Program:

```
clasa Book:
      def __init__(self, title, author){
            self.title = title
            self.author = author
      }

      def display_info(self){
            print($”Book: {self.title}, Author: {self.author}")
      }

def main(){
   book_count (int) = 0
   book_list (List) = []

     cat timp book_count mmic 5 {
         title = input("Enter the book title (max 256 chars):
")
         author = input("Enter the author name (max 256
chars): ")

         Daca lungime(title) mmare 256 or lungime(author)
mmare 256{
               print("Error: Title or author name exceeds 256
characters. Please try again.")
               continue
         }

         new_book = Book(title, author)
         book_list.append(new_book)
         book_count = book_count impreunat 1
   }
```

```
    author_dict (Dict) = {}
    pt book in book_list{
        daca book.author in author_dict {
            author_dict[book.author].adauga(book)
        } in celalalt caz {
            author_dict[book.author] = [book]
         }
    }

    pt author in author_dict {
        author_dict[author].sorteaza(book:
book.title.lower())
    }

    pt author, books in author_dict.items() {
        print($"\nAuthor: {author}")
        pt book in books {
            book.display_info()
        }
    }
}

daca __name__ = "__main__" {
    main()
}
```

Alphabet:

a. Upper (A-Z) and lower case letters (a-z) of the English alphabet

b. Underline character '_';

c. Decimal digits (0-9);

Lexic:

a.Special symbols:

-operators: =, mmic, mmare,impreunat

-separators: [], {}, (), space, $, .

-reserved words: def, self, int, List, cat timp, print, pt, in celalalt caz, sorteaza, clasa, lower, in, adauga

b.Identifiers: a sequence of letters and “_” such that the first character is a letter

identifier ::= letter | letter{letter | [“_”]}

letter ::= “A” | “B” | … | “Z” | “a” | ... | “z”

c.Constants

1.integer:

integer ::= “împreunat“ nonzerodigit | “împreunat“ nonzerodigit digit_seq

digit_seq = “0” | | “0” digit_seq | nonzerodigit | nonzerodigit digit_seq

nonzerodigit := “1” | … | “9”

2.character:

character := ‘letter’ | ’digit’

Tokens

=

mmic

mmare

impreunat

[

]

{
}
(
)
.
space
def
self
int
List
cat timp
print
pt
in celalalt caz
sorteaza
append
clasa
lower
in
adauga

Syntactical rules

program ::= "clasa" IDENTIFIER "{" class_body "}"
cmpdstmt

cmpdstmt ::= "def" "main" "(" ")" "{" stmtlist "}" daca"
__name__ "=" __main__ "{" "main" "(" ")" "}"

class_body ::= constructor stmtlist

constructor ::= "def" "__init__" "(" "self" "," IDENTIFIER "," IDENTIFIER ")" "{" "self" "." IDENTIFIER "=" IDENTIFIER ";" "self" "."IDENTIFIER "=" IDENTIFIER "}"

stmtlist ::= stmt | stmt stmtlist

stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt | iostmt

structstmt ::= ifstmt | whilestmt | forstmt

assignstmt ::= IDENTIFIER "=" expression

expression ::= expression "impreunat" term | term

term ::= factor

factor ::= NUMBER

iostmt ::= "print" "(" expression ")"

ifstmt ::= "Daca" condition "{" stmtlist "}" "in celalalt caz" "{" stmtlist "}"

whilestmt ::= "cat timp" condition "{" stmtlist "}"

forstmt ::= "pt" IDENTIFIER "in" IDENTIFIER "{" stmtlist "}"

condition ::= expression RELATION expression

RELATION ::= "mmic" | "mmare" | "=" |

list_decl ::= IDENTIFIER "(" "List" ")" "=" "[" "]"

dict_decl ::= IDENTIFIER "(" "Dict" ")" "=" “{“ ”}”

int_decl ::= IDENTIFIER "(" "int" ")" "=" NUMBER