ASSIGNMENT 3
Rube Goldberg Machine

<u>Team Members:</u>

N Aditya Sai	: RA1911030010075
Souvik Dey	: RA1911030010076
Souharda Biswas	: RA1911030010082
Dipali Singh	: RA1911030010138

<u>ADTs Used:</u>

1. Queue

Queues are the First in First out (FIFO) type linear data type that can be used to add the data at one end, called the “rear,” and removed from the other end, called the “front.” We implemented Queue ADT using Linked List. We took the record off the text file and inserted them into the queue.

2. Binary Trees

Binary trees are hierarchical, non linear data structures and thus are used to store data that naturally forms a hierarchy. We used a binary tree to store the elements in an unordered approach.

The modes of traversal in binary trees include:

- InOrder Traversal (Here, for insertion in of the elements in the tree)
- Preorder Traversal (Here, for display)
- Postorder Traversal (Here, for display)

Time Complexity for traversals O(n) **:**

The complexity function T(n) for tree traversal is :

$T(n) = T(k) + T(n - k - 1) + c$

where, k is the number of nodes on one side of root
n-k-1 on the other side.

Boundary conditions:

Case 1: Skewed tree (One of the subtrees is empty and other subtree is non-empty)
k is 0 in this case.
T(n) = T(0) + T(n-1) + c
T(n) = 2T(0) + T(n-2) + 2c
T(n) = 3T(0) + T(n-3) + 3c
T(n) = 4T(0) + T(n-4) + 4c

…………………………………………
…………………………………………

T(n) = (n-1)T(0) + T(1) + (n-1)c
T(n) = nT(0) + (n)c
Value of T(0) will be some constant say d. (traversing an empty tree will take some constants time)
T(n) = n(c+d)
T(n) = Θ(n) (Theta of n)

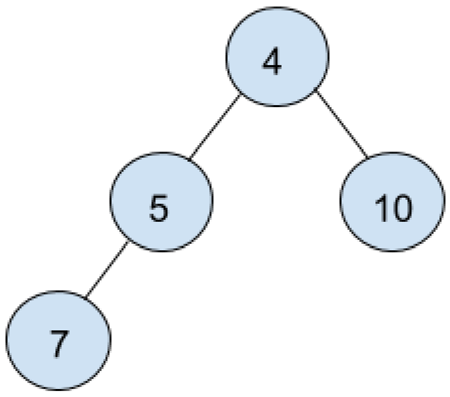
Case 2: Both left and right subtrees have an equal number of nodes.
T(n) = 2T(|_n/2_|) + c
Auxiliary Space: If we don’t consider the size of stack for function calls then O(1) otherwise O(n).

3. Stack
Stacks are the Last in First out (LIFO) type linear data type that only allows access to the top element. It was used in reversal of the queue operation.

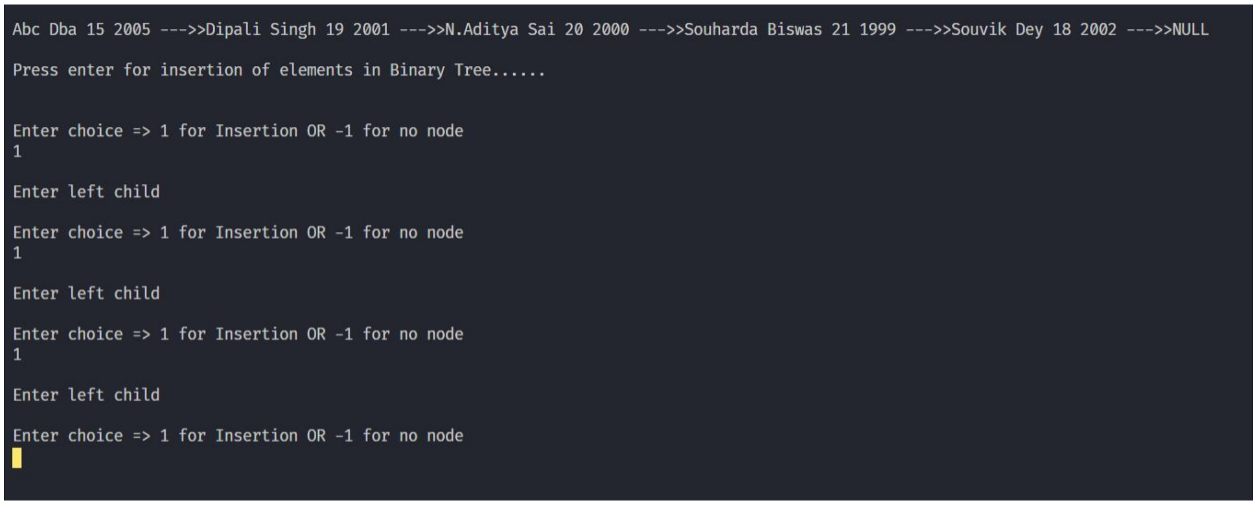
4. Linked List
Linked lists are linear data structures that consist of nodes where each node contains a data field and a reference to the next node in the list.

Code Implementation:

- The data is inserted in the main function from the text file using and is stored in **abc.txt**.
- The data from abc.txt is entered into the queue which is implemented using linked lists. Refer function **insert()**.
- The elements from the queue are inserted in the stack.
- The data from the queue is transferred into a Stack ADT to implement the process of Queue Reversal.
- The elements of the queue are displayed. Refer function **display()**.
- The data from the queue is processed into the binary tree using recursive dequeue.



The first node of the queue is entered in the root node (4 in fig) followed by a prompt to choose the right node or left node till we reach a leaf node (for eg 7) after which the loop backtracks due to the usage of the recursion to the nodes that are still empty. The data is thus successfully entered into the unordered binary tree. We integrated the queues and binary tree by only transferring the data part. The choice of node selection lies in the hands of the user.



```
Abc Dba 15 2005 --->>Dipali Singh 19 2001 --->>N.Aditya Sai 20 2000 --->>Souharda Biswas 21 1999 --->>Souvik Dey 18 2002 --->>NULL

Press enter for insertion of elements in Binary Tree......


Enter choice => 1 for Insertion OR -1 for no node
1

Enter left child

Enter choice => 1 for Insertion OR -1 for no node
1

Enter left child

Enter choice => 1 for Insertion OR -1 for no node
1

Enter left child

Enter choice => 1 for Insertion OR -1 for no node
```

- The entered data is then displayed using PreOrder and PostOrder traversal. Refer functions **traversePostOrder()**, **traversePreOrder()**.
- The data stored in the binary tree is then entered into the linked list using InOrder traversal.
- The data in the linked list is sorted with reference to the age. Refer function **quickSort().**
- The user has the option to add more records to the queue.

<u>**Work Distribution:**</u>

N Aditya Sai	-Queues and Stack Operation
Souvik Dey	-File I/O (Append, Reading)
Souharda Biswas	-Binary Tree ADT, Debugging
Dipali Singh	-Quick Sorting, Documentation