Course: CSCI207 fall 23

Important Rules:

- **This project will be a teamwork project, with a team consisting of 4 or 5 members FROM THE SAME LAB.**
- **Take attention to handle all exception cases by the user throughout the project.**
- **All team members MUST understand everything in the code.**
- **Code quality is measured according to naming conventions, writing comments, following the dynamic and static nature of program logic.**
- **You'll submit a file with members ids and with .cpp extension. o [e.g., 20000000_20000000_20000000_20000000_20000000.cpp].**
- **Grades are based on the discussion**
- **Submission date : 29 December, 2023. No late submission will be accepted.**

Project Statement

Project: Integrated Inventory Management System
Description:
Design and implement an Integrated Inventory Management System for a factory that includes spare parts and their associated suppliers. The system should leverage various data structures, including linked lists, stacks, queues, and trees, to efficiently manage and organize inventory information.

a) Classes Implementation:

1. **SparePart Class:**
   - Data Members: partName, partNumber, cost, existingParts.
   - Member Functions: getdata() to input spare part details, putdata() to display spare part data, modifydata() to modify spare part information.
2. **Supplier Class:**
   - Data Members: supplierName, supplierCode, address, telephone, email.
   - Member Functions: getdata() to input supplier details, putdata() to display supplier data, modifydata() to modify supplier information.

b) Data Structure for Spare Parts and Suppliers:

Develop a structure for each spare part that includes the group of suppliers providing it. Similarly, create a structure for each supplier to express the spare parts they supply. Utilize **linked lists** to establish **connections** between spare parts and suppliers.

c) Search Function:

Develop a function to **search** for a **specific spare part** or **supplier**. Implement a **search algorithm** that efficiently traverses the data structures, utilizing **binary trees** for **quick lookups and delete**.

Implementation Considerations:

- Ensure that part numbers and supplier codes are **unique** to prevent repetition (**No Duplicates**).
- Utilize **linked lists** to dynamically **manage** lists of spare parts and suppliers.
- Implement **stacks** for managing user interactions during data input. For example, an interaction can, be : “"User entered data for Spare Part number 5”. The interactions stack can then be used to **print** the interactions history where the last performed interaction is shown first.
- Utilize **queues** for efficiently processing and updating spare part and supplier data. For example, when a new spare part is added followed by a modification these should be **enqueued** in a spare_part_processing queue. Processing example “ Spare Part 1 data processing in progress.” “Spare Part 1 modifying in progress” and similarly for the suppliers.
- Employ **trees** for organizing, **Delete** and **searching** spare parts and suppliers based on unique identifiers. You should create **two trees**, spareparts tree and supplier’s tree.
- Mention the time **complexity** for each method you implement just the order of it (e.g. O(N))

d) Main Program:

Write a main() program to **test** the **classes** and the **functions**. The program **for example** should:

- Represent the given data by creating instances of spare parts and suppliers.
- Invite the user to input data for up to 100 spare parts and up to 20 suppliers.
- Print out the data for both spare parts and suppliers.
- let the user select the Structure and the operation that he wants to call

Bonus:

Implementing a graphical user interface (**GUI**) for this project is **optional/bonus**.