PRACTICAL NO. :1

WriteaPythonprogramtomanagetheborrowingrecordsofbooksinalibrary. Implement thefollowingfunctionalities: •Computetheaveragenumberofbooksborrowedbyalllibrarymembers. •Findthebookwiththehighestandlowestnumberofborrowingsinthelibrary. •Countthenumberofmemberswhohavenotborrowedanybooks(denotedbya borrowcountof0). •Displaythemostfrequentlyborrowedbook(i.e.,themodeofborrowcounts). Afterperforming,determinethetimeandSpacecomplexityofeachoperation

```
from collections import Counter

class Library:
    def __init__(self):
        self.member_borrow_counts = {}
        self.book_borrow_counts = {}

    def add_member_borrow(self, member_id, borrow_count):
        self.member_borrow_counts[member_id] = borrow_count

    def add_book_borrow(self, book_title, borrow_count):
        self.book_borrow_counts[book_title] = borrow_count

    def average_books_borrowed(self):
        total = sum(self.member_borrow_counts.values())
        count = len(self.member_borrow_counts)
        if count == 0:
            return 0
        return total / count

    def book_with_highest_and_lowest_borrowings(self):
        if not self.book_borrow_counts:
            return None, None
        max_book = max(self.book_borrow_counts.items(), key=lambda x: x[1])
        min_book = min(self.book_borrow_counts.items(), key=lambda x: x[1])
```

```
        return max_book, min_book

    def count_members_with_zero_borrows(self):
        return sum(1 for count in self.member_borrow_counts.values() if count == 0)

    def most_frequently_borrowed_book(self):
        if not self.book_borrow_counts:
            return None
        borrow_freq = Counter(self.book_borrow_counts.values())
        max_freq = max(borrow_freq.values())
        modes = [book for book, count in self.book_borrow_counts.items()
                 if borrow_freq[count] == max_freq]
        return modes

lib = Library()
lib.add_member_borrow("M001", 3)
lib.add_member_borrow("M002", 0)
lib.add_member_borrow("M003", 5)
lib.add_member_borrow("M004", 0)

lib.add_book_borrow("Book A", 10)
lib.add_book_borrow("Book B", 5)
lib.add_book_borrow("Book C", 10)
lib.add_book_borrow("Book D", 2)

print("Average books borrowed per member:", lib.average_books_borrowed())

max_book, min_book = lib.book_with_highest_and_lowest_borrowings()
print("Book with highest borrowings:", max_book)
print("Book with lowest borrowings:", min_book)
```

```
print("Number of members with zero borrows:", lib.count_members_with_zero_borrows())
print("Most frequently borrowed books (by count):", lib.most_frequently_borrowed_book())
```

OUTPUT:

```
Average books borrowed per member: 2.0
Book with highest borrowings: ('Book A', 10)
Book with lowest borrowings: ('Book D', 2)
Number of members with zero borrows: 2
Most frequently borrowed books (by count): ['Book A', 'Book C']
```