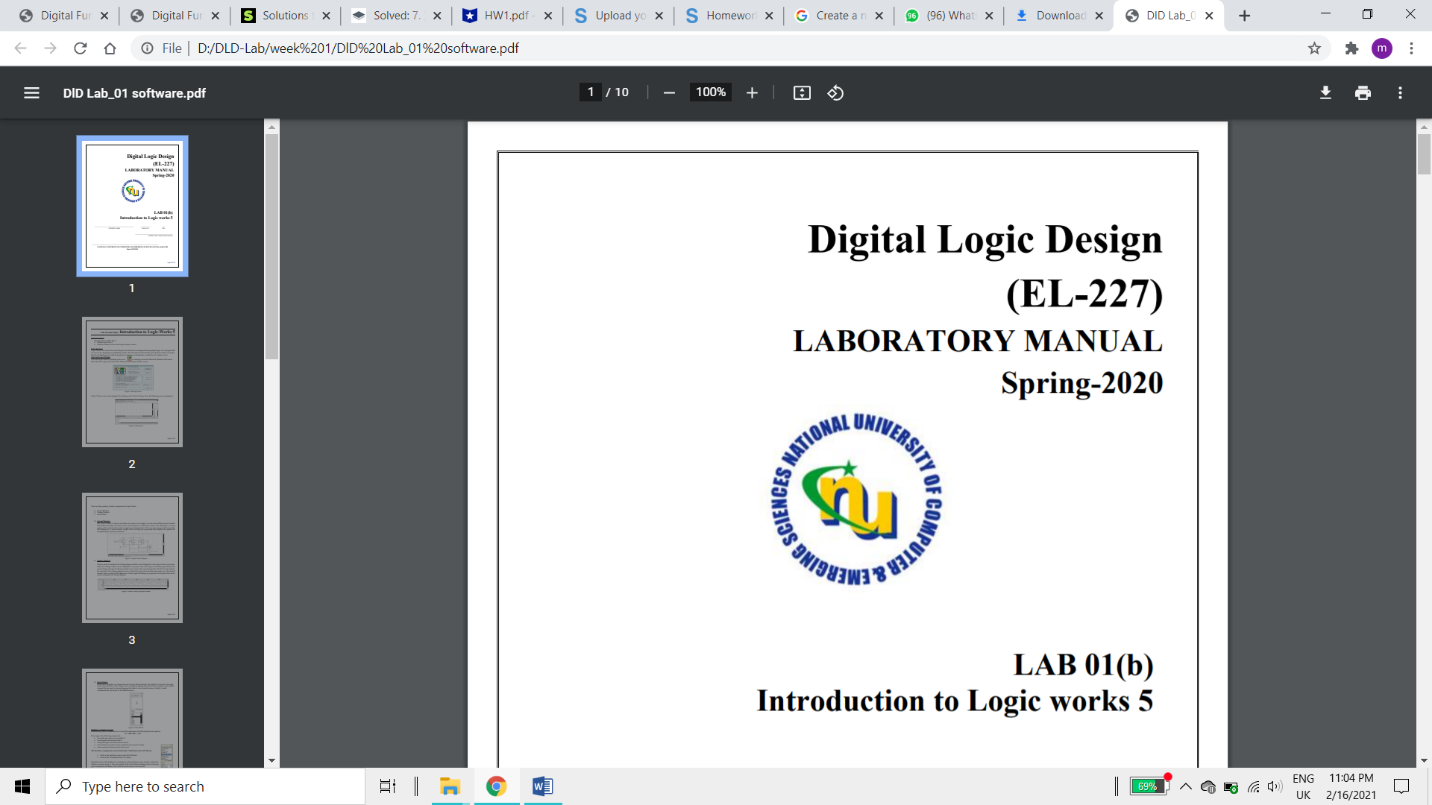
CS-2001 Data Structures

PROJECT

Fall-2021

**Submitted to:** Miss Mubashra Fayyaz

**Group Members:**

* Mohammad Bilal Aziz 20K-0397
* Sabah Mawani 20K-0393
* Maryam Siddiqui 20K-0434

**Section:** BCS-3D

**Project Name:** Stack Sample

**Date of submission:** January 18, 2021

Table of Contents

[Introduction: 3](#_Toc93256107)

[Data set: 3](#_Toc93256108)

[Problem Description: 4](#_Toc93256109)

[Solution Techniques: 5](#_Toc93256110)

[Data processing: 5](#_Toc93256111)

[Data Storage: 6](#_Toc93256112)

[Tags: 6](#_Toc93256113)

[Questions: 6](#_Toc93256114)

[Answers: 6](#_Toc93256115)

[Data Operations: 7](#_Toc93256116)

[Tools and techniques: 8](#_Toc93256117)

[Addition features which may be added: 8](#_Toc93256118)

Introduction:

This project is aimed to help the students in getting their question answered quickly and lessen their search time. We have selected 10% of the stack overflow’s questions and answers.

# Data set:

Dataset with the text of 10% of questions and answers from the Stack Overflow Programming Q&A website.

This dataset was taken from Kaggle.com.

Data is organized as three different CSV files:

• Questions.csv contains the question ID, title, body, creation date, closed date, score, and owner ID for all non-deleted Stack Overflow questions whose Id is a multiple of 10.

• Answers.csv contains the body, creation date, score, and owner ID for each of the answers to these questions. The Parent ID column links back to the Questions table.

• Tags.csv contains the tags on each of these questions and the id of each question in which the tag appears.

The total size of the unprocessed data of all the files combined is 3.98 GB.

The following is the link for the dataset:

https://www.kaggle.com/stackoverflow/stacksample

Problem Description:

The data set contains a large amount of unprocessed data. We need to solve three major problems in this project:

1. Data Processing

* Reading the files
* Filtering the data
* Cleaning the data
* Storing data into new files

2. Data storage

* Storing the data in different data structures.
* Linking the different data structures to each other.

3. Data Operations

* Data Clustering
* Searching
* Sorting

The user will be prompted to enter a question in the search bar. We need to present the user with the most relevant questions available in the data set along with all the answers associated with each question.

We need to create a menu-driven program allowing the user to select any title they want and display the question and answers that come under that title.

Solution Techniques:

## Data processing:

The data was first read from the Questions.csv file, which was then processed to drop the unnecessary columns, only keeping the question ID, creation date, score, title, and body. The records were also dropped if the score was less than 20 or greater than 500. If the record read was valid then it was sanitized so that it could be stored in a new file.

The new file that contained the relevant records from the Questions.csv file is first read and the question IDs are stored into an AVL tree, the Answers.csv file was then processed to only store the records, in a separate file, that had a Parent ID that could be found within the AVL tree. Furthermore, the processed data was sanitized and the irrelevant columns were dropped so that the new file only consisted of the Parent ID, score, and body.

The next part was reading the Tags.csv file and only storing the records, whose ID could be found within the AVL Tree, in a separate file.

Data Storage:

The data is stored in 3 user-defined data structures. The data structures used are Arrays, Linked List, Binary Search Tree, and Hash Maps. Since we required different variations of data structures so only List STL was used. Every other data structure along with a list variation was defined by the user.

The data was stored as follows:

Tags:First, the processed Tags file was read and all the tags were stored in a user-defined Binary Search Tree. The question ID linked with the tag was stored in the node class as a linked list using STL List.

Questions:The questions were read from the processed Questions file and all the details of the questions (id, creation date, score, title, body) were stored in a user-defined hash table. The hash table size was computed by: a prime number greater than 1.3 times the total table entries of the question table. The division method was used for hashing and collision was handled by separate chaining. Hash Tables were used to optimize the searching of questions.

Answers:The answers were read from the processed Answers file and a user-defined linked list was used to store the details of the answers for each Parent ID.

The linked list of each Parent ID contained all the answers, body, and score. This parent ID is used to find the Table Entry of the hash table which contains the same question ID. If the table entry was found, then the list containing the answers was stored in the table entry.

Hence, each table entry now contains the details of the questions and all the answers which have the parent ID same as the question ID.

## **Data Operations:**

**Procedure:**The words were extracted from the question entered by the user and each word was **searched** in the **Binary Search Tree** containing all the tags. If the word was found, then the list of IDs was passed in another Binary Search Tree. This was done to ensure that there are no multiple IDs and each question was printed only once.

To **cluster** the data, the questions containing similar tags as entered by the user were extracted from the data set. The binary tree containing IDs of the tags was used. By traversing the binary search tree, we extracted each ID, and using **searching** on a hash table, we got the table entry for the corresponding ID.

After **clustering** the data and getting all the questions with similar tags, we stored all the table entries of the data in an array of type Hash Table Entry. This array contains all the nodes of the hash table of the clustered data. The size of the array was determined by counting the number of nodes of the BST containing the IDs of the questions with matching tags. Then this array was passed in a **sorting** function. **Merge sort** was used here as the data set was quite large and merge sort is highly suitable for large data sets.

The **clustered** data was **sorted** in descending order of the accumulated score. The question with the highest score was printed at the top. The console screen displays the number of records found. If more than 15 questions were found, only the top 15 questions with the highest score are selected and displayed to the user. Users can select the title of the question and then the question with all the answers is displayed to the user.

Tools and techniques:

The language used to develop this program is C++.   
The DEV C++ IDE is used to write the program.

Techniques used were:

* Filing
* Searching
* Sorting
* Clustering
* Console Designing

# Addition features which may be added:

We may add GUI to make the project more user-friendly and optimize the searching and sorting techniques even more to reduce the time taken to load the data and store it.

--------------------------------------------- THE END ---------------------------------------------------