DSA ASSIGNMENT

GOLDBERG MACHINE

BY:

R. GOWTHAM (RA1911003010234)

BALA VAMSI(RA1911003010214)

SRIRAM REDDY(RA1911003010218)

M.HARIHARA(RA1911003010245)

**Program Description**

This program has been created in C++ language to use the features of encapsulation and data hiding.

The following Abstract Data Types (ADTs or data structures) are used in the program:

1. **Queue**: Queue is a linear data structure that follows first in first out (FIFO) principle. Queue has been used in the program to fulfil the following functionalities:
   * **enqueue**: To store the entries in the order entered by the user
   * **dequeue:** To de-queue and print the data items in the same order in which they are entered
   * Re-queue the items to reverse the order of data items
2. **Stack**: Stack is a linear data structure that follows last in first out (LIFO) principle. The following functionalities have been built in the stack:
   * **push:** To store the elements in the stack
   * **pop:** To remove the elements in the stack from the top of the stack downwards.
3. **Tree:** Tree is a non-linear data structure. It has a root note at top succeeded by two child nodes, which each child becoming a parent to two more children recursively. As per the requirement, an unordered binary tree has been used in this program, in which each node has two child nodes and the nodes are placed in the order they are entered. The tree is created as a balanced tree so that all the left and right nodes are balanced. The following functionality has been built into this tree data structure:
   * **Pre-order traversal:** First the root node is traversed followed by the left child and all its children recursively in the left to right order, followed by be the right child and its children in the left to right order.
   * **Post order traversal:** First the left child of the root node is traversed recursively in the left to right order, followed by the right child of the root node and its children recursively in the left to right order and finally the root node.
   * **In order traversal and insertion into linked list:** The elements are traversed in the order left child, parent and right child in the recursive order and added to the linked list.
4. **Linked list:** Linked list is a linear data structure. The following functionality has been built into this tree data structure:
   * **Quick sort:** Elements present in the list are sorted in the order based on the first name using a quick sort mechanism
   * **Insert:** A new element is inserted in the ordered list based on the value of its first name

**Program Structure**

The folder structure of the program is described below:

* Root Folder of the project
* **Goldberg.cpp:** The main program for the project where the **“main()”** function has been created and the flow of program execution as per desired functionality is available in this file
* **InputFile:** Test file that contains the input to the program. The file contains records for each data item separated by new line. The fields within a data item are separated by comma
* **Queue** folder:
  + **Queue.hpp:** The file is a header file that contains the class declaration for the Queue class
  + **Queue.cpp:** The file contains the definitions of the member functions of the Queue class
* **Stack** folder:
  + **Stack.hpp:** The file is a header file that contains the class declaration for the Stack class
  + **Stack.cpp:** The file contains the definitions of the member functions of the Stack class
* **Tree** folder:
  + **Tree.hpp:** The file is a header file that contains the class declaration for the Tree class
  + **Tree.cpp:** The file contains the definitions of the member functions of the Tree class
* **LinkedList** folder:
  + **LinkedList.hpp:** The file is a header file that contains the class declaration for the LinkedList class
  + **LinkedList.cpp:** The file contains the definitions of the member functions of the LinkedList class
* **include** folder:
  + **Node.hpp:** The file contains the definition of the **Node** structure that is used to hold the data items for the person data

**How to Run the program**

The program has been compiled on Windows platform. The output of the project is the executable **Goldberg\_Proj.exe.** The program can be executed by running this file (by double-clicking it (or) using the command Goldberg\_Proj.exe from the DOS command line)

**Analysis of the program**

The program has been created keeping the performance i.e. speed of execution and memory usage in view.

The memory has been carefully used by ensuring that the data ownership is properly maintained between the ADTs and the main program.

* A copy of data is created and inserted into the ADTs so that data ownership is maintained with ADTs
* The original data is freed after data is handed over to the ADTs to ensure that the memory is optimally used.