CSCI 3901: Project

Black-box test cases:

Input Validation:

addPublication:

- Publication identifier is provided
- Required keys are provided in the publicationInformation Map
- Author field contains only full names separated by commas
- Year field contains a valid year value
- Publication has been added successfully
- Same publication cannot be added twice

addReferences:

- Publication identifier is provided
- Set of references is provided
- References have been added successfully
- Same reference cannot be added twice

addVenue:

- Venue name is provided
- Required keys are provided in the venueInformation Map
- Research areas are provided
- Venue has been added successfully
- Same venue cannot be added twice

addPublisher:

- Publisher identifier is provided
- Required keys are provided in the publisherInformation Map
- Publisher has been added successfully
- Same publisher cannot be added twice

addArea:

- Research area is provided
- Parent areas are provided
- Research area has been added successfully
- Same research area cannot be added twice

getPublications:

- Publication key is provided
- Publication information is returned successfully

authorCitations:

- Author name is provided
- Number of citations is reported successfully

seminalPapers:

- Research area is provided
- Paper citation count and other citation counts are provided
- Seminal papers are reported successfully

Collaborators:

- Valid author name is provided
- Valid distance value is provided
- Distance value is not negative
- Author exists in the library
- Collaborators are reported successfully for a given author and distance value.

authorResearchAreas:

- Author name is providedThreshold is provided
- Research areas are reported successfully

main:

- Input file name is empty
- Input file name does not exist
- Input file format is not a text file
- Input file contains invalid citation format (e.g. missing closing bracket)
- Input file contains duplicate citations
- Output file name is empty
- Output file format is not a text file
- Output file is not writable

Boundary cases:

addPublication:

- Empty publication identifier is provided
- Empty publicationInformation Map is provided
- Minimum and maximum year values are provided
- Minimum and maximum length values for fields are provided (e.g. title, journal name, etc.)

addReferences:

- Empty publication identifier is provided
- Empty set of references is provided
- Same reference is added twice to the same publication identifier

addVenue:

- Empty venue name is provided
- Empty venueInformation Map is provided
- Maximum number of research areas (e.g. 100) are provided
- Same venue is added twice

addPublisher:

- Empty publisher identifier is provided
- Empty publisherInformation Map is provided
- Same publisher is added twice

addArea:

- Empty research area is provided
- Maximum number of parent areas (e.g. 100) are provided
- Same research area is added twice

getPublications:

- Empty publication key is provided
- Invalid publication key is provided

authorCitations:

- Empty author name is provided
- Author name not in the library is provided

seminalPapers:

- Empty research area is provided
- Paper citation count and other citation count are at the minimum and maximum values

collaborators:

- Empty author name is provided
- Distance value is at the minimum and maximum values
- Author has no collaborators within the given distance

authorResearchAreas:

- Empty author name is provided
- Threshold value is at the minimum and maximum values
- Author has no research areas that meet the threshold criteria

main:

- Citations are in random order (not in the order of appearance in the text)
- Citations reference publications not present in the library.

Possible data structures:

boolean addPublication(String identifier, Map<String, String> publicationInformation):

- HashMap<String, String> to store publication information using Map keys as attribute names and Map values as actual information.

boolean addReferences(String identifier, Set<String> references):

- Set<String> to store the set of references for a publication.

boolean addVenue(String venueName, Map<String, String> venueInformation, Set<String> researchAreas):

- HashMap<String, String> to store venue information using Map keys as attribute names and Map values as actual information.
- Set<String> to store research areas associated with a venue.

boolean addPublisher(String identifier, Map<String, String> publisherInformation):

- HashMap<String, String> to store publisher information using Map keys as attribute names and Map values as actual information.

boolean addArea(String researchArea, Set<String> parentArea):

- HashMap<String, Set<String>> to store research areas and the corresponding parent areas.

Map<String, String> getPublications(String key):

- HashMap<String, String> to store all information about a publication using Map keys as attribute names and Map values as actual information.
- The key is the publication identifier.

int authorCitations(String author):

- HashMap<String, Set<String>> to store the set of publications for each author.
- The key is the author's name.

Set<String> seminalPapers(String area, int paperCitation, int otherCitations):

- HashMap<String, Set<String>> to store the set of publications for each research area.
- The key is the research area.
- HashSet<String> to store the set of seminal papers for a given research area.

Set<String> collaborators(String author, int distance):

- HashMap<String, Set<String>> to store the set of co-authors for each author.
- The key is the author's name.

Set<String> authorResearchAreas(String author, int threshold):

- HashMap<String, Set<String>> to store the set of publications for each research area.
- The key is the research area.
- HashMap<String, Set<String>> to store the set of authors for each research area.
- The key is the research area.
- HashMap<String, Integer> to store the number of papers published by each author.
- The key is the author's name.

main():

- HashMap<String, String> to store the citation string and the corresponding reference number. The key is the citation string and the value is the reference number.
- An ArrayList<String> to store the order of citation strings encountered in the input file. This is used to generate the correct order of references in the output file.

Code design:

- Define a Publication class that holds information about a publication, such as its title, authors, journal or conference name, publication date, and references.

- Define a PublicationVenue class that holds information about a journal or conference venue, such as its name, publisher, editor, editor contact information, location, and conference year.

- Define a Publisher class that holds information about a publisher, such as its contact name, email, and location

- Define a ResearchArea class that holds information about a research area, such as its name and parent research areas.

- Define a PublicationLibrary class that manages the information about publications, venues, publishers, and research areas. The class should have methods for adding and retrieving information about these entities, as well as methods for answering the queries specified in the problem statement.

- Define a PaperConverter class that converts citation strings in a research paper into IEEE-formatted references. The class should have a main method that takes an input file name and an output file name, reads the citation strings from the input file, converts them to references, and writes the resulting paper with the references to the output file.

- I might need other classes based on the requirements like PublicationType, Organization, and Citation etc.

In this design user can insert the information about the area first and after that add publisher, venue, publication and references at the end. In in that way insertion of data can be done in the database.

While processing, user will enter the document name and output file name and based on that program will create the final document after processing data manipulations on the database and convert it in IEEE format.

Key algorithms:

- Sorting algorithms like quicksort or mergesort to sort publications based on their publication date or the number of citations they have received.

- String matching algorithms like regular expressions or string indexing to convert citation strings into actual IEEE citations.
- Database querying algorithms like SQL or NoSQL queries to retrieve information from the database based on different search criteria.
- Algorithmic optimization techniques like memoization or dynamic programming to optimize the performance of the system and reduce the response time for queries.
- Text processing algorithms to identify the seminal papers in a given research area based on their citation patterns.