**CS 1120 Computer Science II (with Python),** Fall 2020  
Department of Computer Science, Western Michigan University  
Instructor: Wassnaa Al-Mawee, Lab TA: Utkarsha Prashant Kotkar

Andrew Kroll

**SOFTWARE LIFE CYCLE REPORT FOR LAB ASSIGNMENT 2**

**PHASE 1: SPECIFICATION (“What do we build?)**

Create a Python application to keep track of a library’s collection of materials. Read the collection from an input file. Users should be able to display a list of materials and check out materials using the materials’ call numbers.

**PHASE 2: DESIGN**

**2.1 Modules (Classes) and Their Structure (Class Hierarchy)**

**Main Method:**

Contains the main program loop.

**Controller Class:**

Manages LibraryItem storage, display, and checkout.

Contains methods:

* \_\_init\_\_
  + Initializes empty books and periodicals dictionaries
* show\_menu
  + Displays the application menu text
* display\_collection
  + Displays each book and periodical registered
* find\_item
  + Accepts a call number argument
  + Returns the associated book or periodical if registered
* check\_out\_materials
  + Accepts a call number argument
  + Checks out the specified item if registered and available
  + Alerts the user if the item is not registered or not available
* read\_input
  + Accepts a file name argument
  + Reads library item information from a file

**LibraryItem Class:**

Stores information about call number, check out status, date out, and due date.

Contains methods:

* \_\_init\_\_
  + Accepts a call number argument
  + Initializes call\_num variable with the given call number. Initializes checked\_out with False. date\_checked\_out and due\_date with 0000-00-00 placeholder dates.
* check\_out
  + Marks the item checked out and updates the date\_checked\_out to today.
* get\_call\_number
  + Returns the item’s call number
* is\_checked\_out
  + Returns whether the item is checked out or not
* get\_date\_checked\_out
  + Returns the date the item was checked out
* get\_due\_date
  + Returns the item’s due date
* set\_due\_date
  + Accepts a date string argument
  + Update’s the item’s due\_date value
* \_\_str\_\_
  + Returns information about the item’s call number, check out status, and date out and due date if checked out.

**Book Class: Inherits LibraryItem**

Stores information about the book’s title, author, and genre, in addition to LibraryItem information.

Contains methods:

* \_\_init\_\_
  + Accepts call number, title, author, and genre arguments
  + Calls the LibraryItem’s \_\_init\_\_ method with the call number argument
  + Updates the title, author, and genre values with the provided ones.
* check\_out
  + Calls the LibraryItem’s check\_out method
  + Updates the due date to 21 days from now
* \_\_str\_\_
  + Returns information about the book’s title, author, and genre, as well as information returned by the LibraryItem’s \_\_str\_\_ method.

**Periodical Class: Inherits LibraryItem**

Stores information about the periodical’s title, volume, issue, and subject, in addition to LibraryItem information.

Contains methods:

* \_\_init\_\_
  + Accepts call number, title, volume, issue, and subject arguments
  + Calls the LibraryItem’s \_\_init\_\_ method with the call number argument
  + Updates the title, volume, issue, and subject values with the provided ones.
* check\_out
  + Calls the LibraryItem’s check\_out method
  + Updates the due date to 7 days from now
* \_\_str\_\_
  + Returns information about the periodical’s title, volume, issue, and subject, as well as information returned by LibraryItem’s \_\_str\_\_ method.

**2.2 Pseudocode for the Modules (Classes)**

**File LA2Main.py:**

# Method main:

# create an instance of the Controller class

# tell the controller to read from input.txt

# while True:

# display menu

# get user selection

# if selection is 1:

# display collection

# elif selection is 2:

# check out materials

# elif selection is 3:

# break

# else:

# invalid response

# Controller Class:

# Constructor():

# define empty books dictionary

# define empty periodicals dictionary

#

# Method show\_menu():

# display program menu

#

# Method display\_collection():

# display information about each book

# display information about each periodical

#

# Method find\_item(call\_number):

# if call\_number in books:

# return book

# if call\_number in periodicals:

# return periodical

#

# Method check\_out\_materials():

# ask user which call number to check out

# item = find\_item(call\_number)

# if item wasn't found or is checked out:

# tell user it's not available

# else:

# check out item

# tell user information about the item

#

# Method read\_input(file\_name):

# open file

# for line in file:

# if line starts with p:

# add periodical to periodicals

# elif line starts with b:

# add book to books

# Class LibraryItem:

# Constructor(call\_num):

# update call\_num with provided call number

# set checked\_out to false

# set date\_checked\_out and due\_date to placeholder values

#

# Method check\_out():

# set checked\_out to true

# set date\_checked\_out to today

#

# Method get\_call\_number():

# return call\_num

#

# Method is\_checked\_out():

# return checked\_out

#

# Method get\_date\_checked\_out():

# return date\_checked\_out

#

# Method get\_due\_date():

# return due\_date

#

# Method set\_due\_date(due\_date):

# update due\_date with provided due date

#

# Method \_\_str\_\_():

# return formatted information about the call number, checked out status, and date out and due date if checked out.

# Class Book(LibraryItem):

# Constructor(call\_num, title, author, genre):

# call LibraryItem constructor with call\_num

# update title, author, and genre with provided values

#

# Method check\_out():

# call LibraryItem check\_out

# update due\_date to 21 days from now

#

# Method \_\_str\_\_():

# return information about the book title, author, and genre, along with LibraryItem's \_\_str\_\_ results

# Class Periodical(LibraryItem):

# Constructor(call\_num, title, volume, issue, subject):

# call LibraryItem constructor with call\_num

# update title, volume, issue, and subject with provided values

#

# Method check\_out():

# call LibraryItem check\_out

# update due\_date to 7 days from now

#

# Method \_\_str\_\_():

# return information about the periodical title, volume, issue, and subject, along with LibraryItem's \_\_str\_\_ results.

**PHASE 3: RISK ANALYSIS (“What can go wrong, and how bad can it be?”)**

An improperly formatted input.txt file has the potential to crash the program.

**PHASE 4: VERIFICATION (“Are the algorithms correct?”)**

All input is validated. All “algorithms” are functional.

**PHASE 5: CODING**

**5a) Code Refinement #1 (class structure with pseudocode only; pseudocode is used as comments)**

**File LA2Main.py:**

# Method main:

# create an instance of the Controller class

# tell the controller to read from input.txt

# while True:

# display menu

# get user selection

# if selection is 1:

# display collection

# elif selection is 2:

# check out materials

# elif selection is 3:

# break

# else:

# invalid response

def main():

pass

# Controller Class:

class Controller:

# Constructor():

# define empty books dictionary

# define empty periodicals dictionary

def \_\_init\_\_(self):

pass

#

# Method show\_menu():

# display program menu

def show\_menu(self):

pass

#

# Method display\_collection():

# display information about each book

# display information about each periodical

def display\_collection(self):

pass

#

# Method find\_item(call\_number):

# if call\_number in books:

# return book

# if call\_number in periodicals:

# return periodical

def find\_item(self, call\_num):

pass

#

# Method check\_out\_materials():

# ask user which call number to check out

# item = find\_item(call\_number)

# if item wasn't found or is checked out:

# tell user it's not available

# else:

# check out item

# tell user information about the item

def check\_out\_materials(self):

pass

#

# Method read\_input(file\_name):

# open file

# for line in file:

# if line starts with p:

# add periodical to periodicals

# elif line starts with b:

# add book to books

def read\_input(self, file\_name):

pass

# Class LibraryItem:

class LibraryItem:

# Constructor(call\_num):

# update call\_num with provided call number

# set checked\_out to false

# set date\_checked\_out and due\_date to placeholder values

def \_\_init\_\_(self, call\_num):

pass

#

# Method check\_out():

# set checked\_out to true

# set date\_checked\_out to today

def check\_out(self):

pass

#

# Method get\_call\_number():

# return call\_num

def get\_call\_number(self):

pass

# Method is\_checked\_out():

# return checked\_out

def is\_checked\_out(self):

pass

#

# Method get\_date\_checked\_out():

# return date\_checked\_out

def get\_date\_checked\_out(self):

pass

#

# Method get\_due\_date():

# return due\_date

def get\_due\_date(self):

pass

#

# Method set\_due\_date(due\_date):

# update due\_date with provided due date

def set\_due\_date(self, due\_date):

pass

#

# Method \_\_str\_\_():

# return formatted information about the call number, checked out status, and date out and due date if checked out.

def \_\_str\_\_():

pass

# Class Book(LibraryItem):

class Book(LibraryItem):

# Constructor(call\_num, title, author, genre):

# call LibraryItem constructor with call\_num

# update title, author, and genre with provided values

def \_\_init\_\_(self, call\_num, title, author, genre):

pass

#

# Method check\_out():

# call LibraryItem check\_out

# update due\_date to 21 days from now

def check\_out(self):

pass

#

# Method \_\_str\_\_():

# return information about the book title, author, and genre, along with LibraryItem's \_\_str\_\_ results

def \_\_str\_\_(self):

pass

# Class Periodical(LibraryItem):

class Periodical(LibraryItem):

# Constructor(call\_num, title, volume, issue, subject):

# call LibraryItem constructor with call\_num

# update title, volume, issue, and subject with provided values

def \_\_init\_\_(self, call\_num, title, volume, issue, subject):

pass

#

# Method check\_out():

# call LibraryItem check\_out

# update due\_date to 7 days from now

def check\_out(self):

pass

#

# Method \_\_str\_\_():

# return information about the periodical title, volume, issue, and subject, along with LibraryItem's \_\_str\_\_ results.

def \_\_str\_\_(self):

pass

**5b) Code Refinement #2 (still incomplete program: class and constructor/method structure with pseudocode only; pseudocode is used as comments)**

**File LA2Main.py:**

# Method main:

# create an instance of the Controller class

# tell the controller to read from input.txt

# while True:

# display menu

# get user selection

# if selection is 1:

# display collection

# elif selection is 2:

# check out materials

# elif selection is 3:

# break

# else:

# invalid response

def main():

control = Controller()

control.read\_input("input.txt")

response = ""

quit\_flag = False

while quit\_flag is False:

control.show\_menu()

response = input("Please choose an option: ")

if response == "1":

control.display\_collection()

elif response == "2":

control.check\_out\_materials()

elif response == "3":

quit\_flag = True

else:

print("Invalid response!")

print("Good bye!")

# Controller Class:

class Controller:

# Constructor():

# define empty books dictionary

# define empty periodicals dictionary

def \_\_init\_\_(self):

self.books = {}

self.periodicals = {}

#

# Method show\_menu():

# display program menu

def show\_menu(self):

pass

#

# Method display\_collection():

# display information about each book

# display information about each periodical

def display\_collection(self):

pass

#

# Method find\_item(call\_number):

# if call\_number in books:

# return book

# if call\_number in periodicals:

# return periodical

def find\_item(self, call\_num):

pass

#

# Method check\_out\_materials():

# ask user which call number to check out

# item = find\_item(call\_number)

# if item wasn't found or is checked out:

# tell user it's not available

# else:

# check out item

# tell user information about the item

def check\_out\_materials(self):

pass

#

# Method read\_input(file\_name):

# open file

# for line in file:

# if line starts with p:

# add periodical to periodicals

# elif line starts with b:

# add book to books

def read\_input(self, file\_name):

pass

# Class LibraryItem:

class LibraryItem:

# Constructor(call\_num):

# update call\_num with provided call number

# set checked\_out to false

# set date\_checked\_out and due\_date to placeholder values

def \_\_init\_\_(self, call\_num):

self.call\_num = call\_num

self.checked\_out = False

self.date\_checked\_out = "0000-00-00"

self.due\_date = "0000-00-00"

#

# Method check\_out():

# set checked\_out to true

# set date\_checked\_out to today

def check\_out(self):

pass

#

# Method get\_call\_number():

# return call\_num

def get\_call\_number(self):

pass

# Method is\_checked\_out():

# return checked\_out

def is\_checked\_out(self):

pass

#

# Method get\_date\_checked\_out():

# return date\_checked\_out

def get\_date\_checked\_out(self):

pass

#

# Method get\_due\_date():

# return due\_date

def get\_due\_date(self):

pass

#

# Method set\_due\_date(due\_date):

# update due\_date with provided due date

def set\_due\_date(self, due\_date):

pass

#

# Method \_\_str\_\_():

# return formatted information about the call number, checked out status, and date out and due date if checked out.

def \_\_str\_\_():

pass

# Class Book(LibraryItem):

class Book(LibraryItem):

# Constructor(call\_num, title, author, genre):

# call LibraryItem constructor with call\_num

# update title, author, and genre with provided values

def \_\_init\_\_(self, call\_num, title, author, genre):

super().\_\_init\_\_(call\_num)

self.title = title

self.author = author

self.genre = genre

#

# Method check\_out():

# call LibraryItem check\_out

# update due\_date to 21 days from now

def check\_out(self):

pass

#

# Method \_\_str\_\_():

# return information about the book title, author, and genre, along with LibraryItem's \_\_str\_\_ results

def \_\_str\_\_(self):

pass

# Class Periodical(LibraryItem):

class Periodical(LibraryItem):

# Constructor(call\_num, title, volume, issue, subject):

# call LibraryItem constructor with call\_num

# update title, volume, issue, and subject with provided values

def \_\_init\_\_(self, call\_num, title, volume, issue, subject):

super().\_\_init\_\_(call\_num)

self.title = title

self.volume = volume

self.issue = issue

self.subject = subject

#

# Method check\_out():

# call LibraryItem check\_out

# update due\_date to 7 days from now

def check\_out(self):

pass

#

# Method \_\_str\_\_():

# return information about the periodical title, volume, issue, and subject, along with LibraryItem's \_\_str\_\_ results.

def \_\_str\_\_(self):

pass

**5c) Code Refinement #3 (complete program – with complete fields/properties, code for constructor/methods)**

**LA2Main.py:**

*from* datetime *import* datetime  
*from* datetime *import* timedelta  
  
  
*class* LibraryItem:  
 *def \_\_init\_\_*(*self*, call\_num):  
 *"""  
 Initializes the LibraryItem object. Updates call\_num, checked\_out,  
 date\_checked\_out, and due\_date with provided or default values.* ***:param*** *call\_num: the LibraryItem's call number.  
 """  
 self*.call\_num = call\_num  
 *self*.checked\_out = *False  
 self*.date\_checked\_out = "0000-00-00"  
 *self*.due\_date = "0000-00-00"  
  
 *def* check\_out(*self*):  
 *"""  
 Marks the LibraryItem as checked out and updates the date\_checked\_out  
 to today.  
 """  
 self*.checked\_out = *True  
 self*.date\_checked\_out = datetime.date(datetime.now())  
  
 *def* get\_call\_number(*self*) -> *str*:  
 *"""  
 Returns the LibraryItem's call number.* ***:return****: the LibraryItem's call number.  
 """  
 return self*.call\_num  
  
 *def* is\_checked\_out(*self*) -> *bool*:  
 *"""  
 Returns whether the LibraryItem is checked out or not.* ***:return****: the boolean value of checked\_out.  
 """  
 return self*.checked\_out  
  
 *def* get\_date\_checked\_out(*self*) -> *str*:  
 *"""  
 Returns when the LibraryItem was checked out.* ***:return****: the date\_checked\_out string.  
 """  
 return self*.date\_checked\_out  
  
 *def* get\_due\_date(*self*) -> *str*:  
 *"""  
 Returns the LibraryItem's due date.* ***:return****: the due\_date string.  
 """  
 return self*.due\_date  
  
 *def* set\_due\_date(*self*, due\_date: *str*):  
 *"""  
 Updates the LibraryItem's due date.* ***:param*** *due\_date: the new due date string.  
 """  
 self*.due\_date = due\_date  
  
 *def \_\_str\_\_*(*self*) -> *str*:  
 *"""  
 Returns information about the LibraryItem's call number and checkout  
 status.* ***:return****: a formatted string of the call number, checked out status,  
 date out, and due date (if checked out).  
 """  
 if self*.checked\_out:  
 *return* "Call Number: {}\n" \  
 "Checked Out: YES\n" \  
 "Date Out: {}\n" \  
 "Date Due: {}".format(*self*.call\_num, *self*.date\_checked\_out,  
 *self*.due\_date)  
 *return* "Call Number: {}\n" \  
 "Checked Out: NO".format(*self*.call\_num)  
  
  
*class* Book(LibraryItem):  
 *def \_\_init\_\_*(*self*, call\_num: *str*, title: *str*, author: *str*, genre: *str*):  
 *"""  
 Initializes the Book object with provided call number, title, author,  
 and genre values.* ***:param*** *call\_num: the book's call number.* ***:param*** *title: the book's title.* ***:param*** *author: the book's author.* ***:param*** *genre: the book's genre.  
 """  
 super*().*\_\_init\_\_*(call\_num)  
 *self*.title = title  
 *self*.author = author  
 *self*.genre = genre  
  
 *def* check\_out(*self*):  
 *"""  
 Marks the book as checked out and updates the due date to 21 days  
 from now.  
 """  
 super*().check\_out()  
 *self*.set\_due\_date(*str*(datetime.date(datetime.now()  
 + timedelta(days=21))))  
  
 *def \_\_str\_\_*(*self*) -> *str*:  
 *"""  
 Returns information about the book's title, author, genre, and  
 checked out status.* ***:return****: a formatted string of the title, author, genre, call number,  
 checked out status, and the date out and due date if checked out.  
 """  
 return* "\n" \  
 "Book Title: {}\n" \  
 "Author: {}\n" \  
 "Genre: {}\n" \  
 "{}".format(*self*.title, *self*.author, *self*.genre,  
 *super*().\_\_str\_\_())  
  
  
*class* Periodical(LibraryItem):  
 *def \_\_init\_\_*(*self*, call\_num: *str*, title: *str*, volume: *int*, issue: *int*,  
 subject: *str*):  
 *"""  
 Initializes the Periodical object with the provided call number,  
 title, volume number, issue number, and subject.* ***:param*** *call\_num: the periodical's call number.* ***:param*** *title: the periodical's title.* ***:param*** *volume: the periodical's volume number.* ***:param*** *issue: the periodical's issue number.* ***:param*** *subject: the periodical's subject.  
 """  
 super*().*\_\_init\_\_*(call\_num)  
 *self*.title = title  
 *self*.volume = volume  
 *self*.issue = issue  
 *self*.subject = subject  
  
 *def* check\_out(*self*):  
 *"""  
 Marks the book as checked out and updates the due date to 7 days  
 from now.  
 """  
 super*().check\_out()  
 *self*.set\_due\_date(*str*(datetime.date(datetime.now()  
 + timedelta(days=7))))  
  
 *def \_\_str\_\_*(*self*) -> *str*:  
 *"""  
 Returns information about the periodical's title, volume and issue  
 numbers, subject, and checked out status.* ***:return****: a formatted string of title, volume and issue numbers,  
 subject, checked out status, and the date out and due date if  
 checked out.  
 """  
 return* "\n" \  
 "Periodical Title: {}\n" \  
 "Volume: {}\n" \  
 "Issue: {}\n" \  
 "Subject: {}\n" \  
 "{}".format(*self*.title, *self*.volume, *self*.issue, *self*.subject,  
 *super*().\_\_str\_\_())  
  
  
*class* Controller:  
 *def \_\_init\_\_*(*self*):  
 *"""  
 Initializes the Controller object with empty books and periodicals  
 dictionaries.  
 """  
 self*.books = {}  
 *self*.periodicals = {}  
  
 *def* show\_menu(*self*):  
 *"""  
 Displays the Library Catalog System menu.  
 """  
 print*()  
 *print*("--------- Menu ---------")  
 *print*(" 1) Display Collection")  
 *print*(" 2) Check out materials")  
 *print*(" 3) Quit")  
 *print*("------------------------")  
  
 *def* display\_collection(*self*):  
 *"""  
 Displays each registered book and periodical.  
 """  
 for* book *in self*.books: *# Display each book  
 print*(*self*.books[book])  
 *for* periodical *in self*.periodicals: *# Display each periodical  
 print*(*self*.periodicals[periodical])  
  
 *def* find\_item(*self*, call\_num: *str*) -> LibraryItem:  
 *"""  
 Returns the requested book or periodical if registered.* ***:param*** *call\_num: the book or periodical's call number.* ***:return****: the book or periodical object if registered, otherwise None.  
 """* call\_num = call\_num.upper()  
 *if* call\_num *in self*.books: *# Check in books  
 return self*.books[call\_num]  
 *if* call\_num *in self*.periodicals: *# Check in periodicals  
 return self*.periodicals[call\_num]  
  
 *def* check\_out\_materials(*self*):  
 *"""  
 Asks the user which item they want to check out. Alerts user if  
 item is unavailable. Checks out item if available.  
 """* call\_num = *input*("\nEnter the call number: ").upper()  
 item = *self*.find\_item(call\_num)  
 *if* item *is None or* item.is\_checked\_out(): *# Not registered or already  
 # checked out  
 print*("Item is not available.")  
 *else*: *# Available* item.check\_out()  
 *print*(*str*(item))  
  
 *def* read\_input(*self*, file\_name: *str*):  
 *"""  
 Reads all library item information from the targeted file.* ***:param*** *file\_name: the name of the library item information file.  
 """  
 with open*(file\_name, 'r') *as* file: *# Opens the specified file to read  
 for* line *in* file:  
 line = line.rstrip("\n") *# Removes trailing newline* lines = line.split(",") *# Splits line to be passed  
 if* line.startswith("P"): *# Periodical  
 self*.periodicals[lines[1].upper()] = \  
 Periodical(lines[1], lines[2], *int*(lines[3]),  
 *int*(lines[4]), lines[5])  
 *elif* line.startswith("B"): *# Book  
 self*.books[lines[1].upper()] = \  
 Book(lines[1], lines[2], lines[3], lines[4])  
  
  
*def* main():  
 *"""  
 The main method. Controls the main application loop.  
 """* control = Controller() *# Creates a Controller object* control.read\_input("input.txt") *# Reads the input.txt file* response = ""  
 quit\_flag = *False # Tells the loop whether to continue or to stop  
 while* quit\_flag *is False*: *# Infinitely loops until told to stop* control.show\_menu() *# Displays the program menu* response = *input*("Please choose an option: ")  
 *if* response == "1": *# Display Collection* control.display\_collection()  
 *elif* response == "2": *# Checkout Item* control.check\_out\_materials()  
 *elif* response == "3": *# Quit* quit\_flag = *True  
 else*: *# Invalid  
 print*("Invalid Response!")  
 *print*("Goodbye!")  
  
  
main()

**PHASE 6: TESTING (“Did we build it correctly?”)**

Yes. See test output below…

--------- Menu ---------

1) Display Collection

2) Check out materials

3) Quit

------------------------

Please choose an option: 1

Book Title: The Cat in the Hat

Author: Dr. Seuss

Genre: Children's Literature

Call Number: C124.S17

Checked Out: NO

Book Title: A Game of Thrones

Author: George R. R. Martin

Genre: Fantasy Literature

Call Number: F380.M1

Checked Out: NO

Periodical Title: Computational Linguistics

Volume: 37

Issue: 4

Subject: Computational Linguistics

Call Number: QJ072.C23.37.4

Checked Out: NO

Periodical Title: Communications of the ACM

Volume: 55

Issue: 2

Subject: Computer Science

Call Number: QJ015.C42.55.2

Checked Out: NO

--------- Menu ---------

1) Display Collection

2) Check out materials

3) Quit

------------------------

Please choose an option: 2

Enter the call number: F380.M1

Book Title: A Game of Thrones

Author: George R. R. Martin

Genre: Fantasy Literature

Call Number: F380.M1

Checked Out: YES

Date Out: 2020-10-06

Date Due: 2020-10-27

--------- Menu ---------

1) Display Collection

2) Check out materials

3) Quit

------------------------

Please choose an option: 2

Enter the call number: F380.M1

Item is not available.

--------- Menu ---------

1) Display Collection

2) Check out materials

3) Quit

------------------------

Please choose an option: 2

Enter the call number: QJ072.C23.37.4

Periodical Title: Computational Linguistics

Volume: 37

Issue: 4

Subject: Computational Linguistics

Call Number: QJ072.C23.37.4

Checked Out: YES

Date Out: 2020-10-06

Date Due: 2020-10-13

--------- Menu ---------

1) Display Collection

2) Check out materials

3) Quit

------------------------

Please choose an option: 2

Enter the call number: invalid

Item is not available.

--------- Menu ---------

1) Display Collection

2) Check out materials

3) Quit

------------------------

Please choose an option: 1

Book Title: The Cat in the Hat

Author: Dr. Seuss

Genre: Children's Literature

Call Number: C124.S17

Checked Out: NO

Book Title: A Game of Thrones

Author: George R. R. Martin

Genre: Fantasy Literature

Call Number: F380.M1

Checked Out: YES

Date Out: 2020-10-06

Date Due: 2020-10-27

Periodical Title: Computational Linguistics

Volume: 37

Issue: 4

Subject: Computational Linguistics

Call Number: QJ072.C23.37.4

Checked Out: YES

Date Out: 2020-10-06

Date Due: 2020-10-13

Periodical Title: Communications of the ACM

Volume: 55

Issue: 2

Subject: Computer Science

Call Number: QJ015.C42.55.2

Checked Out: NO

--------- Menu ---------

1) Display Collection

2) Check out materials

3) Quit

------------------------

Please choose an option: 3

Goodbye!

Process finished with exit code 0

**PHASE 7: REFINING THE PROGRAM (“Add bells and whistles to the program”)**

Nothing to add.

**PHASE 8: PRODUCTION**

Uploaded, along with this document, to the dropbox.

**PHASE 9: MAINTENANCE**

If any maintenance is required, I may do so upon receiving feedback.