4th November 2020

Wednesday

Virtual Rube Goldberg Machine

Description

A Rube Goldberg Machine is a complex device that performs simple tasks in indirect and convoluted ways. We were to create a virtual Rube Goldberg Machine with ADTs like Dynamic Arrays, Queue, Stack, and Binary tree. Our program can support any number of entries. The data is read from a file and initially stored in a queue. It is then passed on to stacks and linked lists for further operations to be executed according to the scenario. A queue is a linear structure, which follows a particular order in which the operations are performed. The stack ADT allows all data operations at one end only. We could only access the top element of a stack, the element that is placed last is accessed first. In a stack, we remove the element the most recently added; in a queue, we remove the element the least recently added. The Binary Tree provided efficient insertion and searching in our program. The data structures we selected are:

1. Arrays
2. Queue
3. Stack
4. Binary tree

How to use our program?

- Firstly, the input given by the user is read from a file and stored in a queue. The user is asked to press 1 to read the data. Once this is initiated, the data is then displayed with its Serial Number and hence the input is successfully stored.
- To dequeue each element from the queue the user is then asked to press 2. Accordingly, the dequeued data is displayed and the queue is empty. To continue the processing the user is then asked to press Y.
- Now reversing the order of the data in the queue is done by dequeuing each element and pushing them onto a stack. The user is asked to press 1 to dequeue and push the elements to stack. Once the compilation is complete, the data printed is dequeued and pushed to stack successfully. The user should press 2 for the reverse order of stack to be printed.
- After the reversed data is printed, the user is asked to press 3 to pop off and requeue data one after the other. The popped elements are then printed and the stack is empty.
- Keeping in mind that the **Queue is preserved after all the operations executed** the user is then to press 4 to display the final data of the queue.

- To continue the operations, when the user presses Y, the data from the queue is placed into an unordered binary tree. The contents of the tree are then printed in Pre-Order where in this traversal method, the root node is visited first, then the left subtree, and finally the right subtree.
- The user is then asked to press 1, which results in printing the contents of the tree in the Post-Order where in this traversal method, the root node is visited last. First, we traverse the left subtree, then the right subtree, and finally the root node.
- The user will be then asked to press 1 to pull the contents from the tree and push it to a Linked–List using In-Order traversal. The contents of linked lists are then printed.
- The user is asked to press 1 to sort the contents of the list using a quick sort and printing it according to the birth year.
- The user is asked to press 1 to continue processing. We created an additional node where it would allow the user to be interactive i.e. to enter another name, age, and birthday. This is then inserted in the linked list in the proper location to maintain the sorted order.
- The contents of the list are printed and the user is asked to press any key to continue the processing. At this point, the user is done with the processing.
- The user is asked to press 1 to final exit the program after all processing are done successfully.

Analysis of our application

The performance of our program is efficient and our application is easy to use, easy to enter input, easy to read, and understand the output. It is dynamic i.e. any number of inputs can be given by the user and the application will work effectively following the given scenario systematically. As mentioned in the source code, the comments are clearly describing the problem scenario. If the user enters any number other than the said value; the program would show it as an invalid input and continue the process.

In terms of complexity:

Stacks and queues follow the principle of first-in-last-out (stacks) and first-in-first-out (queues). The time complexity for stacks is O(1) and the time complexity for queues is O(n).

Quick Sort time complexity:

Class	**Sorting algorithm**
Worst-case performance	$O(n^2)$
Best-case performance	O(n log n) (simple partition) or O(n) (three-way partition and equal keys)
Average performance	O(n log n)

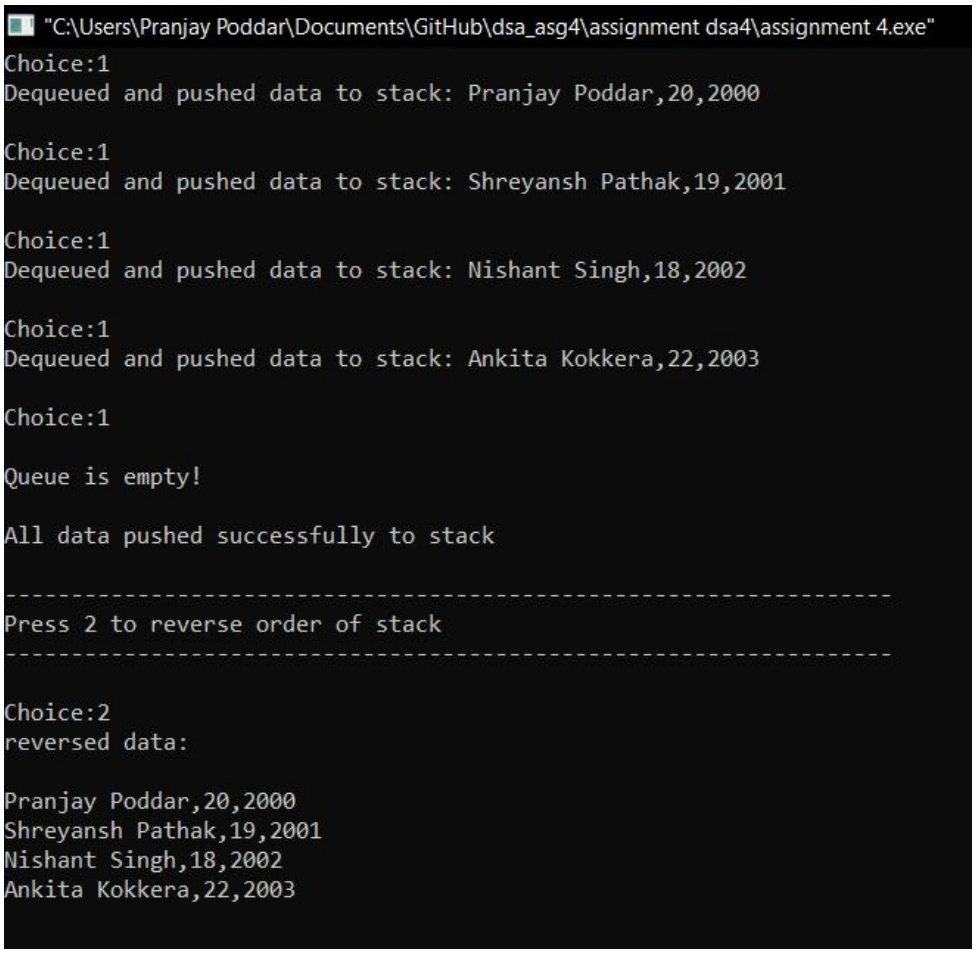
Our full working code:

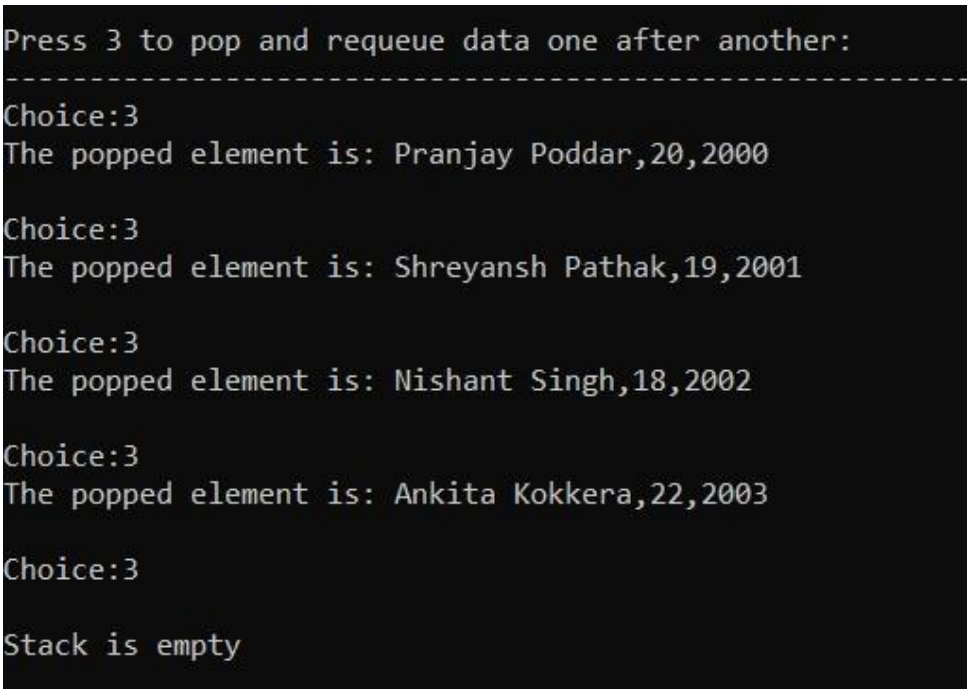
https://github.com/pranjay-poddar/dsa_asg4

Final outputs of the code:

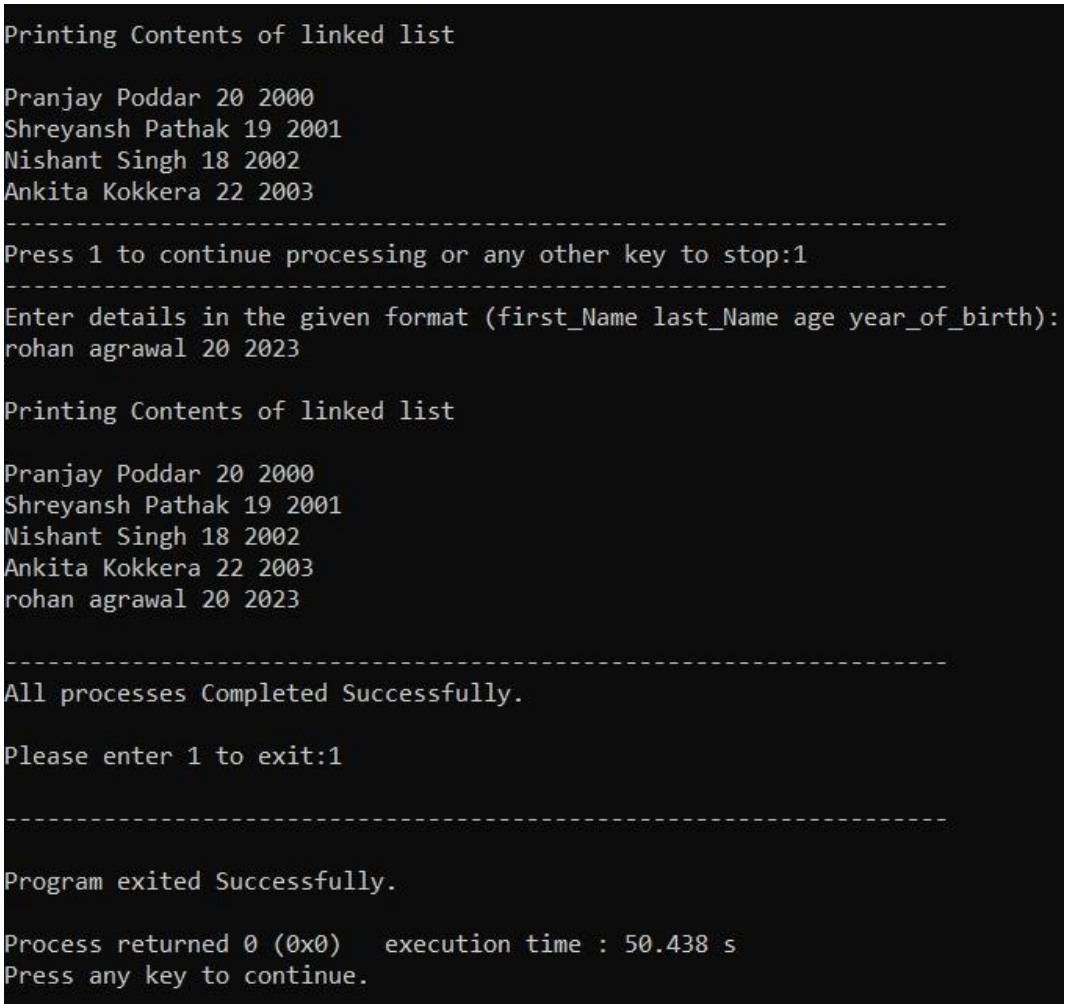
```
Press 1 to read data from file
Press 0 to exit
--------------------------------------------------------------------

Choice:1
Data Serial Number:1
Pranjay Poddar,20,2000

Choice:1
Data Serial Number:2
Shreyansh Pathak,19,2001

Choice:1
Data Serial Number:3
Nishant Singh,18,2002

Choice:1
Data Serial Number:4
Ankita Kokkera,22,2003

Data inputs Successfull
```

```
Press 2 to dequeue data
Press 0 to exit
--------------------------------------------------------------------

Choice:1
Please enter valid Input
Choice:2
Dequeued Data: Pranjay Poddar,20,2000

Choice:2
Dequeued Data: Shreyansh Pathak,19,2001

Choice:2
Dequeued Data: Nishant Singh,18,2002

Choice:2
Dequeued Data: Ankita Kokkera,22,2003

Choice:2

Queue is empty!

--------------------------------------------------------------------
Press Y to continue processing or any key to exit:Y
--------------------------------------------------------------------
Press 1 to dequeue and push element to stack
Press 0 to exit
```


"C:\Users\Pranjay Poddar\Documents\GitHub\dsa_asg4\assignment dsa4\assignment 4.exe"
Choice:1
Dequeued and pushed data to stack: Pranjay Poddar,20,2000
Choice:1
Dequeued and pushed data to stack: Shreyansh Pathak,19,2001
Choice:1
Dequeued and pushed data to stack: Nishant Singh,18,2002
Choice:1
Dequeued and pushed data to stack: Ankita Kokkera,22,2003
Choice:1
Queue is empty!
All data pushed successfully to stack
Press 2 to reverse order of stack
Choice:2
reversed data:
Pranjay Poddar,20,2000
Shreyansh Pathak,19,2001
Nishant Singh,18,2002
Ankita Kokkera,22,2003


Press 3 to pop and requeue data one after another:
Choice:3
The popped element is: Pranjay Poddar,20,2000
Choice:3
The popped element is: Shreyansh Pathak,19,2001
Choice:3
The popped element is: Nishant Singh,18,2002
Choice:3
The popped element is: Ankita Kokkera,22,2003
Choice:3
Stack is empty

```
Press 4 to display the final data of Queue
-----------------------------------------------------------------
Choice:4

Initial Queue is preserved

Final Queue data:

Pranjay Poddar,20,2000
Shreyansh Pathak,19,2001
Nishant Singh,18,2002
Ankita Kokkera,22,2003
-----------------------------------------------------------------
Press Y to continue or any key to exit:Y
-----------------------------------------------------------------
Printing Contents of tree in Pre-Order:

Pranjay Poddar 20 2000
Shreyansh Pathak 19 2001
Ankita Kokkera 22 2003
Nishant Singh 18 2002
-----------------------------------------------------------------
Press 1 to continue processing or any other key to stop:1
-----------------------------------------------------------------
Printing Contents of tree in Post-Order:

Ankita Kokkera 22 2003
Shreyansh Pathak 19 2001
Nishant Singh 18 2002
Pranjay Poddar 20 2000

-----------------------------------------------------------------
Press 1 to continue processing or any other key to stop:1
-----------------------------------------------------------------
Moving Contents of tree in a Linked List using In-Order traversal

Printing Contents of linked list

Ankita Kokkera 22 2003
Shreyansh Pathak 19 2001
Pranjay Poddar 20 2000
Nishant Singh 18 2002
-----------------------------------------------------------------
Press 1 to continue processing or any other key to stop:1
-----------------------------------------------------------------

Sorting Contents of linked List:

Printing Contents of linked list
```



Zero-kelvin Team Details-

- Shreyansh Pathak (RA1911028010097)
- Nishant Singh (RA1911028010108)
- Ankita Kokkera (RA1911028010111)
- Pranjay Poddar (RA1911028010129)