Data Structures Library

Submitted as Mini Project for DAA: Design and Analysis of Algorithms Semester IV

**BACHELOR OF TECHNOLOGY**

in

Computer Science and Engineering

Mehul Thakral

PES1201701122

Skanda VC

PES1201700987

Under Guidance of

Prof. Shruti Kaivalya

Assistant Professor

**Jan 2019- May 2019**

**Department of Computer Science and Engineering**

**PES UNIVERSITY**

Outer Ring Rd, Banashankari 3rd Stage, Banashankari, Bengaluru, Karnataka 560085

[www.pes.edu](http://www.pes.edu)

## Abstract

A high performance generic Library of Popular Data Structures for C implemented using macros for handling input of various data types and thus providing basic functionalities associated with each data structure implemented in algorithmically superior ways.Data structures provided involve namely vector, doubly linked list, stack, queue, heap, self-balancing tree(AVL tree), dictionary, matrix.User of the library can use a particular data structure by simply including the header file for the corresponding structure and usage of functions understood by going through a simple documentation.

## High Level Algorithm

### Vector: Functions implemented include initializing a new vector(with variable size input), referencing/assigning an element, pushing(with variable size input), popping an element and freeing the entire structure.

### Doubly Linked List: Functions implemented include initializing a new node, inserting a new node at head/tail/middle,deleting node from any position,printing DLL in forward/backward direction.

### Stack: Functions implemented include initializing a new node of stack, checking is stack empty, pushing a new element, popping a new element, reading element at top.

### Queue: Functions implemented include creating a new queue, creating a new node, enqueueing an element, dequeuing an element.

* **Dictionary** : Functions implemented include creating a new dictionary, inserting into a dictionary, inserting into dictionary, search a dictionary and deleting from a dictionary.
* **Heap :** Functions implemented include creating a new min or max heap, inserting into heap**,** deleting max/min element from heap
* **Matrix :** Functions implemented include functions for creating a new matrix, inserting elements dynamically rowise, multiply matrices, gaussian elimination, add and subtract 2 matrices, find transpose and inverse of matrices.
* **AVL Tree :** Functions implemented include creating a new avl tree, inserting into AVL tree, deleting from AVL tree and level order traversal.

## Test Results

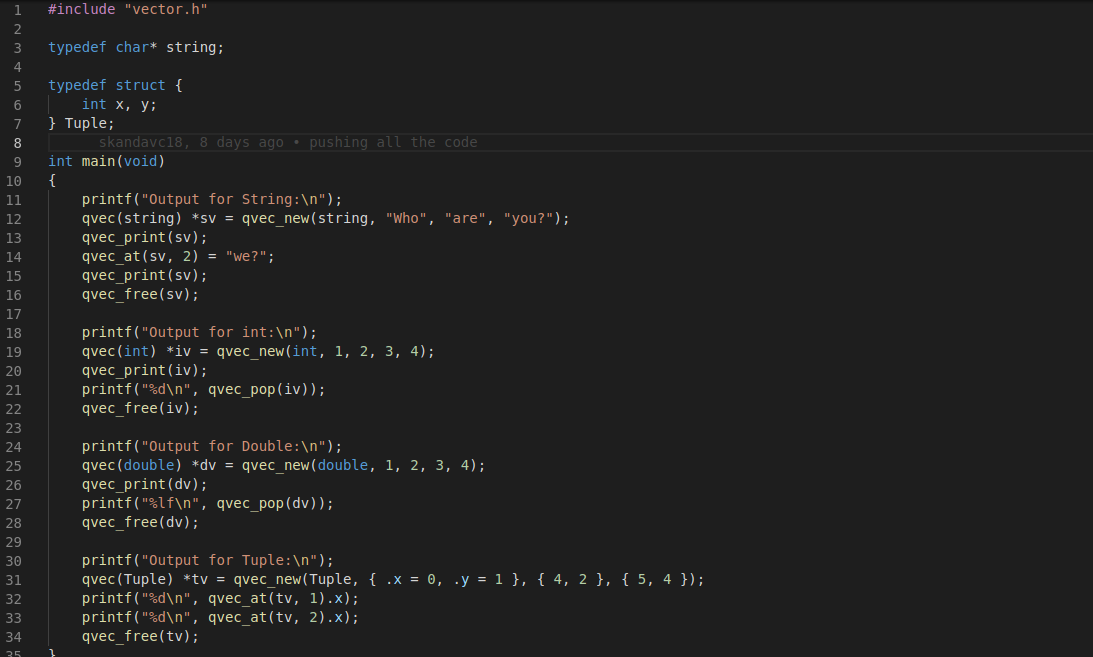


Fig 1: Showing typical usage of library particularly vector for different core data types( string[ char\*] , int, double ) and even for user defined data type( tuple ).

### Outputs obtained by using for different data types for each data structure:

#### Vector:

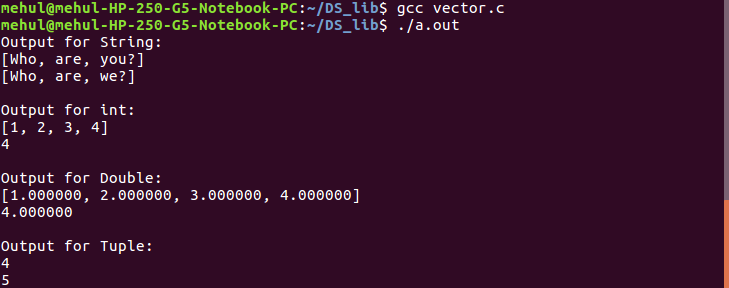


Fig 2: Showing output of library usage particularly vector for different core data types( string[ char\*] , int, double ) and even for user defined data type( tuple ).

#### Doubly linked list:

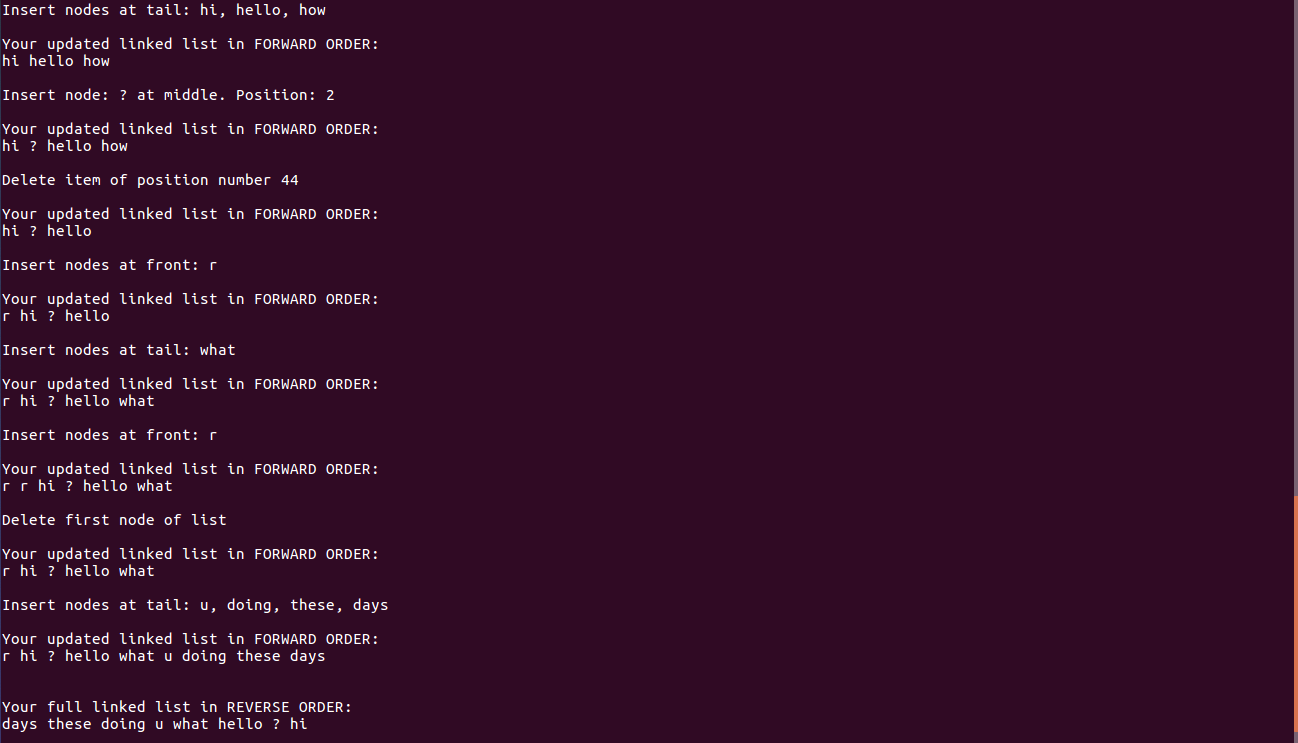


Fig 3: Showing output of library usage particularly doubly linked list for different core data types( string[ char\*] , int, double ).

#### Stack:

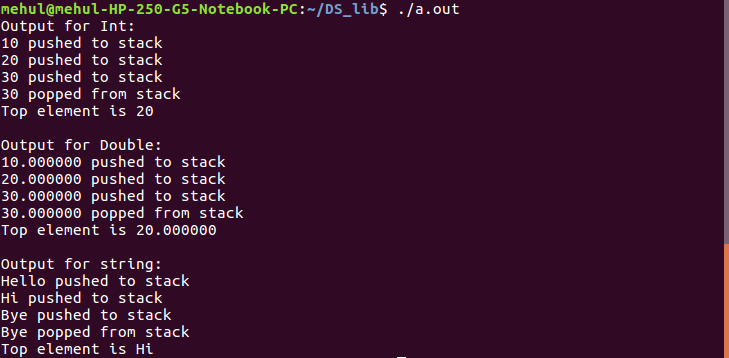


Fig 4: Showing output of library usage particularly stack for different core data types( string[ char\*] , int, double ).

#### Queue:

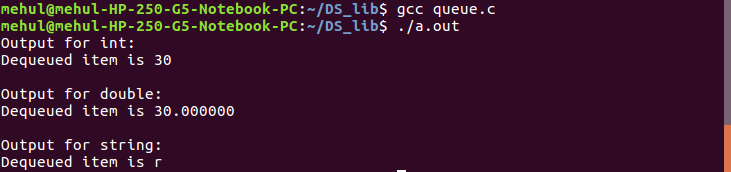


Fig 5: Showing output of library usage particularly queue for different core data types( string[ char\*] , int, double ).

* **Dictionary :**

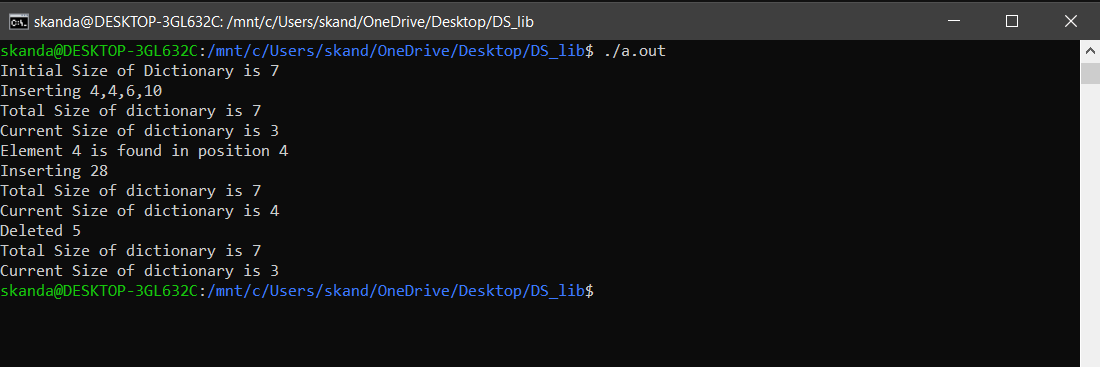
****

Fig 6: Showing output of library usage particularly for Dictionary of int type

* **Heap :**

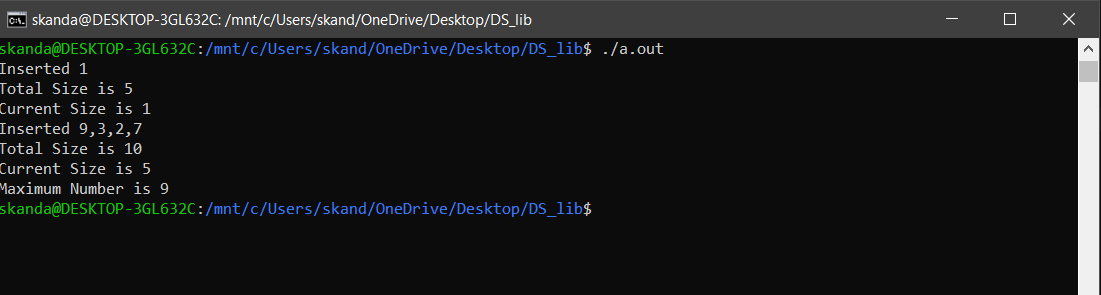
****

Fig 7: Showing output of library usage particularly for Max Heap of int type

* **Matrix :**

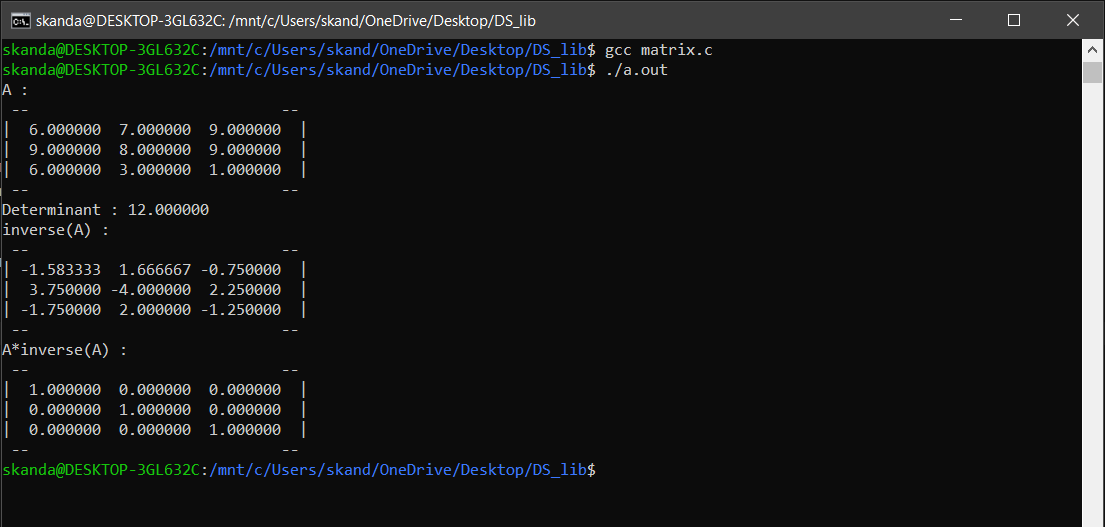
****

Fig 8: Showing output of library usage particularly for matrix of double type

* **AVL Tree :**

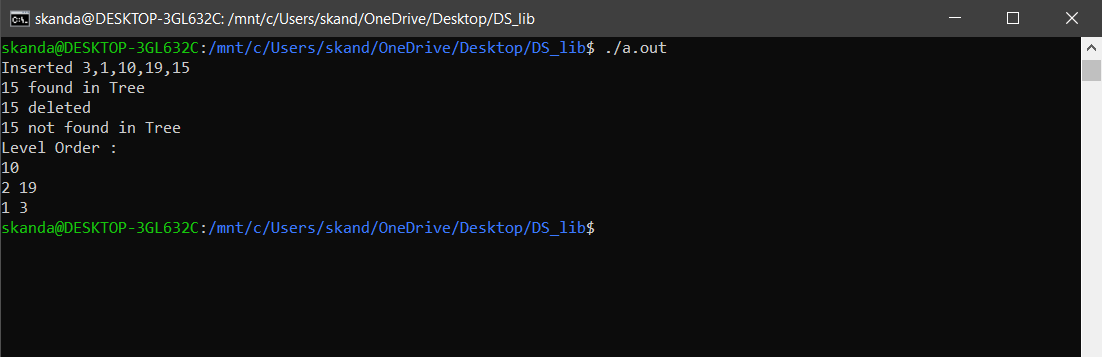
****

Fig 9: Showing output of library usage particularly for tree of int type