

***Name:*** *Taha Nawaz*

***Section:*** *Bssem-3A*

***ROLL NO:*** *Bssem-015*

***Assignment DSA Lab***

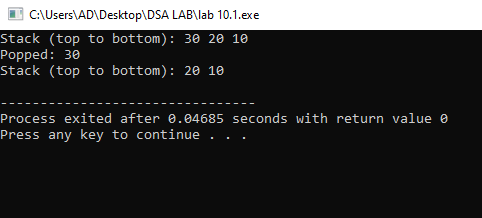
***SUBMITTED TO:***

***Sir Rasikh ALi***

***LAB TASK 10 T0 13***

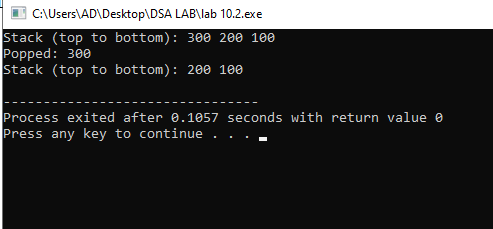
***TASK #10.1***

*This program implements a stack using a fixed-size array.  
It supports push, pop, and display operations based on the LIFO principle.  
The code handles overflow and underflow conditions safely.*

**

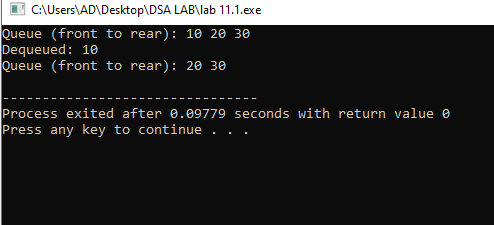
***TASK #10.2***

*This program implements a stack using a linked list.  
It supports push, pop, and display operations using dynamic memory.  
The stack follows the LIFO principle and avoids fixed size limitations****.***

**

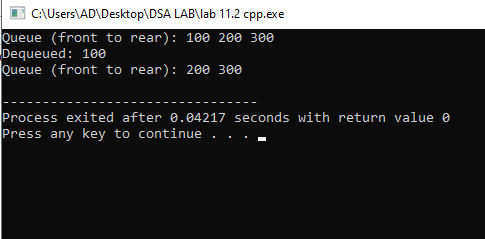
***TASK #11.1***

*This program implements a queue using a fixed-size array.  
It supports enqueue, dequeue, and display operations in FIFO order.  
Overflow and underflow conditions are handled properly.*

******

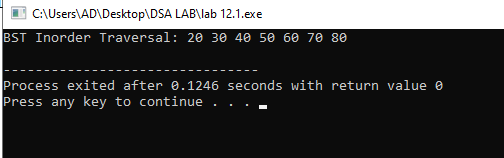
***TASK #11.2***

*This program implements a queue using a linked list.  
It supports enqueue, dequeue, and display operations in FIFO order.  
It dynamically allocates memory, avoiding fixed size limitations.*

**

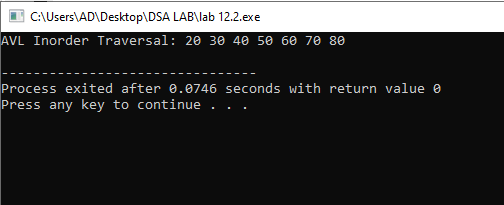
***TASK #12.1***

*This program implements a Binary Search Tree with insertion and inorder traversal.  
Nodes are inserted recursively based on BST rules (left < root < right).  
The inorder traversal displays values in ascending sorted order.*

**

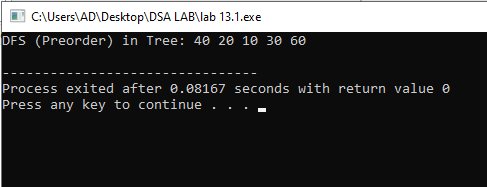
***TASK #12.2***

*This program implements an AVL tree with insertion and inorder traversal.  
It maintains balance using rotations after insertions to ensure O(log n) operations.  
The inorder traversal displays elements in ascending order from the balanced tree..*

**

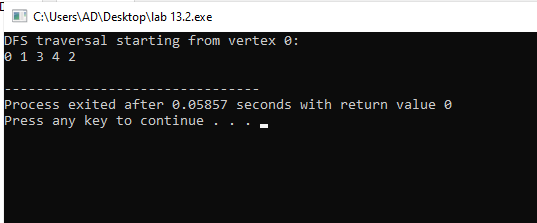
***TASK #13.1***

*This program implements DFS (Preorder) traversal in a binary tree.  
It builds the tree using insertions and visits nodes in Root-Left-Right order.  
DFS is used to explore all nodes deeply before backtracking****.***

******

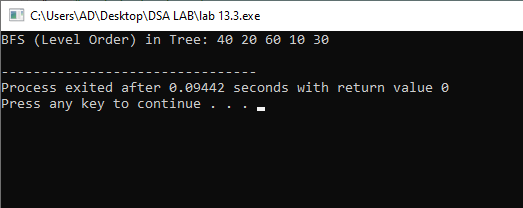
***TASK #13.2***

*This C++ code defines a graph using an adjacency matrix and performs Depth First Search (DFS) traversal starting from a given vertex.  
It inserts directed edges between vertices using insertEdge(u, v).  
The DFS() function uses recursion (DFSUtil) to visit and print each reachable node from the start vertex****.***

******

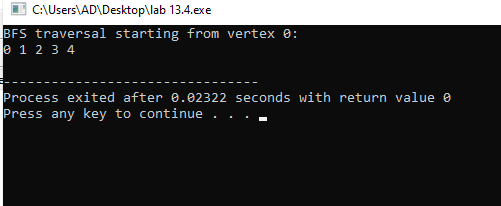
***TASK #13.3***

*This program implements BFS (Level Order) traversal in a binary tree.  
It builds the tree using insertions and explores nodes level by level using a queue.  
BFS visits nodes in the order of their depth, from left to right.*

******

***TASK #13.4***

*This C++ program creates a graph using an* ***adjacency matrix*** *and performs* ***Breadth First Search (BFS)*** *starting from a specified vertex.  
It uses a manual queue to explore vertices level by level and marks each visited vertex to avoid cycles.  
Edges are added with insertEdge(u, v), and BFS(0) prints the traversal from vertex 0.*

**