexibility and customisation if the program is modified further, which is discussed in more detail in the "Notes on Results and Potential for Further Modification" section following the output examples below.

The program code presented below was written and run using Python version 3.8.3. Insofar as possible, the code has been formatted in line with the Python PEP8 style guide (van Rossum, Warsaw and Coghlan, 2001). The code was written with the aid of the Spyder IDE and all testing was performed using this IDE, with output screenshots taken from its console window.

With regard to testing, three documented tests have been performed and presented in the "Output Examples" section that follows the code, each with a different set of purposes. The first test uses a Russian pangram as the input string for a simple demonstration of the program, showing how it modifies each letter according to each of the selectable transliteration systems. The second test explores how the program could be used for a real-world application such as the automatic transliteration of names and titles of works in bibliographic information, and tests how it handles more complex features such as irrelevant characters and line breaks. The final test has mainly been included to show how the input restrictions work and to demonstrate how the program can be used to convert multiple strings in a single session.

Program Code

```
# Computational Linguistics Essay Program:
# A Basic Program for Transliterating (Russian) Cyrillic Characters
# to the Latin Alphabet


# A function is defined here at the start of the program that ensures
# the user can only enter "Y" (for yes) or "N" (for no) when prompted.
# The function's if condition ensures that only these letters
# can be entered to break the infinite while loop within the function
# and continue the program.  This is used to control program flow
# towards the end of the program and to allow the user to loop back
# to various steps, enabling them to test multiple inputs
# and/or transliteration systems in single run of the program.
def y_or_n_input(prompt):
    """Restrict user input to return either "Y" or "N"."""
    while True:
        y_or_n_answer = input(prompt).upper()
        if (y_or_n_answer != "Y" and y_or_n_answer != "N"):
            print("Error: Please enter either Y or N to proceed.")
            continue
        else:
            break
    return y_or_n_answer


# To begin, instructions are printed here that introduce the program
# and describe its functionality to the user.
print("""Welcome to the Russian text transliteration program!

This program allows the user to input a string of text containing
Russian letters, and have this string returned with the Cyrillic
characters romanised to Latin equivalents according to
a transliteration system selected by the user.

Please note that this program is only designed to convert
the 33 letters that appear in the Russian alphabet. Certain Cyrillic
characters that feature in other languages, such as Ukrainian "ї"
or Serbian "ђ", will not be converted by this program,
and the systems used to convert compatible letters will handle these
according to conventions used in transliterating Russian (i.e. "г"
will become "g", rather than "h" as it is commonly romanised
from Ukrainian and Belarusian.)

Although there is no commonly accepted standard for romanising Russian,
a few official systems exist for use in various applications such as
linguistics research, producing machine readable documents or
transliterating place names on road signs. The user can have
their string of text transliterated according to one of three systems:

- The International Scholarly System (also known as the scientific
  transliteration system), which has been used in linguistics
  publications since the 19th century

- The ISO 9 standard for transliterating Cyrillic characters into Latin
  equivalents, in its most recent version from 1995, which is notable
  for its use of one-to-one character mappings (which make use
  of letters with diacritics) that also allow for reliable reverse
  transliteration.

  The ISO 9 system has also been adopted into the GOST (ГОСТ) standards
  used by the countries of the CIS as GOST 7.79-2000,
  with an "A" subsystem that is identical to the ISO standard
  and retains the one Cyrillic character to one Latin character
  mapping system. A subsystem "B" also exists that features mappings
  from single Cyrillic characters to multiple Latin characters,
  although this version is not covered by this program.

- The system stipulated by the International Civil Aviation
  Organization (ICAO) for machine readable travel documents,
  which has been used in Russian passports since 2013
  according to Order No. 320 of Russia's Federal Migratory Service.

These transliteration systems are summarised in a table of character
mappings on Wikipedia, which can be found here:
https://en.wikipedia.org/wiki/Romanization_of_Russian

More information on the International Scholarly System can be found at:
https://en.wikipedia.org/wiki/Scientific_transliteration_of_Cyrillic

More information on ISO 9:1995 can be found at:
https://www.iso.org/standard/3589.html

More information on the ICAO system can be found at:
https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf

This program comprises the following steps:
    - Input the string for transliteration
    - Input the code for the transliteration system to use

After the final step, you will have the option of entering a code
for a different transliteration system to use on the same string,
or to start from scratch by entering a new string for processing.
Alternatively, you will also be able to exit the program.
""")

# The program's main loop is initiated here using a while loop
# in conjunction with a sentry variable called loop_string_selection.
# If the user does not enter "N" as a value for this variable
# at the end when asked if they wish to end the program,
# the program flow reverts back here and repeats the steps,
# allowing them to enter a new string for conversion.
loop_string_selection = ""
while loop_string_selection != "N":

    # The user's first prompt is printed here, which asks them to enter
    # a string of text for transliteration.  This could be,
    # for example, the Russian pangram "Разъяренный чтец эгоистично
    # бьёт пятью жердями шустрого фехтовальщика."
    input_string = input("Please enter a string for processing: ")

    # Since strings are immutable objects in Python, the string
    # is converted into a mutable list format for ease of modification
    # and processing.  The list() function breaks the string down
    # into a list with its characters (including spaces and punctuation
    # marks) stored as individual items.  This list is then assigned
    # to the string_as_list variable.
    string_as_list = list(input_string)

    # A secondary loop is initiated here, this time using
    # a sentry variable called loop_dict_selection.
    # If the user does not enter "N" as a value for this variable
    # at the end when asked if they wish to enter a new string
    # for conversion or end the program, the program flow
    # reverts back here to repeat the last step,
    # allowing them to select a new transliteration system
    # for converting the string that was previously entered.
    loop_dict_selection = ""
    while loop_dict_selection != "N":

        # Mapping dictionaries for the various transliteration systems
        # are established here, along with the codes that are used
        # to refer to them and to select them.  The relevant dictionary
        # is accessed later in the program to return Latin character
        # sets (in the value entry) for each identified Cyrillic
        # character (in the key entry).
        #
        # This dictionary has been created for the International
        # Scholarly System.  Note that since double quotation marks
        # are used for the strings here, a backslash is required to
        # escape the double quotation mark used to represent
        # the hard sign "ъ" in this system.  Also note that
        # this system uses one-to-two character mappings
        # in a few cases.
        scholarly_code = "SCHOLARLY"
        scholarly_dict = {"А": "A", "а": "a",
                          "Б": "B", "б": "b",
                          "В": "V", "в": "v",
                          "Г": "G", "г": "g",
                          "Д": "D", "д": "d",
                          "Е": "E", "е": "e",
                          "Ё": "Ë", "ё": "ë",
                          "Ж": "Ž", "ж": "ž",
                          "З": "Z", "з": "z",
                          "И": "I", "и": "i",
                          "Й": "J", "й": "j",
                          "К": "K", "к": "k",
                          "Л": "L", "л": "l",
                          "М": "M", "м": "m",
                          "Н": "N", "н": "n",
                          "О": "O", "о": "o",
                          "П": "P", "п": "p",
                          "Р": "R", "р": "r",
                          "С": "S", "с": "s",
                          "Т": "T", "т": "t",
                          "У": "U", "у": "u",
                          "Ф": "F", "ф": "f",
                          "Х": "X", "х": "x",
                          "Ц": "C", "ц": "c",
                          "Ч": "Č", "ч": "č",
                          "Ш": "Š", "ш": "š",
                          "Щ": "Šč", "щ": "šč",
                          "Ъ": "\"", "ъ": "\"",
                          "Ы": "Y", "ы": "y",
                          "Ь": "'", "ь": "'",
                          "Э": "È", "э": "è",
                      "Ю": "Ju", "ю": "ju",
                      "Я": "Ja", "я": "ja"}

    # This dictionary has been created for the ISO 9:1995/
    # GOST 7.79-2000(A) system.  Note that since double quotation
    # marks are used for the strings here, a backslash is required
    # to escape the double quotation mark used to represent
    # the hard sign "ъ" in this system.  This system is notable
    # for consisting solely of one-to-one character mappings,
    # which would mean that text romanised with this system
    # would be quite easy to back-transliterate as well
    # using a similar method.
    ISO_9_code = "ISO9"
    ISO_9_dict = {"А": "A", "а": "a",
                  "Б": "B", "б": "b",
                  "В": "V", "в": "v",
                  "Г": "G", "г": "g",
                  "Д": "D", "д": "d",
                  "Е": "E", "е": "e",
                  "Ё": "Ë", "ё": "ë",
                  "Ж": "Ž", "ж": "ž",
                  "З": "Z", "з": "z",
                  "И": "I", "и": "i",
                  "Й": "J", "й": "j",
                  "К": "K", "к": "k",
                  "Л": "L", "л": "l",
                  "М": "M", "м": "m",
                  "Н": "N", "н": "n",
                  "О": "O", "о": "o",
                  "П": "P", "п": "p",
                  "Р": "R", "р": "r",
                  "С": "S", "с": "s",
                  "Т": "T", "т": "t",
                  "У": "U", "у": "u",
                  "Ф": "F", "ф": "f",
                  "Х": "H", "х": "h",
                  "Ц": "C", "ц": "c",
                  "Ч": "Č", "ч": "č",
                  "Ш": "Š", "ш": "š",
                  "Щ": "Ŝ", "щ": "ŝ",
                  "Ъ": "\"", "ъ": "\"",
                  "Ы": "Y", "ы": "y",
                  "Ь": "'", "ь": "'",
                  "Э": "È", "э": "è",
                  "Ю": "Û", "ю": "û",
                  "Я": "Â", "я": "â"}

    # This dictionary has been created for the ICAO system
    # for producing machine readable travel documents.
    # This system is notable for not using any diacritics,
    # so there is greater reliance on one-to-multiple character
    # mappings instead.  The hard sign "ъ" is transliterated
    # more unusually as "ie", while the soft sign "ь" is not
    # reflected in the transliteration at all in this case,
    # and so it returns an empty string (i.e. no character) here.
    ICAO_code = "ICAO"
    ICAO_dict = {"А": "A", "а": "a",
                 "Б": "B", "б": "b",
                 "В": "V", "в": "v",
                 "Г": "G", "г": "g",
                 "Д": "D", "д": "d",
                 "Е": "E", "е": "e",
                 "Ё": "E", "ё": "e",
                 "Ж": "Zh", "ж": "zh",
                 "З": "Z", "з": "z",
                 "И": "I", "и": "i",
                 "Й": "I", "й": "i",
                 "К": "K", "к": "k",
                 "Л": "L", "л": "l",
                 "М": "M", "м": "m",
                 "Н": "N", "н": "n",
                 "О": "O", "о": "o",
                 "П": "P", "п": "p",
                 "Р": "R", "р": "r",
                 "С": "S", "с": "s",
                 "Т": "T", "т": "t",
                 "У": "U", "у": "u",
                 "Ф": "F", "ф": "f",
                 "Х": "Kh", "х": "kh",
                 "Ц": "Ts", "ц": "ts",
                 "Ч": "Ch", "ч": "ch",
                 "Ш": "Sh", "ш": "sh",
                 "Щ": "Shch", "щ": "shch",
                 "Ъ": "Ie", "ъ": "ie",
                 "Ы": "Y", "ы": "y",
                 "Ь": "", "ь": "",
                 "Э": "E", "э": "e",
                 "Ю": "Iu", "ю": "iu",
                 "Я": "Ia", "я": "ia"}

    # The user's second prompt is printed here, with instructions
    # printed before the input prompt itself that ask the user
    # to enter the corresponding code for the transliteration
    # system to be used.  The variables that contain the strings
    # that will be matched against the user's input
    # are concatenated to the instruction string in this case.
    #
    # The input itself is framed within an infinite while loop
    # that restricts input in a similar way to the y_or_n_input()
    # function defined above: the user can only enter one
    # of the specified codes (with the upper() method applied,
    # to remove case sensitivity) to trigger the if condition
    # that will break out of the while loop and continue
    # the program.  If a non-matching string is entered,
    # the else condition will continue the loop until a valid
    # input is entered.
    while True:
        print("\nPlease enter the relevant code following the colon "
              "for one of the transliteration systems below "
              "for converting the text."
              "\nInternational Scholarly System: "
              + scholarly_code +
              "\nISO 9:1995/GOST 7.79-2000(A): "
              + ISO_9_code +
              "\nICAO MRTD specification: "
              + ICAO_code)
        input_code = input("Enter code: ")
        if (input_code.upper() == scholarly_code
           or input_code.upper() == ISO_9_code
           or input_code.upper() == ICAO_code):
            break
        else:
            print("The code entered was invalid. Please try again.")
            continue
    # With a valid input entered, a series of (el)if conditions
    # are defined that assign the relevant mapping dictionary
    # to the chosen_dict variable depending on the code
    # that was entered.
```

```python
if input_code.upper() == scholarly_code:
    chosen_dict = scholarly_dict
elif input_code.upper() == ISO_9_code:
    chosen_dict = ISO_9_dict
elif input_code.upper() == ICAO_code:
    chosen_dict = ICAO_dict

# The crux of the program occurs here in the form of a list
# comprehension expression that assigns the modified characters
# of the string (along with any unchanged characters)
# to a new variable named modified_list.  The expression itself
# makes use of Python's get() method to access the chosen
# dictionary and return the Latin equivalents
# in this dictionary for any Russian Cyrillic characters
# that appear.  The program does this by iterating
# over each item in string_as_list, which in this
# case is each character extracted from the original string.
# Each item is assigned to a temporary variable, char,
# for each iteration of the for loop.  By passing this char
# variable as the first parameter for the get() method,
# the method will return the value for any dictionary key
# that matches the string contained in this variable
# should this string be a Russian Cyrillic character, and add
# this value to the new list.  The second parameter indicates
# the value that should be returned if no dictionary key
# is found, and in this case, the char variable is passed
# again, meaning that the character will be simply be re-added
# to the new list unmodified if no matching dictionary key
# is found.  This creates a new list in the modified_list
# variable with the same characters in the same order as the
# string_as_list variable, but with only the Russian Cyrillic
# characters changed.
modified_list = [chosen_dict.get(char, char)
                 for char in string_as_list]

# The items in modified_list are then combined back
# into a single string using the join() method, with an empty
# string as a separator to ensure that no spaces are inserted
# between the characters.
modified_string = "".join(modified_list)

# Finally, the results of the program are printed:
# The original string is printed for reference,
# along with the modified string and the reference code
# for the system that was used to transliterate it.
print("\nOrginal string:\n" + input_string)
print("Transliteration of string using the "
      + input_code.upper() + " system:\n"
      + modified_string)

# With the final step of the program concluded, the user
# is asked to enter "Y" or "N" according to the earlier
# defined input function to determine whether they would
# like to process the same string again with a different
# system (i.e. revert to the beginning of the secondary
# loop), or choose to enter a new string or exit the program
# (i.e. exit the secondary loop and proceed to the end
# of the main loop).
loop_dict_selection = y_or_n_input("To process the same string again "
                                   "using a different "
                                   "transliteration system, "
                                   "please enter the letter Y."
                                   "\nTo process a different string "
                                   "or exit the program, "
```

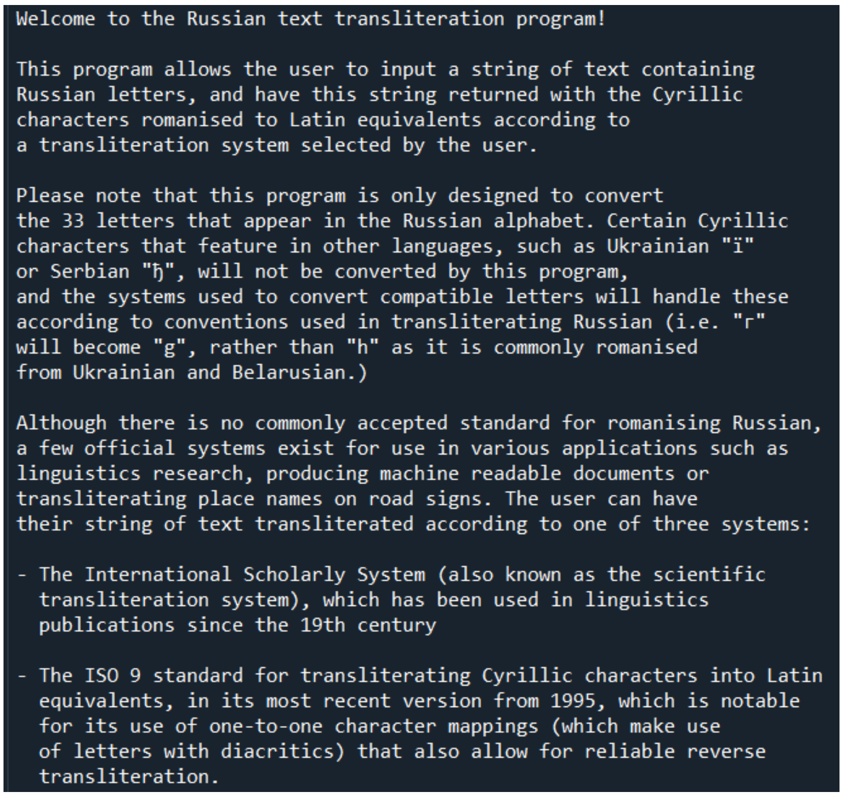
```
                                                       "please enter the letter N."
                                                       "\nEnter Y/N: ")

        # Having entered "N" to exit the secondary loop above,
        # the user is asked to enter "Y" or "N" according to
        # the earlier defined input function to determine whether
        # they would like to enter a new string for processing
        # (i.e. revert to the beginning of the main loop) or exit
        # the program (i.e exit the main loop and proceed to the end
        # of the program).
        loop_string_selection = y_or_n_input("To enter a new string "
                                             "for processing, "
                                             "please enter the letter Y."
                                             "\nTo exit the program, "
                                             "please enter the letter N."
                                             "\nEnter Y/N: ")

# With the program's main loop ended, the user is simply prompted
# to press enter to end the program.
input("Thank you for trying out this program. "
      "Press enter to exit.")
```

Output Examples

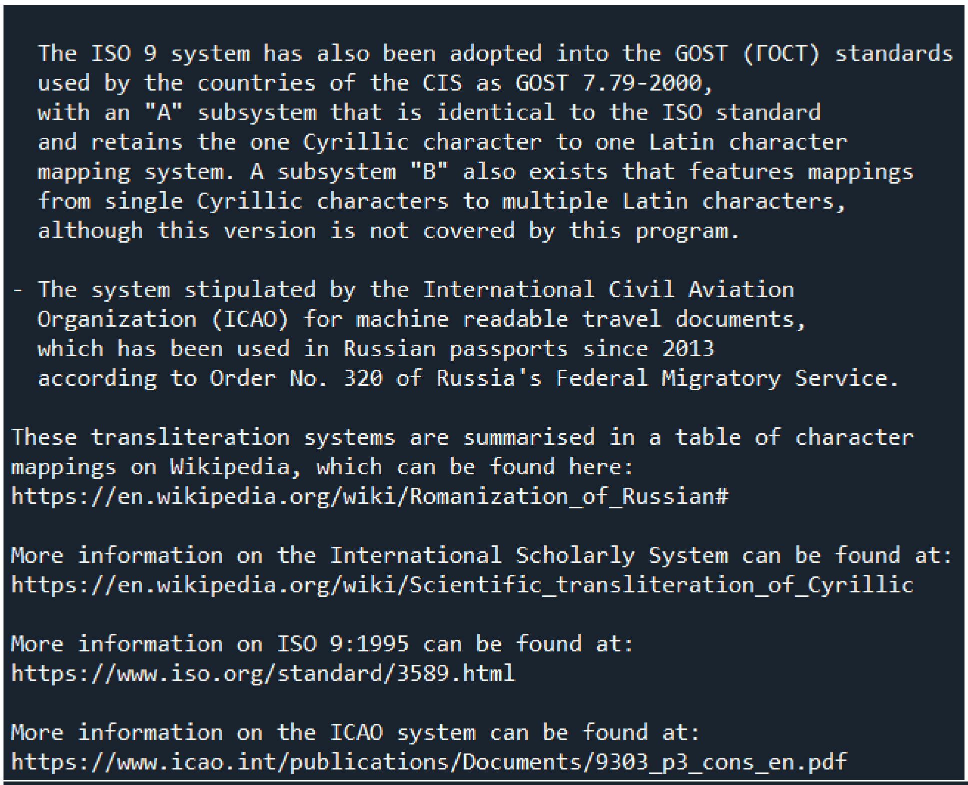
Test 1:

For the first test run of the program, a pangram was used that incorporates all 33 letters of the Russian alphabet. This string was tested with all three transliteration systems.

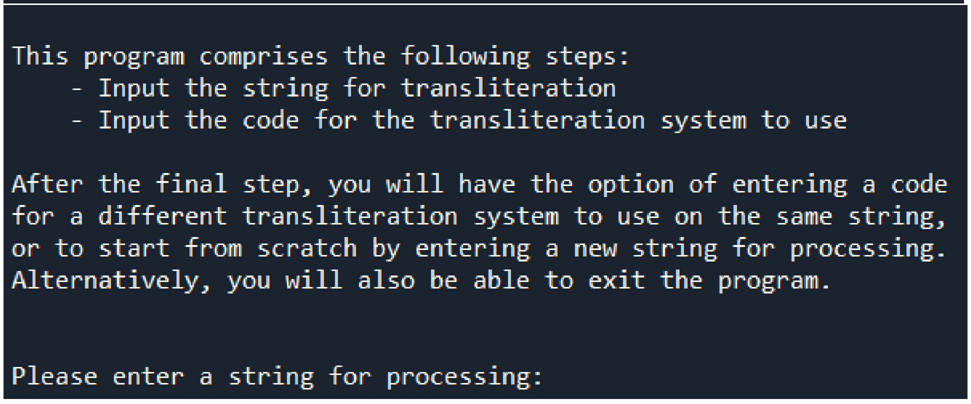
Input string:
"Разъяренный чтец эгоистично бьёт пятью жердями шустрого фехтовальщика."
(Translation: "An enraged narrator selfishly beats a nimble fencer with five poles.")



```
Welcome to the Russian text transliteration program!

This program allows the user to input a string of text containing
Russian letters, and have this string returned with the Cyrillic
characters romanised to Latin equivalents according to
a transliteration system selected by the user.

Please note that this program is only designed to convert
the 33 letters that appear in the Russian alphabet. Certain Cyrillic
characters that feature in other languages, such as Ukrainian "ї"
or Serbian "ђ", will not be converted by this program,
and the systems used to convert compatible letters will handle these
according to conventions used in transliterating Russian (i.e. "г"
will become "g", rather than "h" as it is commonly romanised
from Ukrainian and Belarusian.)

Although there is no commonly accepted standard for romanising Russian,
a few official systems exist for use in various applications such as
linguistics research, producing machine readable documents or
transliterating place names on road signs. The user can have
their string of text transliterated according to one of three systems:

- The International Scholarly System (also known as the scientific
  transliteration system), which has been used in linguistics
  publications since the 19th century

- The ISO 9 standard for transliterating Cyrillic characters into Latin
  equivalents, in its most recent version from 1995, which is notable
  for its use of one-to-one character mappings (which make use
  of letters with diacritics) that also allow for reliable reverse
  transliteration.
```

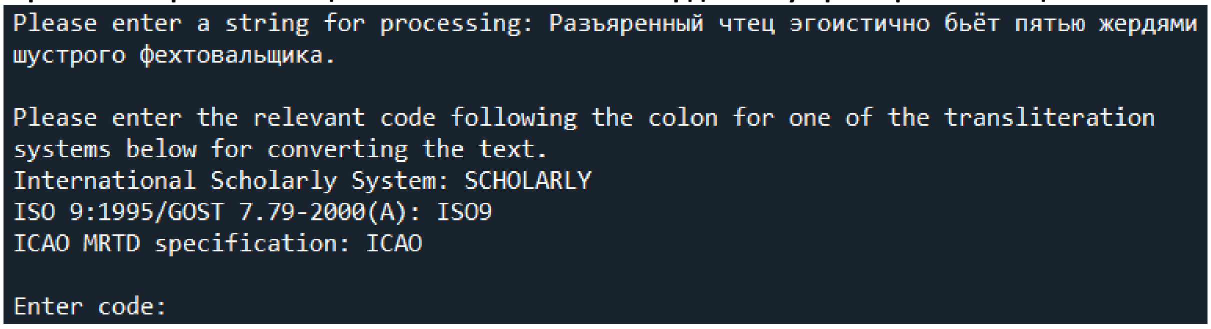


```
  The ISO 9 system has also been adopted into the GOST (ГОСТ) standards
  used by the countries of the CIS as GOST 7.79-2000,
  with an "A" subsystem that is identical to the ISO standard
  and retains the one Cyrillic character to one Latin character
  mapping system. A subsystem "B" also exists that features mappings
  from single Cyrillic characters to multiple Latin characters,
  although this version is not covered by this program.

- The system stipulated by the International Civil Aviation
  Organization (ICAO) for machine readable travel documents,
  which has been used in Russian passports since 2013
  according to Order No. 320 of Russia's Federal Migratory Service.

These transliteration systems are summarised in a table of character
mappings on Wikipedia, which can be found here:
https://en.wikipedia.org/wiki/Romanization_of_Russian#

More information on the International Scholarly System can be found at:
https://en.wikipedia.org/wiki/Scientific_transliteration_of_Cyrillic

More information on ISO 9:1995 can be found at:
https://www.iso.org/standard/3589.html

More information on the ICAO system can be found at:
https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf
```



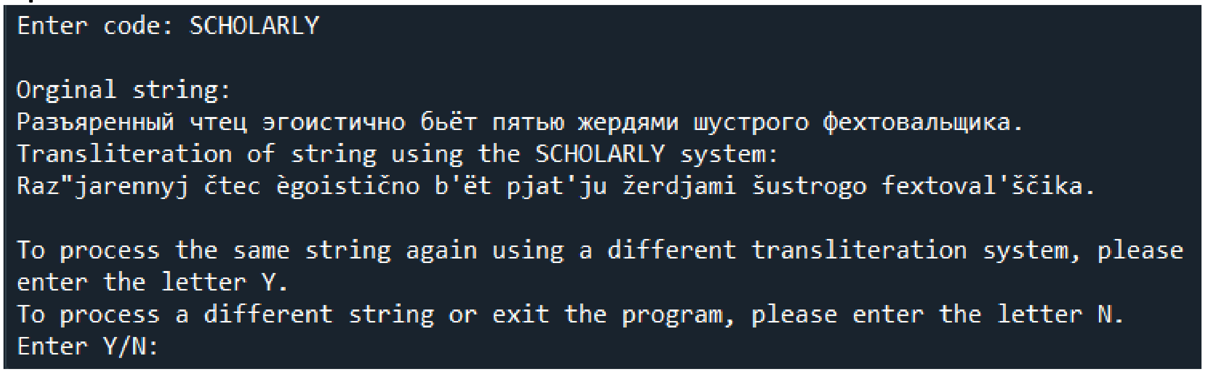
```
This program comprises the following steps:
    - Input the string for transliteration
    - Input the code for the transliteration system to use

After the final step, you will have the option of entering a code
for a different transliteration system to use on the same string,
or to start from scratch by entering a new string for processing.
Alternatively, you will also be able to exit the program.


Please enter a string for processing:
```

Input: "Разъяренный чтец эгоистично бьёт пятью жердями шустрого фехтовальщика."



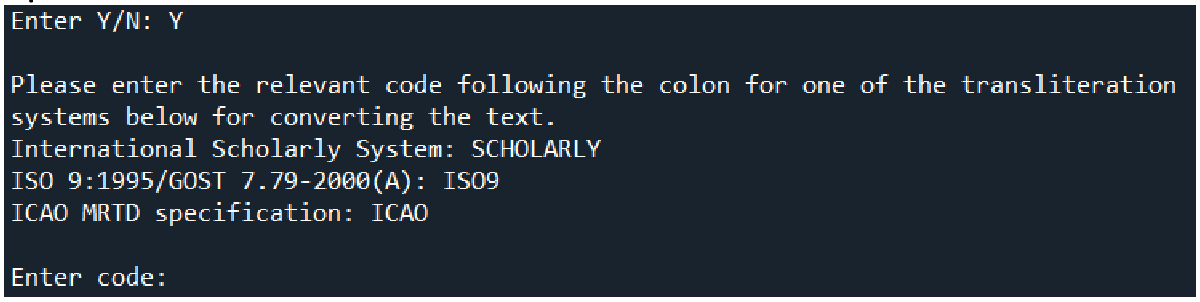
```
Please enter a string for processing: Разъяренный чтец эгоистично бьёт пятью жердями
шустрого фехтовальщика.

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "SCHOLARLY"



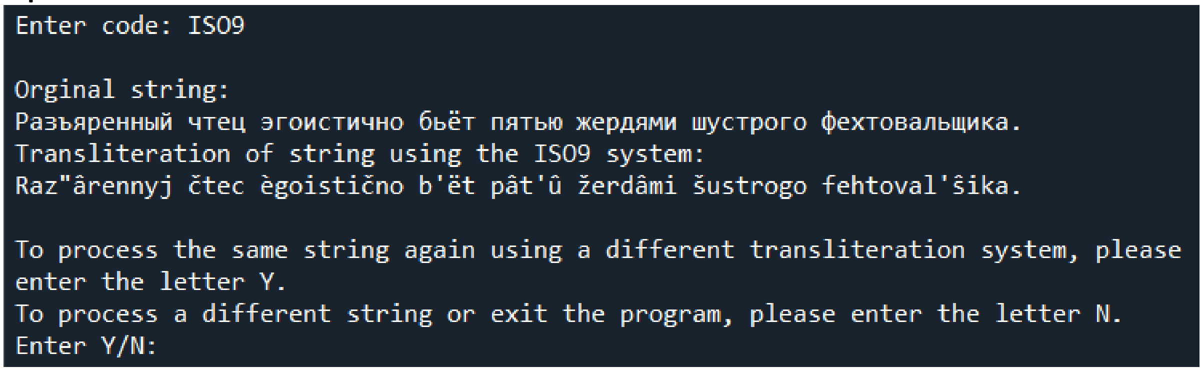
```
Enter code: SCHOLARLY

Orginal string:
Разъяренный чтец эгоистично бьёт пятью жердями шустрого фехтовальщика.
Transliteration of string using the SCHOLARLY system:
Raz"jarennyj čtec ègoistično b'ët pjat'ju žerdjami šustrogo fextoval'ščika.

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "Y"



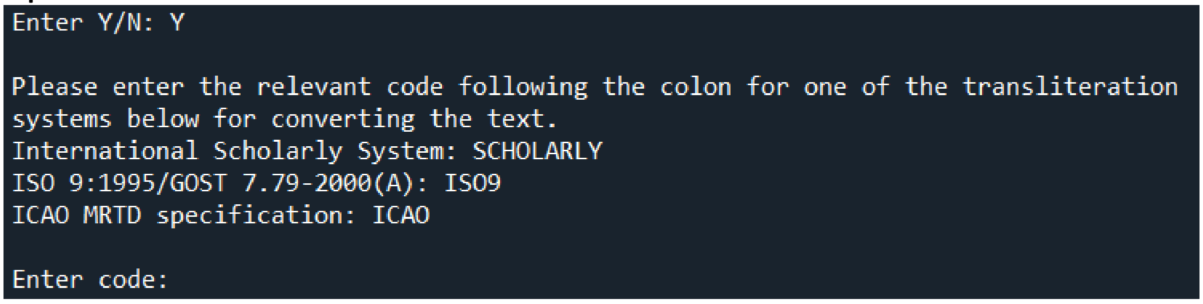
```
Enter Y/N: Y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "ISO9"



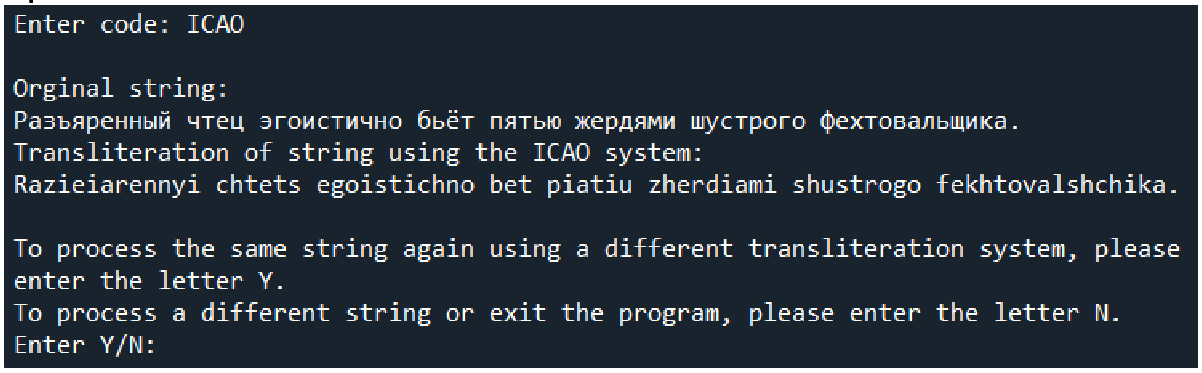
```
Enter code: ISO9

Orginal string:
Разъяренный чтец эгоистично бьёт пятью жердями шустрого фехтовальщика.
Transliteration of string using the ISO9 system:
Raz"ârennyj čtec ègoistično b'ët pât'û žerdâmi šustrogo fehtoval'ŝika.

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "Y"



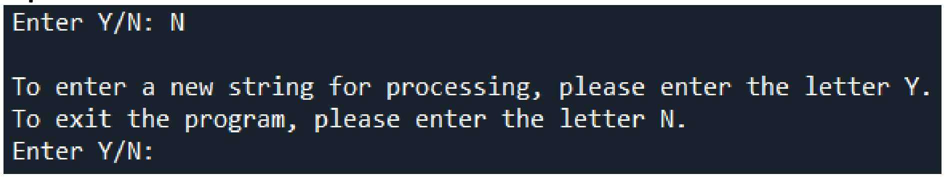
```
Enter Y/N: Y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "ICAO"



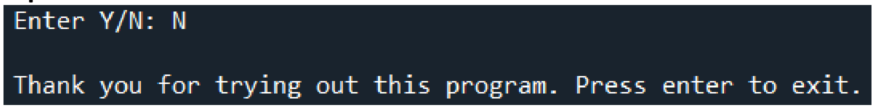
```
Enter code: ICAO

Orginal string:
Разъяренный чтец эгоистично бьёт пятью жердями шустрого фехтовальщика.
Transliteration of string using the ICAO system:
Razieiarennyi chtets egoistichno bet piatiu zherdiami shustrogo fekhtovalshchika.

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "N"



```
Enter Y/N: N

To enter a new string for processing, please enter the letter Y.
To exit the program, please enter the letter N.
Enter Y/N:
```

Input: "N"



```
Enter Y/N: N

Thank you for trying out this program. Press enter to exit.
```

Test 2:

For a second test run, a slightly more complex string was used that consists of some bibliographic information. This test was carried out to resemble a possible real-world use case for the program – namely, for the transliteration of bibliographic information containing names of authors and titles of works in Cyrillic, if these were hypothetically required to be romanised as part of a submission.

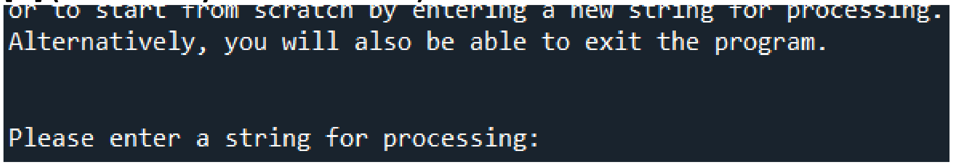
In this case, translations of the (arbitrarily selected) works have been provided alongside the Russian text in square brackets within the string itself, in part to demonstrate how these can be run through the program alongside the Cyrillic characters and left unchanged in the modified string.

Other elements have been purposefully included to demonstrate how the program deals with elements such as initials, abbreviations, digits, and various kinds of punctuation marks. The string also includes a line break that was copied into the input in the console.

It should be noted that for transliterating bibliographic information of this kind, a system such as the American Library Association – Library of Congress (ALA-LC) table for the romanisation of Russian (The Library of Congress, 2012) would normally be used, as opposed to the three systems presently incorporated into my program. This system is very similar to the ICAO system, however, and only handles a few letters differently, with ligature ties added to a few other combinations that remain the same (e.g. "t͡s"). In practice, if the program needed to be modified to include this system, it would be relatively simple to do this (see "Notes on Results and Potential for Further Modification" below), although additional code would need to written to handle the exception for the hard sign "ъ", which is normally omitted rather than converted to a double quotation mark when it appears at the end of a word in this system, as indicated in the notes to the multi-system transliteration table in Wikipedia's article on the romanisation of Russian (2021).

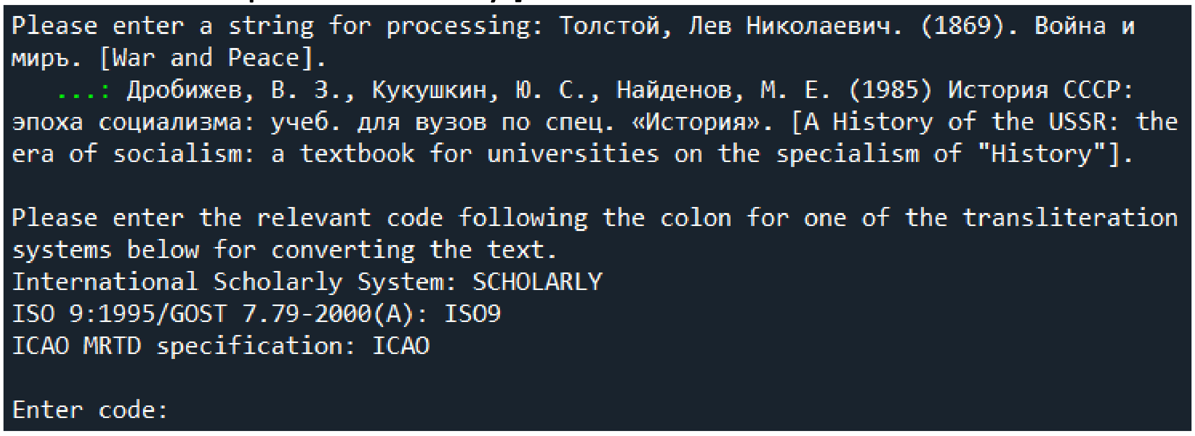
Input string:
"Толстой, Лев Николаевич. (1869). Война и миръ. [War and Peace].
Дробижев, В. З., Кукушкин, Ю. С., Найденов, М. Е. (1985) История СССР: эпоха социализма: учеб. для вузов по спец. «История». [A History of the USSR: the era of socialism: a textbook for universities on the specialism of "History"]."

[…] (Introductory text as for Test1)



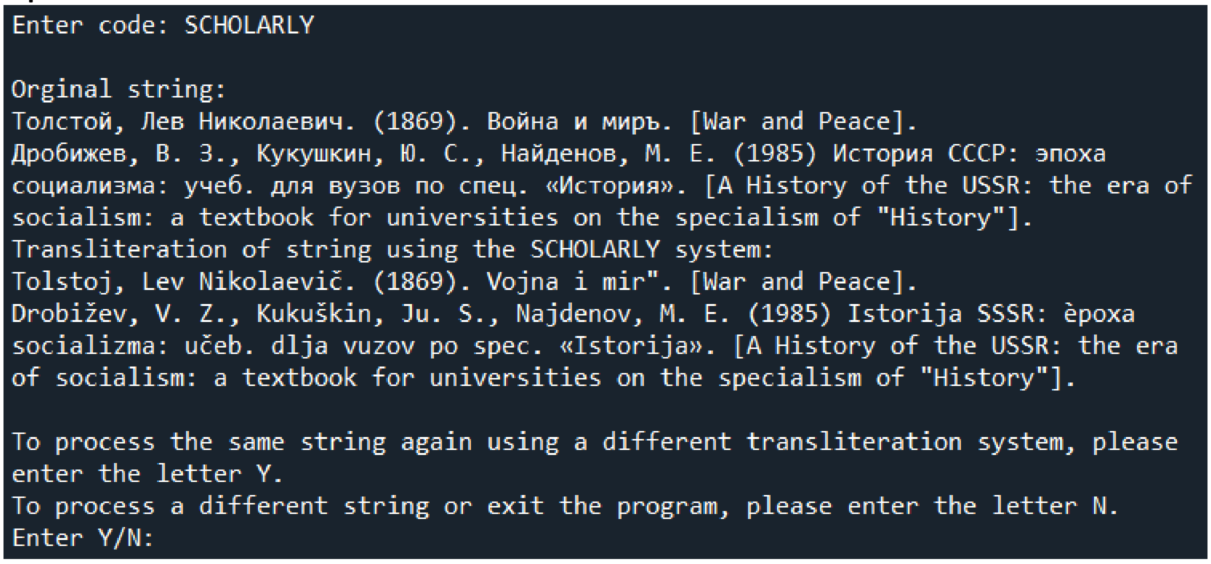
```
or to start from scratch by entering a new string for processing.
Alternatively, you will also be able to exit the program.


Please enter a string for processing:
```

Input: "Толстой, Лев Николаевич. (1869). Война и миръ. [War and Peace].
Дробижев, В. З., Кукушкин, Ю. С., Найденов, М. Е. (1985) История СССР: эпоха социализма: учеб. для вузов по спец. «История». [A History of the USSR: the era of socialism: a textbook for universities on the specialism of "History"]."



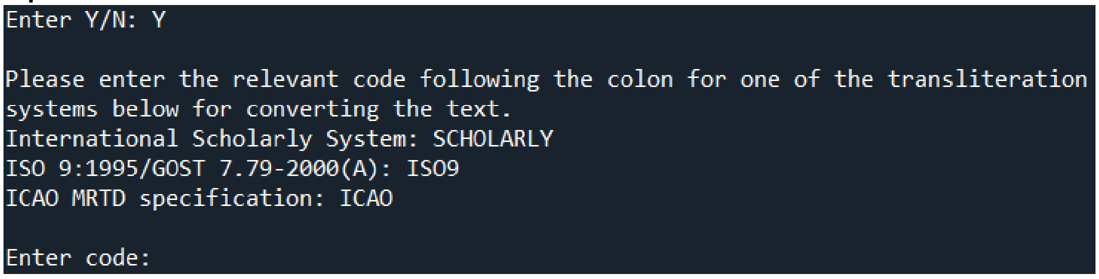
```
Please enter a string for processing: Толстой, Лев Николаевич. (1869). Война и
миръ. [War and Peace].
   ...: Дробижев, В. З., Кукушкин, Ю. С., Найденов, М. Е. (1985) История СССР:
эпоха социализма: учеб. для вузов по спец. «История». [A History of the USSR: the
era of socialism: a textbook for universities on the specialism of "History"].

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "SCHOLARLY"



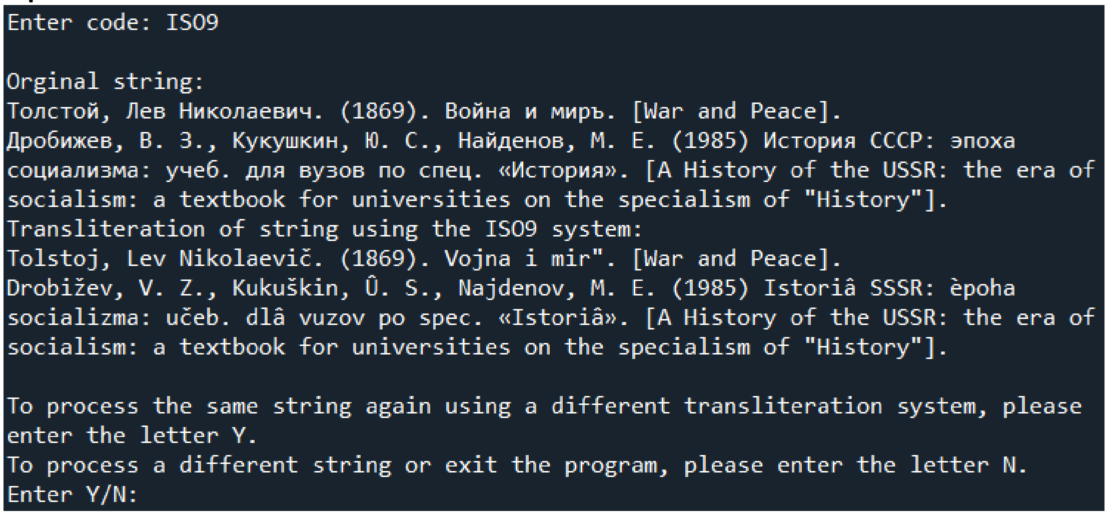
```
Enter code: SCHOLARLY

Orginal string:
Толстой, Лев Николаевич. (1869). Война и миръ. [War and Peace].
Дробижев, В. З., Кукушкин, Ю. С., Найденов, М. Е. (1985) История СССР: эпоха
социализма: учеб. для вузов по спец. «История». [A History of the USSR: the era of
socialism: a textbook for universities on the specialism of "History"].
Transliteration of string using the SCHOLARLY system:
Tolstoj, Lev Nikolaevič. (1869). Vojna i mir". [War and Peace].
Drobižev, V. Z., Kukuškin, Ju. S., Najdenov, M. E. (1985) Istorija SSSR: èpoxa
socializma: učeb. dlja vuzov po spec. «Istorija». [A History of the USSR: the era
of socialism: a textbook for universities on the specialism of "History"].

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "Y"



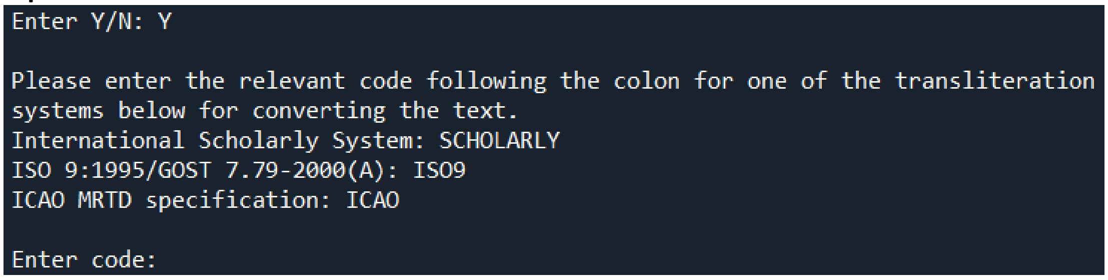
```
Enter Y/N: Y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "ISO9"



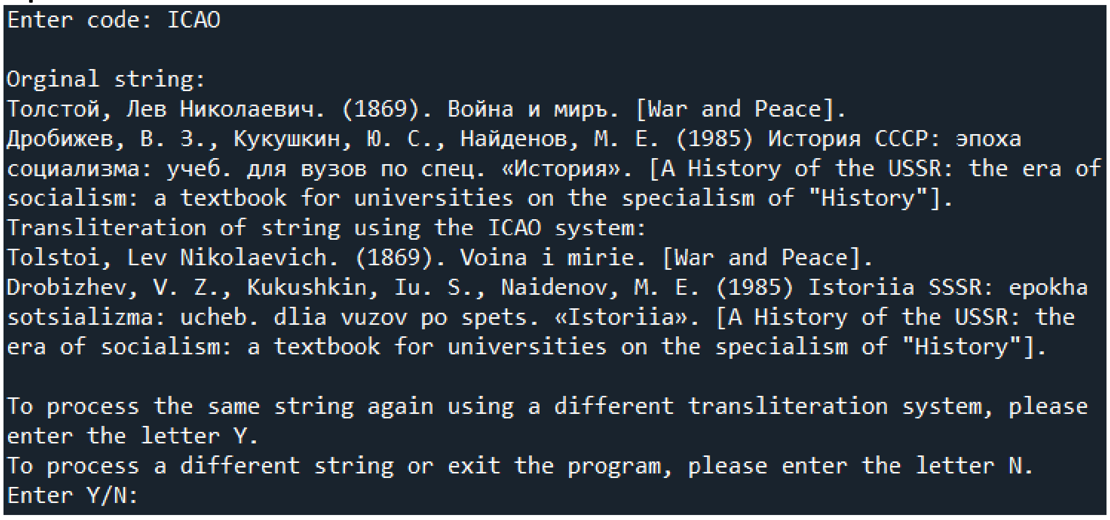
```
Enter code: ISO9

Orginal string:
Толстой, Лев Николаевич. (1869). Война и миръ. [War and Peace].
Дробижев, В. З., Кукушкин, Ю. С., Найденов, М. Е. (1985) История СССР: эпоха
социализма: учеб. для вузов по спец. «История». [A History of the USSR: the era of
socialism: a textbook for universities on the specialism of "History"].
Transliteration of string using the ISO9 system:
Tolstoj, Lev Nikolaevič. (1869). Vojna i mir". [War and Peace].
Drobižev, V. Z., Kukuškin, Û. S., Najdenov, M. E. (1985) Istoriâ SSSR: èpoha
socializma: učeb. dlâ vuzov po spec. «Istoriâ». [A History of the USSR: the era of
socialism: a textbook for universities on the specialism of "History"].

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "Y"



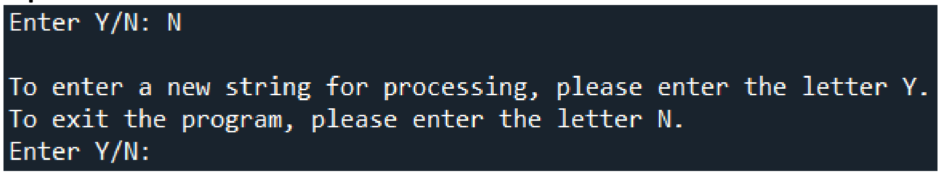
```
Enter Y/N: Y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "ICAO"



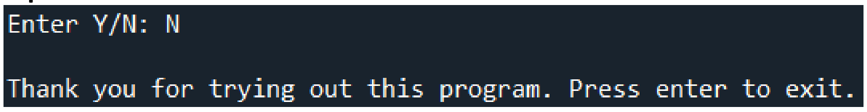
```
Enter code: ICAO

Orginal string:
Толстой, Лев Николаевич. (1869). Война и миръ. [War and Peace].
Дробижев, В. З., Кукушкин, Ю. С., Найденов, М. Е. (1985) История СССР: эпоха
социализма: учеб. для вузов по спец. «История». [A History of the USSR: the era of
socialism: a textbook for universities on the specialism of "History"].
Transliteration of string using the ICAO system:
Tolstoi, Lev Nikolaevich. (1869). Voina i mirie. [War and Peace].
Drobizhev, V. Z., Kukushkin, Iu. S., Naidenov, M. E. (1985) Istoriia SSSR: epokha
sotsializma: ucheb. dlia vuzov po spets. «Istoriia». [A History of the USSR: the
era of socialism: a textbook for universities on the specialism of "History"].

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "N"



```
Enter Y/N: N

To enter a new string for processing, please enter the letter Y.
To exit the program, please enter the letter N.
Enter Y/N:
```

Input: "N"



```
Enter Y/N: N

Thank you for trying out this program. Press enter to exit.
```

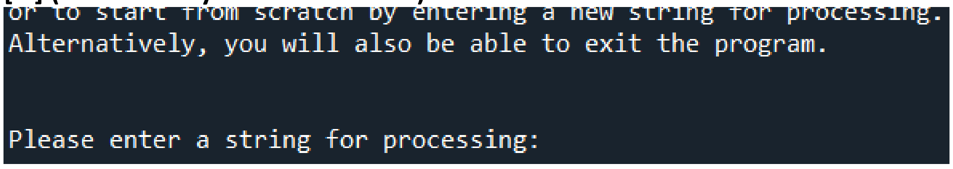
Test 3:

The final test of the program was mainly carried out to show how the functions and code blocks for restricting input work, which prevent the program flow from continuing until the user has entered a valid input string. Inputs aside from the actual strings for testing were given in lowercase this time, to additionally demonstrate how the program is case-insensitive in this regard.

Two different strings have been tested here to also demonstrate how the program can be used to test multiple inputs in a single session. The input strings used in this case are a couple of abbreviations, which have primarily been chosen to show how the program handles these according to the various systems – particularly with regard to letters such as "Ж" and "Х", which can occasionally present an issue for the transliteration of abbreviations due to these letters often being rendered with multiple characters in the Latin alphabet.

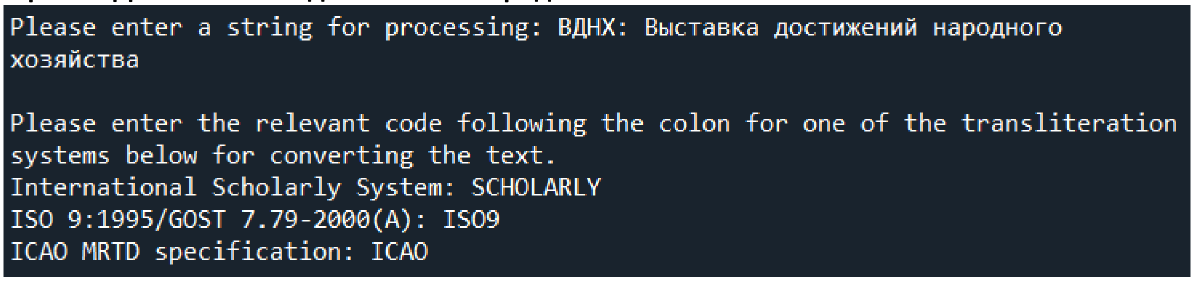
Input string 1:
"ВДНХ: Выставка достижений народного хозяйства"
(Translation: "VDNKh: Exhibition of Achievements of National Economy")
Input string 2:
"ЖЖ: Живой Журнал"
(Translation: "ZhZh: LiveJournal")

[…] (Introductory text as for Test1)



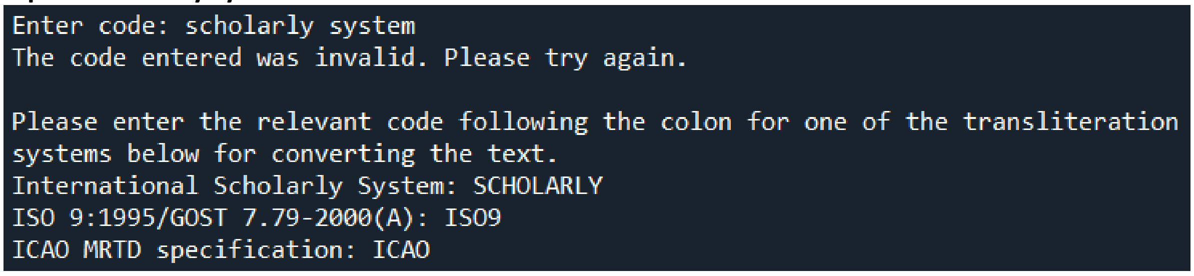
```
or to start from scratch by entering a new string for processing.
Alternatively, you will also be able to exit the program.


Please enter a string for processing:
```

Input: "ВДНХ: Выставка достижений народного хозяйства"



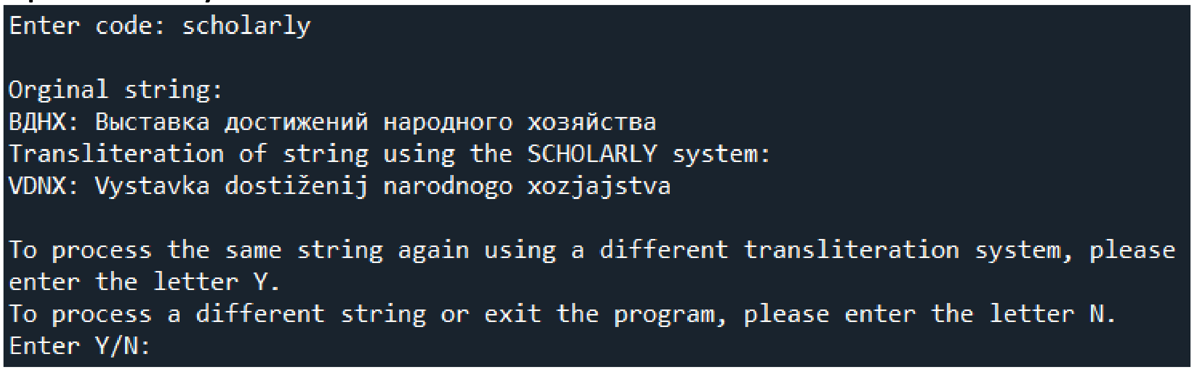
```
Please enter a string for processing: ВДНХ: Выставка достижений народного
хозяйства

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO
```

Input: "scholarly system"



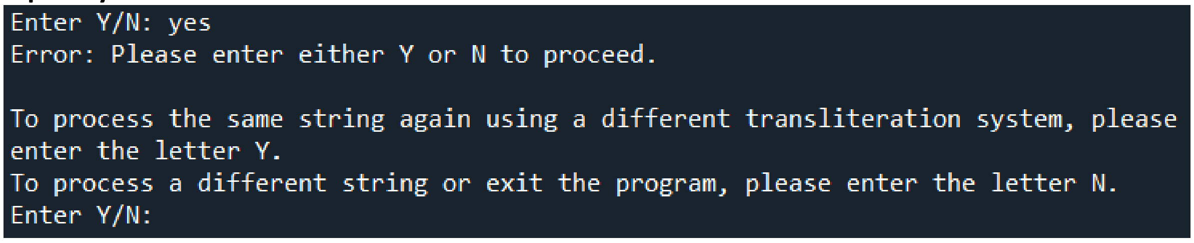
```
Enter code: scholarly system
The code entered was invalid. Please try again.

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO
```

Input: "schorlarly"



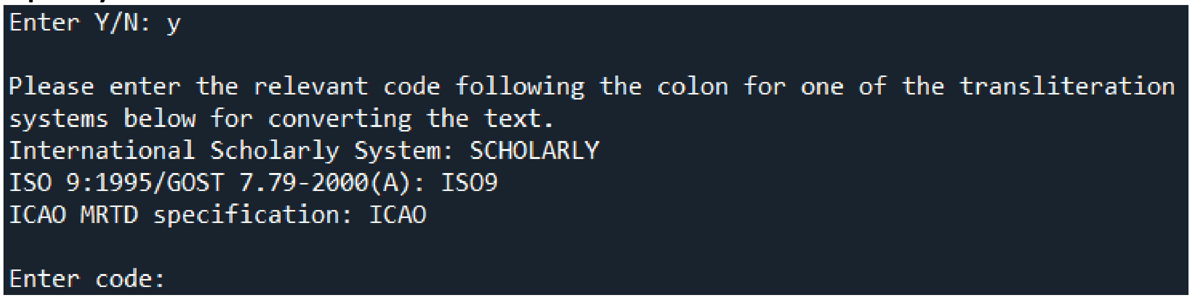
```
Enter code: scholarly

Orginal string:
ВДНХ: Выставка достижений народного хозяйства
Transliteration of string using the SCHOLARLY system:
VDNX: Vystavka dostiženij narodnogo xozjajstva

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "yes"



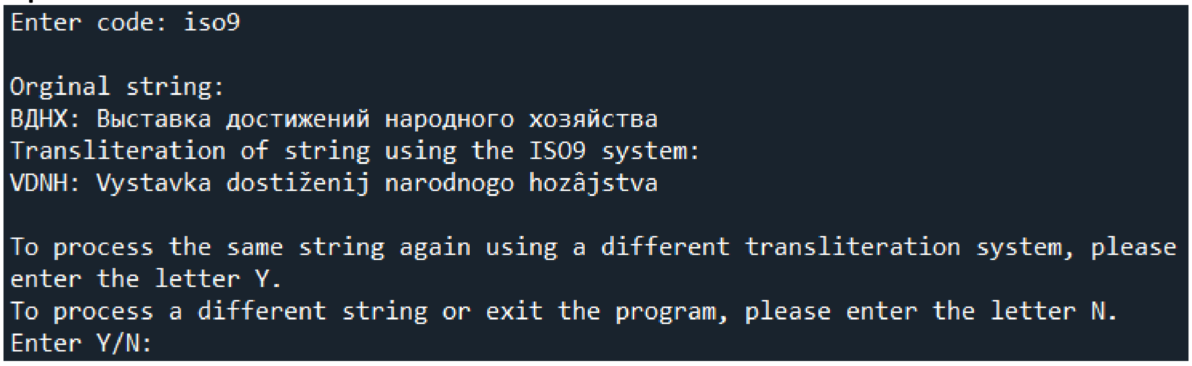
```
Enter Y/N: yes
Error: Please enter either Y or N to proceed.

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "y"



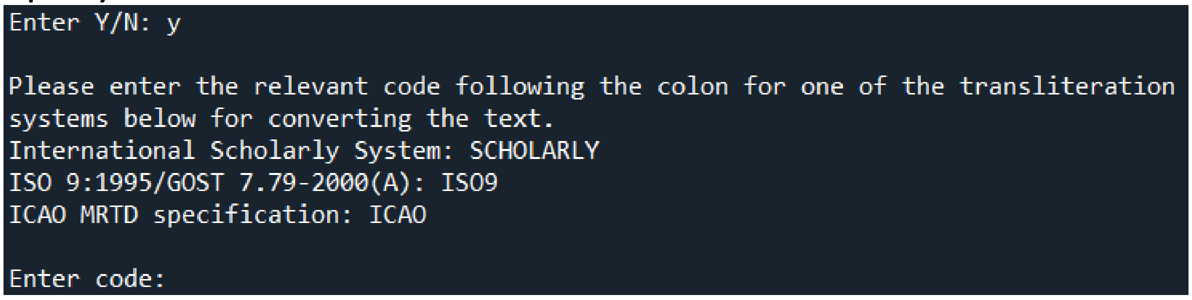
```
Enter Y/N: y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "iso9"



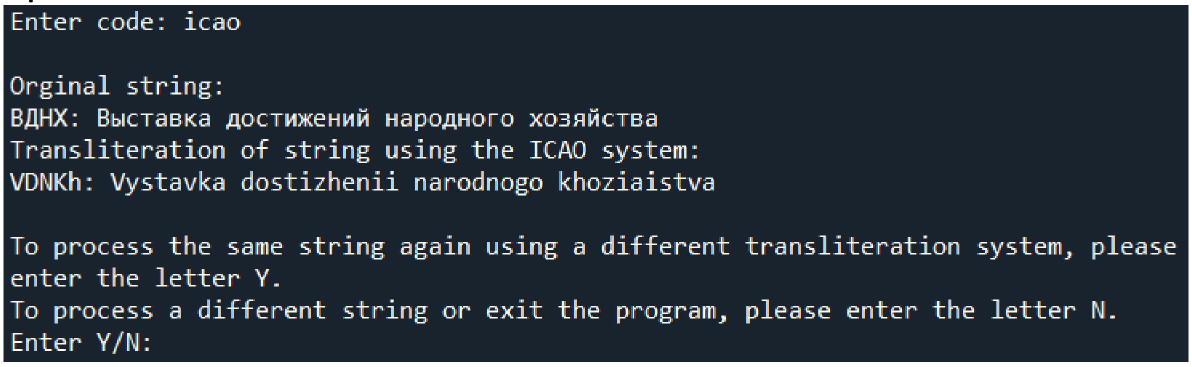
```
Enter code: iso9

Orginal string:
ВДНХ: Выставка достижений народного хозяйства
Transliteration of string using the ISO9 system:
VDNH: Vystavka dostiženij narodnogo hozâjstva

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "y"



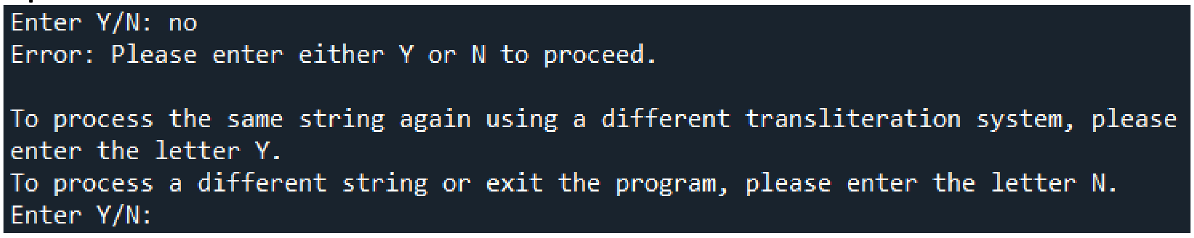
```
Enter Y/N: y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "icao"



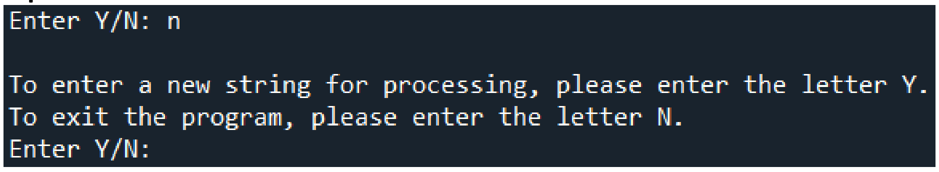
```
Enter code: icao

Orginal string:
ВДНХ: Выставка достижений народного хозяйства
Transliteration of string using the ICAO system:
VDNKh: Vystavka dostizhenii narodnogo khoziaistva

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "no"



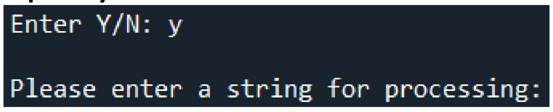
```
Enter Y/N: no
Error: Please enter either Y or N to proceed.

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "n"



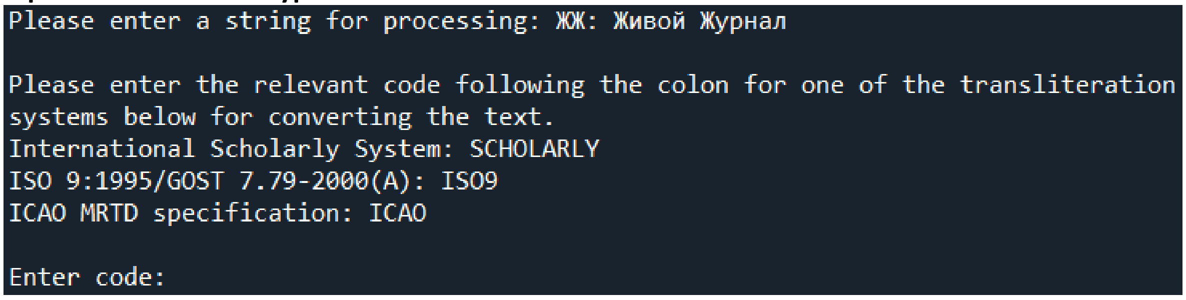
```
Enter Y/N: n

To enter a new string for processing, please enter the letter Y.
To exit the program, please enter the letter N.
Enter Y/N:
```

Input: "y"



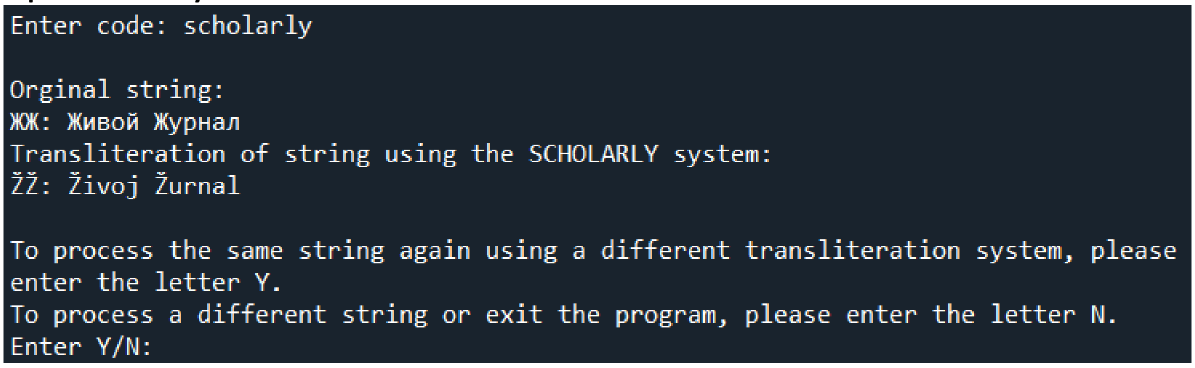
```
Enter Y/N: y

Please enter a string for processing:
```

Input: "ЖЖ: Живой Журнал"



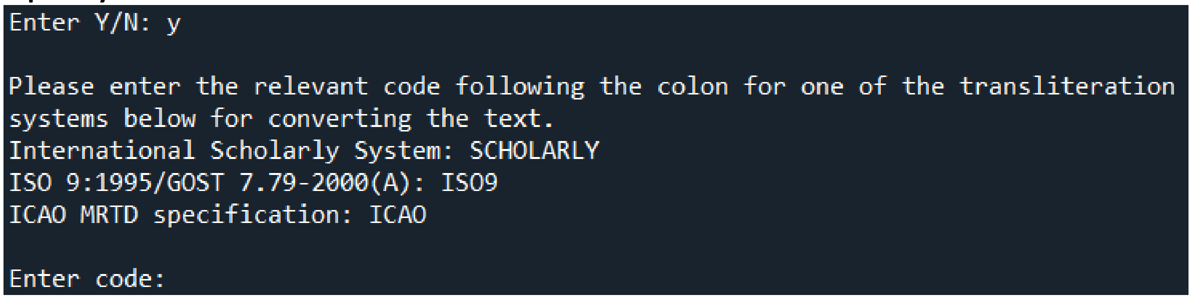
```
Please enter a string for processing: ЖЖ: Живой Журнал

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "scholarly"



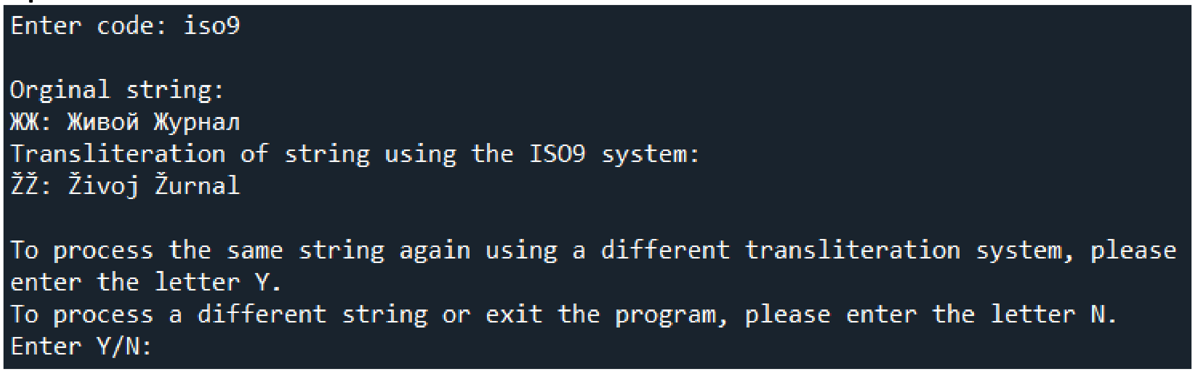
```
Enter code: scholarly

Orginal string:
ЖЖ: Живой Журнал
Transliteration of string using the SCHOLARLY system:
ŽŽ: Živoj Žurnal

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "y"



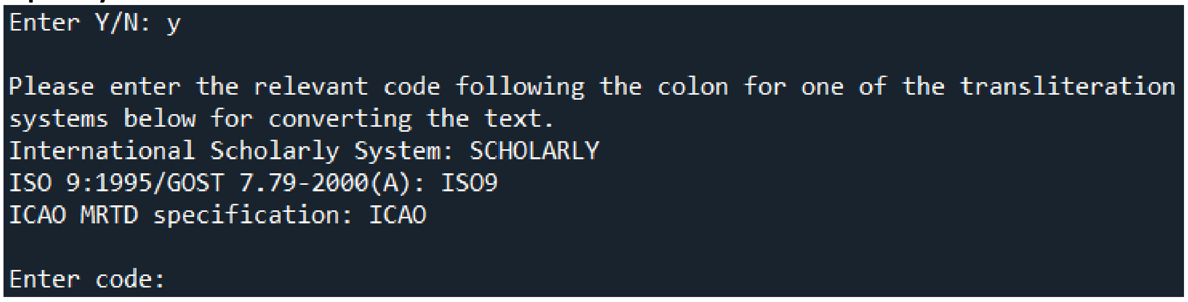
```
Enter Y/N: y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "iso9"



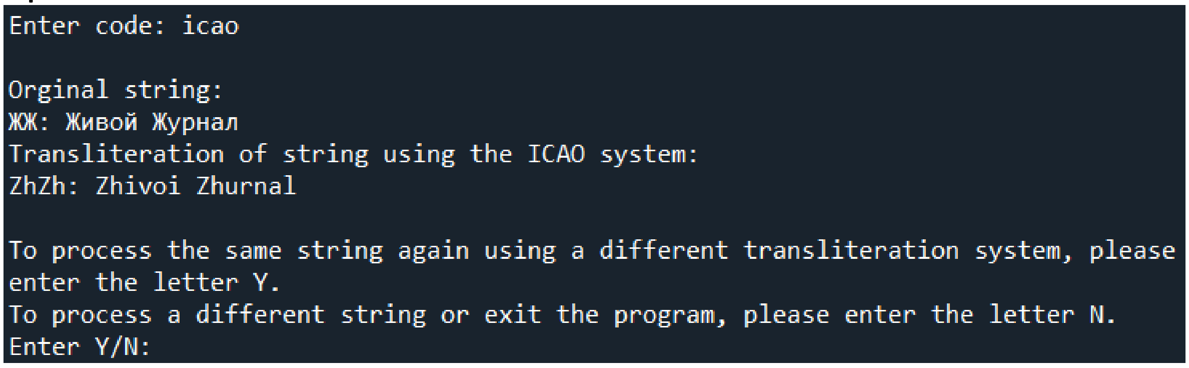
```
Enter code: iso9

Orginal string:
ЖЖ: Живой Журнал
Transliteration of string using the ISO9 system:
ŽŽ: Živoj Žurnal

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "y"



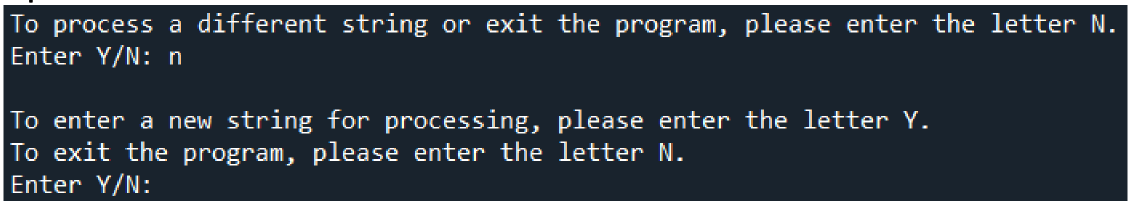
```
Enter Y/N: y

Please enter the relevant code following the colon for one of the transliteration
systems below for converting the text.
International Scholarly System: SCHOLARLY
ISO 9:1995/GOST 7.79-2000(A): ISO9
ICAO MRTD specification: ICAO

Enter code:
```

Input: "icao"



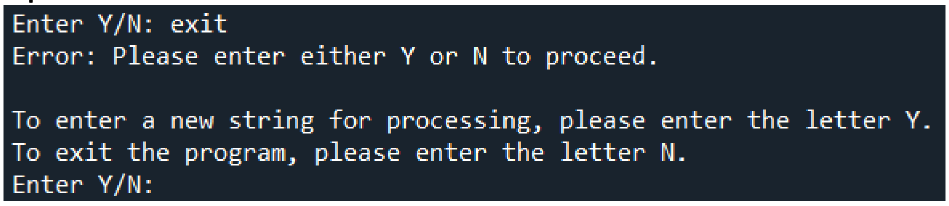
```
Enter code: icao

Orginal string:
ЖЖ: Живой Журнал
Transliteration of string using the ICAO system:
ZhZh: Zhivoi Zhurnal

To process the same string again using a different transliteration system, please
enter the letter Y.
To process a different string or exit the program, please enter the letter N.
Enter Y/N:
```

Input: "n"



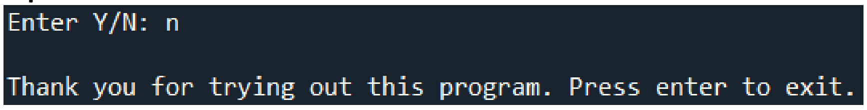
```
To process a different string or exit the program, please enter the letter N.
Enter Y/N: n

To enter a new string for processing, please enter the letter Y.
To exit the program, please enter the letter N.
Enter Y/N:
```

Input: "exit"



```
Enter Y/N: exit
Error: Please enter either Y or N to proceed.

To enter a new string for processing, please enter the letter Y.
To exit the program, please enter the letter N.
Enter Y/N:
```

Input: "n"



```
Enter Y/N: n

Thank you for trying out this program. Press enter to exit.
```

Notes on Results and Potential for Further Modification

Overall, the program worked as expected in all cases, substituting all Russian Cyrillic characters for their Latin equivalents in the specified dictionary as required while leaving any other characters unaffected. For the first test that included all letters of the Russian alphabet, perhaps the most important character to check was "ь" ("мягкий знак", the "soft sign"), since this character is omitted completely without any form of equivalent in many transliteration systems, such as the ICAO system. For the places where this character appears in the pangram ("бьёт пятью… фехтовальщика"), this character was indeed omitted completely using the ICAO system ("bet piatiu… fekhtovalshschika"), rather than being substituted with an alternative such as space, which proves that the "ь" key in this dictionary does indeed return an empty string when it is accessed.

The second test shows that the program can indeed convert bibliographic information alongside translations of this information, although it should be emphasised here that the program only goes as far as "romanising" names and other proper nouns in the sense of returning Latin characters that mirror the characters used in the original Russian. This is evident in the case of the author Tolstoy, whose first name is commonly given as "Leo" in English texts, yet is handled here by all three systems as "Lev", reflecting the characters and pronunciation used in the original Russian ("Лев").

Aside from demonstrating the program's ability to test multiple input strings and to reject invalid procedural input entered by the user, the third test more pointedly demonstrates how the program handles capital letters, particularly those used in abbreviations. When specifying separate dictionary entries for uppercase versions of the characters, any characters that take one-to-multiple mappings in the Latin alphabet have been given with their first letter in uppercase and any following characters in lowercase: this ensures that any entered sentence beginning with a capital letter will always have its transliterated equivalent beginning with only one capital letter too. Generally, this presents no problem for abbreviations or combinations of uppercase letters either: in practice, transliterated uppercase letters that take multiple characters in Latin systems use this same approach (to avoid confusing capital "Ж" ("Zh") with capital "З" ("Z"), for example, given that both words that these letters stand for will start with "Z"), and this can be seen the handling of the Latin name for the Moscow park/metro station "ВДНХ" which is commonly rendered in Latin characters as "VDNKh".

Regarding potential ways that the program could be modified further, it would theoretically be quite simple for a user with sufficient knowledge of Python programming to add in their own custom transliteration system alongside the three existing systems if desired: provided that the character mappings are applied consistently in all cases, they would just need to take a copy of one of the existing dictionaries, change the mappings as necessary, provide an accompanying code for this new dictionary and update the relevant blocks of the program to incorporate the new code and dictionary in line with the procedures used for accessing the other dictionaries.

Another relatively simple modification that could be made would involve changing the initial input method so that the program reads the content of a file specified by the user and returns a transliterated version of the text in this file, as opposed to having the user manually enter a string. In practice, this would make more sense for real-world use cases that involve processing larger amounts of text, such as an entire bibliography of the kind sampled in the second test.

With regard to more advanced modifications, one potential feature I would have liked to have implemented would be a more complex transliteration system that takes into account certain pronunciation-related features and exceptions of the Russian language and returns these exceptions where necessary, in order to create a more "phonetic" transliteration that more closely resembles everyday transliterations of Russian words that are produced for the benefit of non-Russian speakers. For example, the masculine and neuter genitive singular adjectival ending -ого/-его (as featured in the pangram for the first test with "шустрого") is literally translated -ogo/-ego: however, the "г"/"g" in such endings is actually pronounced as a "v", and so the envisioned transliteration program would include additional rules (possibly making use of regular expressions, for example) to identify these endings where they occur and instead romanise them as -ovo/-evo. Similar exception rules could be created for words such as "что" (literally "chto" but pronounced "shto"), or to determine how the letter "е" is transliterated in some systems depending on the character that precedes it – e.g. for road signs, this is converted to "ye" at the beginning of a word, after other vowels and after soft and hard signs, but "e" in all other cases (Wikipedia, 2021b).

An even more complex version of the program would begin to take into account exceptions that occur in the romanisation of names of people and places, such as the issue of "Leo" vs "Lev" highlighted in the commentary on the results of the second test. However, at this stage, it makes more sense to move away from rules-based approaches and consider a different strategy, such as named entity recognition and other approaches that involve machine learning, as highlighted in the literature surrounding the topic. At this point, therefore, it is likely the program would have to be significantly rewritten, rather than undergo a case of simple modification.

Bibliography

International Civil Aviation Organization (2015) *Doc 9303: Machine Readable Travel Documents, Seventh Edition: Part 3: Specifications Common to all MRTDs* [online]. [Accessed 11 February 2021]. Available at: <https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf>.

International Organization for Standardization (1995) *ISO 9:1995: Information and documentation — Transliteration of Cyrillic characters into Latin characters — Slavic and non-Slavic languages* [online]. [Accessed 11 February 2021] Available at: <https://www.iso.org/standard/3589.html>.

The Library of Congress (2012) *ALA-LC Romanization Tables: Transliteration Schemes for Non-Roman Scripts: Russian* [online]. [Accessed 11 February 2021]. Available at: <https://www.loc.gov/catdir/cpso/romanization/russian.pdf>.

van Rossum, G., Warsaw, B., Coghlan, N. (2001) *PEP 8 -- Style Guide for Python Code* [online]. [Accessed 11 February 2021]. Available at: <https://www.python.org/dev/peps/pep-0008/>.

Wikipedia (2021b) *Romanization of Russian: Transliteration Table* [online]. [Accessed 11 February 2021]. Available at: <https://en.wikipedia.org/wiki/Romanization_of_Russian#Transliteration_table>.

Wikipedia (2021a) *Scientific transliteration of Cyrillic* [online]. [Accessed 11 February 2021]. Available at: <https://en.wikipedia.org/wiki/Scientific_transliteration_of_Cyrillic>.