Lab Manual
Course: Data Structures

Week 1:

I. Given an array of integers, write an algorithm and a program to left rotate the array by specific number of elements.

Input format:
The first line contains number of test cases, T.
For each test case, there will be three input lines.
First line contains n (the size of array).
Second line contains n space-separated integers describing array.
Third line will take number (k<=n) of times elements of array is rotated.

Output format:
The output will have T number of lines.
For each test case, output will be the new rotated array.

Sample:

Input:	**Output:**
3 5 2 4 -3 2 3 3 10 1 6 3 2 9 1 4 2 3 6 4 15 -2 8 7 1 2 6 8 9 0 2 -6 2 9 7 1 6	2 3 2 4 -3 9 1 4 2 3 6 1 6 3 2 8 9 0 2 -6 2 9 7 1 -2 8 7 1 2 6

II. Given an unsorted array of integers and two numbers a and b. Design an algorithm and write a program to find minimum distance between a and b in this array. Minimum distance between any two numbers a and b present in array is the minimum difference between their indices.

Input format:
The first line contains number of test cases, T.
For each test case, there will be three input lines.
First line contains n (the size of array).
Second line contains n space-separated integers describing array.
Third line will take two numbers a and b.

Output format:
The output will have T number of lines.
For each test case, output will be the minimum distance between a and b.

Sample:

Input:	**Output:**

3 5 2 4 -3 2 3 4 2 10 1 6 3 2 9 1 4 3 6 2 3 2 15 -2 8 7 1 2 6 8 9 0 2 -6 12 9 7 1 7 2	1 1 2

III. Given an array of nonnegative integers, where all numbers occur even number of times except one number which occurs odd number of times. Write an algorithm and a program to find this number. (Time complexity = O(n))

Input format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output format:
The output will have T number of lines.
For each test case, output will be the element which occured odd number of times. If no such element is present in the array, then print "**No such element present**".

Sample:

Input:	**Output:**
3 5 2 4 3 2 3 11 1 6 3 2 4 1 4 2 3 6 6 15 2 8 7 1 2 6 8 9 0 2 6 2 9 7 1	4 6 0

Week 2:

I. Given a matrix of size n X n, where every row and every column is sorted in increasing order. Write an algorithm and a program to find whether the given key element is present in the matrix or not. (Linear time complexity)

Input Format:
First line contains n (the size of matrix).
For next n input lines, each line contains space-separated n integers describing each row of the matrix.
Last line of input will contain key integer to be searched

Output Format:

Output will be "**Present**" if the key element is found in the array, otherwise print "**Not Present**".

Sample:

Input:	**Output:**
4 10 20 30 40 15 25 34 41 27 29 35 45 32 33 38 49 33	Present

II. Given a boolean matrix (contains only 0 and 1) of size m X n where each row is sorted, write an algorithm and a program to find which row has maximum number of 1's. (Linear time complexity)

Input Format:

First line contains m and n (the size of matrix).

For next m input lines, each line contains space-separated n integers describing each row of the matrix.

Output Format:

Output will be row number which has maximum number of 1's. If all the elements of matrix is 0 then print "**Not Present**".

Sample:

Input:	**Output:**
4 3 0 1 1 0 0 1 1 1 1 0 0 0	row - 3

III. Given a matrix of size n X n containing positive integers, write an algorithm and a program to rotate the elements of matrix in clockwise direction.

Input Format:

First line contains n (the size of matrix).

For next n input lines, each line contains space-separated n integers describing each row of the matrix.

Output Format:

Output will be rotated matrix.

Sample:

Input:	**Output:**
4 11 12 13 14 15 16 17 18	15 11 12 13 19 20 16 14 23 21 17 18

19 20 21 22 23 24 25 26	24 25 26 22

Week 3:

I. Design an algorithm and a program to implement stack using array. The program should implement following stack operations:

a) Create() - create an empty stack
b) Push(k) - push an element k into the stack
c) Pop() - pop an element from the stack snd return it
d) IsEmpty() - check if stack is empty or not
e) Size() - finds the size of the stack
f) Print() - prints the contents of stack

Input Format:
Ask user first whether it wants to push/pop/find the size of the stack, then accordingly perform operation.

Output Format:
Output will be the final stack contents if push or pop operation is performed. Output will be size of the stack if user asks for size.

Sample:

Input:	**Output:**
Press: 1 to push 2 to pop 3 to calculate size 4 to exit 1 34 1 42 1 6 3 4	Stack - 34 Stack - 34 42 Stack - 34 42 6 Size = 3

II. Given an expression string consisting of opening and closing brackets “{“,”}”,”(“,”)”,”[“,”]”, design an algorithm and a program to check whether this expression has balanced paranthesis or not.

Input Format:
The first line contains number of test cases, T.
For each test case, there will be expression string.

Output Format:
The output will have T number of lines. For each test case, output will be “**Balance**”, if brackets are balanced otherwise print “**Unbalanced**”.

Sample:

Input:	**Output:**
3 {{(()())}} ([][])(){(())} {()(()}	Balanced Balanced Unbalanced

III. Given a string of opening and closing paranthesis, design an algorithm and a program to find the length of the longest valid paranthesis substring. Valid paranthesis substring is a string which contains balanced paranthesis.

Input Format:
The first line contains number of test cases, T.
For each test case, there will be string of paranthesis.

Output Format:
The output will have T number of lines. For each test case, output will be length of longest valid paranthesis substring

Sample:

Input:	**Output:**
3 ()()))) ((()()((()()(()))()	4 6 12

Week 4:

I. Given an empty stack, design an algorithm and a program to reverse a string using this stack (with and without recursion).

Input Format:
The first line contains number of test cases, T.
For each test case, there will be string of characters.

Output Format:
The output will have T number of lines. For each test case, output will be new reversed string.

Sample:

Input:	**Output:**
3 qwerty computerscience graphicerauniversity	ytrewq ecneicsretupmoc ytisrevinuarecihparg

II. Design an algorithm and a program to implement two stacks by using only one array i.e. both the

stacks should use the same array for push and pop operation. Array should be divided in such a manner that space should be efficiently used if one stack contains very large number of elements and other one contains only few elements.

Input Format:
First line will take size of stack.
Second line will ask user whether it wants to push/pop in stack1 or stack2, then accordingly perform operation.

Output Format:
Output will be the final contents of both stacks after performing all the operations.

Sample:

Input:	**Output:**
10 Press: 1 to push in stack 1 2 to pop from stack 1 3 to push in stack 2 4 to pop from stack 2 5 to exit 1 342 1 564 3 601 3 970 3 123 5	stack 1: 342 564 stack 2: 601 970 123

III. Given an expression in the form of postfix representation, design an algorithm and a program to find result of this expression.

Input Format: The first line contains number of test cases, T.
For each test case, there will be expression string.

Output Format: The output will have T number of lines. For each test case, output will be the result of the evaluated expression.

Sample:

Input:	**Output:**
3 2 3 1 * + 4 - 40 8 / 5 * 10 2 / - 40 8 / 5 2 * -	1 20 -5

Week 5:

I. Design an algorithm and a program to implement queue using array. The program should

implement following queue operations:

a) Create() - create a queue
b) EnQueue(k) - insert an element k into the queue
c) DeQueue() - delete an element from the queue
d) IsEmpty() - check if queue is empty or not
e) Size() - finds the size of the queue

Input Format:
Ask user first whether it wants to insert/delete/find size of the queue, then accordingly perform operation.

Output Format:
Output will be the final queue contents after every operation is performed.
Output will be size of the queue if user asks for size.

Sample:

Input:	**Output:**
Press: 1 to enqueue 2 to dequeue 3 to calculate size 4 to exit 1 34 1 42 1 6 3 4	Queue - 34 Queue - 34 42 Queue - 34 42 6 Size = 3

II. Design an algorithm and write a program to reverse a queue.

Input format: Queue elements are taken as an input.

Output Format: Output will be update reverse queue.

Sample:

Input:	**Output:**
Press: 1 to insert 2 to exit 1 924 1 380 1 206 2	Initial Queue : 924 380 206 Reverse Queue : 206 380 924

III. Design an algorithm and write a program to implement Deque i.e. double ended queue. Double ended queue is a queue in which insertion and deletion can be done from both ends of the queue. The program should implement following operations:

a) insertFront() - insert an element at the front of Deque

b) insertEnd() - insert an element at the end of Deque
c) deleteFront() - delete an element from the front of Deque
d) deleteEnd() - delete an element from the end of Deque
e) isEmpty() - checks whether Deque is empty or not
f) isFull() - checks whether Deque is full or not
g) printFront() - print Deque from front
h) printEnd() - print Deque from end

Input Format:
Ask user first whether it wants to insert/delete/ at the front or end of the queue, then accordingly perform operation.

Output Format:
Output will be the final queue contents from both directions if enqueue or dequeue operation is performed.

Sample:

Input:	**Output:**
Press: 1 insert at front 2 insert at end 3 delete from front 4 delete from end 5 to exit 1 382 1 299 2 543 5	Contents of queue from front - 382 299 543 Contents of queue from end - 543 299 382

Week 6:

I. Design an algorithm and write a program to implement stack operations using minimum number of queues.

Input Format:
Ask user first whether it wants to push/pop/find the size of the stack, then accordingly perform operation.

Output Format:
Output will be the final stack contents if push or pop operation is performed.
Output will be size of the stack if user asks for size.

Sample:

Input:	**Output:**
Press: 1 to push 2 to pop 3 to exit 1 349 1	Stack - 349 Stack - 349 427 Stack - 349 427 610 Element popped from stack – 610 Stack after pop operation – 349 427

427 1 610 2 3	

II. Design an algorithm and write a program to implement queue operations using minimum number of stacks.

Input Format:
Ask user first whether it wants to insert into or delete from queue, then accordingly perform operation.

Output Format:
Output will be the final queue contents after every operation.

Sample:

Input:	**Output:**
Press: 1 to enqueue 2 to dequeue 3 to exit 1 816 1 398 1 961 2 3	Queue - 816 Queue - 816 398 Queue - 816 398 961 Element deleted from queue - 816 Queue after deletion- 398 961

III. Design an algorithm and write a program to implement circular queue. Circular queue is a queue in which last position of queue is connected with first position of queue to make a circle. The program should implement following operations:

a) Create() - create a queue of specific size
b) EnQueue(k) - insert an element k into the queue
c) DeQueue() - delete an element from the queue
d) IsEmpty() - check if queue is empty or not
e) Front() - return front item from queue
f) Rear() - return rear item from queue

Input Format:
Ask user first whether it wants to insert/delete/find front or rear of the queue, then accordingly perform operation.

Output Format:
Output will be the final queue contents after every operation is performed.
Output will be front/rear item of the queue if user asks for it.

Sample:

Input:	**Output:**
Enter size of circular queue : 4	Circular Queue : 816

Press: 1 to enqueue 2 to dequeue 3 to find front 4 to find rear 5 to exit 1 816 1 398 1 961 2 4 5	Circular Queue : 816 398 Circular Queue : 816 398 961 Circular Queue : 398 961 Rear : 961

Week 7:

I. Write an algorithm and a program to implement singly linked list. The program should implement following operations:
   a) Create(k) - create a linked list with single node of value k
   b) InsertFront(k) - insert node of value k at the front of the linked list
   c) InsertEnd(k) - insert a node of value k at the end of the linked list
   d) InsertAnywhere(p) - insert a node at specific position p
   e) DeleteFront() - delete a node from the front of the linked list
   f) DeleteEnd() - delete a node from the end of the linked list
   g) DeleteAnywhere(p) - delete a node from a specific position p
   h) Size() - find the size of the linked list
   i) IsEmpty() - checks if the linked list is empty or not
   j) FindMiddle() - finds the middle element of the linked list

Input Format:
Ask user first whether it wants to insert/delete/find size/ find middle element of the list then accordingly perform operation.

Output Format:
Output will be the final contents of the list after every operation is performed.
Output will be size of the list if user asks for size.

Sample:

Input:	**Output:**
Press: 1 insert node at front 2 insert node at end 3 insert node at specific position 4 delete node from front 5 delete node from end 6 delete node from specific position 7 find size of linked list 8 find middle element of the list 9 exit 2 568 1 894 4	Linked List – 568 Linked List – 894 568 Node deleted – 894 Linked List after deletion - 568

9	

II. Write an algorithm and a program to implement queue using linked list. The program should implement following stack operations:
   a) Create()
   b) EnQueue()
   c) DeQueue()
   d) IsEmpty()
   e) Size()

Input Format:
Ask user first whether it wants to insert/delete/find size of the queue, then accordingly perform operation.

Output Format:
Output will be the final queue contents after every operation is performed.
Output will be size of the queue if user asks for size.

Sample:

Input:	**Output:**
Press: 1 to enqueue 2 to dequeue 3 to calculate size 4 to exit 1 34 1 42 1 6 3 4	Queue - 34 Queue - 34 42 Queue - 34 42 6 Size = 3

III. Write an algorithm and a program to implement stack using linked list. he program should implement following queue operations:
   a) Create()
   b) Push()
   c) Pop()
   d) IsEmpty()
   e) IsFull()
   f) Size()

Input Format:
Ask user first whether it wants to push/pop/find the size of the stack, then accordingly perform operation.

Output Format:
Output will be the final stack contents if push or pop operation is performed.
Output will be size of the stack if user asks for size.

Sample:

Input: Press: 1 to push 2 to pop 3 to calculate size 4 to exit 1 34 1 42 1 6 3 4	**Output:** Stack - 34 Stack - 34 42 Stack - 34 42 6 Size = 3

Week 8:

I. Write an algorithm and a program to implement doubly linked list. The program should implement following operations:
   a) Create(k) - create a doubly linked list node with value k.
   b) InsertFront(k) - insert node at the front of the linked list.
   c) InsertEnd(k) - insert a node at the end of the linked list.
   d) InsertIntermediate(k,p) - insert a node at specific position p.
   e) DeleteFront() - delete a node from the front of the linked list.
   f) DeleteEnd() - delete a node from the end of the linked list.
   g) DeleteIntermediate(p) – delete a node from a specific position p.
   h) Size() - returns the size of doubly linked list.
   i) IsEmpty() - checks whether the list is empty or not.
   j) FindMiddle() - returns the contents of middle node of the list.

Input Format:
Ask user first whether it wants to insert/delete/find size/ find middle element of the list then accordingly perform operation.

Output Format:
Output will be the final contents of the list after every operation is performed.
Output will be size of the list if user asks for size.

Sample:

Input: Press: 1 insert node at front 2 insert node at end 3 insert node at specific position 4 delete node from front 5 delete node from end 6 delete node from specific position 7 find size of linked list 8 find middle element of the list 9 exit 2 568 1 894 4 9	**Output:** Doubly Linked List – 568 Doubly Linked List – 894 568 Node deleted – 894 Doubly Linked List after deletion - 568

II. Given a doubly linked list, design an algorithm and write a program to reverse this list without using any extra space.

Input Format: Elements to insert into the list are taken as an input.

Output Format: First line of output will give original linked list. Second line of output will give updated reversed linked list.

Sample:

Input:	**Output:**
Press: 1 to insert element to linked list 2 to exit 1 689 1 961 1 731 2	Doubly Linked List : 689 961 731 Rversed Doubly Linked List : 731 961 689

III. Given a sorted doubly linked list, design an algorithm and write a program to eliminate duplicates from this list.

Input Format: Elements to insert into the list are taken as an input.

Output Format:
First line of output will give original linked list.
Second line of output will give updated linked list with no duplicate elements.

Sample:

Input:	**Output:**
Press: 1 to insert element to linked list 2 to exit 1 689 1 961 1 731 1 961 2	Doubly Linked List : 689 961 731 961 Doubly Linked List after duplicate removal : 986 961 731

Week 9:

I. Design an algorithm and write a program to implement circular linked list. The program should implement following operations:
   a) Create(k) - creates a circular linked list node with value k.
   b) InsertFront() - insert node at the front of list.

c) InsertEnd() - insert node at the end of list.
d) InsertIntermediate() - insert node at any specified position of list.
e) DeleteFront() - delete node from the front of list.
f) DeleteEnd() - delete node from the end of list.
g) DeleteIntermediate() - delete node from any specified position of list.
h) Size() - return size of circular linked list.
i) IsEmpty() - checks whether list is empty or not.

Input Format:
Ask user first whether it wants to insert/delete/find size/ find middle element of the list then accordingly perform operation.

Output Format:
Output will be the final contents of the list after every operation is performed.
Output will be size of the list if user asks for size.

Sample:

Input:	**Output:**
Press: 1 insert node at front 2 insert node at end 3 insert node at specific position 4 delete node from front 5 delete node from end 6 delete node from specific position 7 find size of linked list 8 exit 2 568 1 894 1 157 4 8	Circular Linked List – 568 Circular Linked List – 894 568 Circular Linked List – 157 894 568 Node deleted – 157 Circular Linked List after deletion – 894 568

II. Design an algorithm and write a program to concatenate two circularly linked lists.

Input Format:
First n lines of input contain elements of first circular linked list.
Next m lines contain elements of second circular linked list.

Output Format:
First line of output will give first circular linked list.
Second line of output will give second circular linked list.
Third line of output will give updated linked list after merging both the lists.

Sample:

Input:	**Output:**
Press: 1 to insert node 2 to exit 1 864 1	First circular linked list: 864 628 193 Second circular linked list : 197 496 Final Concatenaed linked list : 864 624 193 197 496

628 1 193 2 Press: 1 to insert node 2 to exit 1 197 1 496 2	

III. Write an algorithm and a program that will split a circularly linked list into two circularly linked list provided position from where circular linked list has to be splitted.

Input Format:
Elements to insert into linked list are taken as an input.
Last line of input will take node number k from where list has to be splitted.

Output Format:
First line of output will give original linked list.
Second line of output will give first part of splitted linked list.
Third line of output will give other part of splitted linked list.

Sample:

Input:	**Output:**
Press: 1 to insert 2 to exit 1 864 1 628 1 193 1 496 2 3	Original List : 864 628 193 496 First part of list : 864 628 193 Second part of list : 496

Week 10:

I. Write an algorithm that will split a linked list into two linked lists, so that successive nodes go to different lists. Odd numbered nodes to the first list while even numbered nodes to the second list.

Input Format: Elements to be inserted are taken as an input.

Output Format:
First line of output will give original linked list.
Second line of output will give linked list formed by taking odd numbered nodes.
Third linke of output will give linked list formed by taking even numbered nodes.

Sample:

Input: Press: 1 to insert node 2 to exit 1 380 1 234 1 963 1 125 1 154 2	**Output:** Original List : 380 234 963 125 154 Odd numbered nodes list : 380 963 154 Even numbered nodes list: 234 125

II. Given a linked list and a number n, design an algorithm and write a program to find the value at the n'th node from end of this linked list.

Input Format: Elements to be inserted in the linked list are taken as an input.
Last line of input will take n.

Output Format:
First line of output will give original linked list.
Second line of output will be the value present at n'th node from the end.

Sample:

Input: Press: 1 to insert node 2 to exit 1 380 1 963 1 125 1 154 2 Enter value n: 3	**Output:** Linked List : 380 963 125 154 Value at position 3 from end : 963

III. Write a program that will reverse a linked list only by traversing it once and without using extra space.

Input Format: Elements to be inserted in the linked list are taken as an input.

Output Format:
First line of output will give original linked list.
Second line of output will be reversed linked list.

Sample:

Input: Press:	**Output:** Original Linked List : 780 563 125 154

1 to insert node 2 to exit 1 780 1 563 1 125 1 154 2	Reversed Linked List : 154 125 563 780

Note: Consider the following input format for tree based questions.

Input Format: Tree elements are provided in the form of array for input. Consider example of below given tree, its array represenation will be: 1 2 3 4 -1 5 6 7 8 -1 -1 9 -1 -1 -1, where for a node i its children are present at node 2i+1 and 2i+2. Here array representation of the tree represents full binary tree.
Note: index of array must start from 0. Here -1 represents NULL.
Next step is to create tree for input array. For each element in the array, a tree node is created and its value is inserted in that node. Tree representation of the array is shown in Figure (a). For the array indices where value = -1, since its value is NULL therefore node for NULL values should not be created. Hence, the actual tree should look like Figure (b).

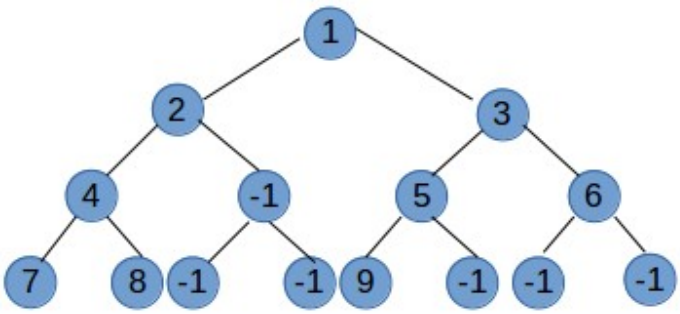

Figure (a)

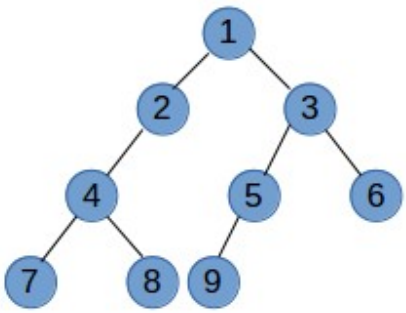

Figure (b)

Week 11:

I. Design an algorithm to implement binary trees. Write a program which implements following basic operations on binary tree:

a) CreateTree() - creates a root node
b) InsertNode() - insert a node in tree
c) DeleteNode() - delete a node from tree
d) FindHeight() - find the height of tree
e) FindSize() - find number of nodes in tree
f) Search() - search whether given specific element present in tree or not.

Input Format:
First line of input will take size of array form of tree. Size of array will be on the basis of full binary tree.
Second line of input contains n space-separated integers describing array.
Third line of input will ask user to insert / delete a node or not.

Output Format:
First output line will be final tree contents.
Second line of output will give height of tree.
Third line of output will give number of nodes in tree.

Sample:

Input:	**Output:**
15 1 2 3 4 -1 5 6 7 8 -1 -1 9 -1 -1 -1 Press: 1 to insert a node 2 to delete a node 3 to exit 3	Tree: 1 2 3 4 5 6 7 8 9 Height = 3 Size = 9

II. Write an algorithm and a program to implement following types of traversing in the tree:

a) inorder() - (1) traverse left subtree, (2) traverse root, (3) traverse right subtree
b) postorder() - (1) traverse root, (2) traverse left subtree, (3) traverse right subtree
c) preorder() - (1) traverse left subtree, (2) traverse right subtree, (3) traverse root

Input Format:
First line of input will take size of array form of tree. Size of array will be on the basis of full binary tree.
Second line of input contains n space-separated integers describing tree in the form of array.

Output Format:
First output line will give inorder traversal of the tree.
Second output line will give postorder traversal of the tree.
Third output line will give preorder traversal of the tree.

Sample:

Input:	**Output:**
15 1 2 3 4 -1 5 6 7 8 -1 -1 9 -1 -1 -1	Inorder : 7 4 8 2 1 9 5 3 6 Postorder : 7 8 4 2 9 5 6 3 1 Preorder : 1 2 4 7 8 3 5 9 6

III.For an expression in the form of binary tree, infix representation of the expression is given in the form of character array.. Design an algorithm and a program to convert infix expression of this tree to postfix expression. Postfix representation of expression should follow BODMAS rule.

Input Format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated characters describing array.

Output Format:
The output will have T number of lines. For each test case, output will be the postfix

representation for the corresponding infix expression.

Sample:

Input:	**Output:**
2 a+b*c a+b*(c^d-e)^(f+g*h)-i	abc*+ abcd^e-fgh*+^*+i-

Week 12:

I. Design an algorithm and a program to implement binary search tree. The program should implement following BST operations:

a) Create() - creates a root node.
b) Insert() - insert a node in tree.
c) Delete() - delete a node from tree.
d) FindHeight() - return height of tree.
e) FindDepth() - return depth of tree.
f) Search() - search whether given key is present in tree or not.

Input Format:

First line of input will take size of array form of tree. Size of array will be on the basis of full binary tree.
Second line of input contains n space-separated integers describing tree in the form of array.
Third line of input will ask user first whether it wants to insert/delete/search a node from tree or not.

Output Format:

First output line will be level order traversal of final tree contents.
Second line of output will give height of tree.
Third line of output will give depth of tree.
Fourth line of output will be only shown when user asks to search a key in tree. If key is present in tree, output will be "**key – present**", otherwise prints "**Not present**".

Sample:

Input:	**Output:**
15 4 2 8 1 3 7 -1 -1 -1 -1 -1 6 -1 -1 -1 Press: 1 to insert a node 2 to delete a node 3 to search a key 4 to exit 3 Enter key: 7 4	BST: 4 2 8 1 3 7 6 Height = 3 Depth = 3 7 - present

II. Design an algorithm and a program to convert binary search tree created in previous question into balanced BST. Also find maximum and minimum element from this balanced BST.

Input Format:

First line of input will take size of array form of tree. Size of array will be on the basis of full binary tree.

Second line of input contains n space-separated integers describing tree in the form of array.

Output Format:

First output line will be level order traversal of final tree contents.
Second output line will be level order traversal of balanced tree contents.
Third output line will give maximum element of tree.
Fourth output line will give minimum element of tree.

Sample:

Input:	**Output:**
15 4 3 5 2 -1 -1 6 1 -1 -1 -1 -1 -1 -1 7	Initial BST: 4 3 5 2 6 1 7 Balanced BST: 4 2 6 1 3 5 7 Maximum element: 7 Minimum element: 1

III. Given a binary search tree, design an algorithm and write a program to find which level number of tree has maximum number of nodes. Also print maximum number of nodes present at this level.

Input Format:

First line of input will take size of array form of tree. Size of array will be on the basis of full binary tree.
Second line of input contains n space-separated integers describing tree in the form of array.

Output Format:

First output line will be level order traversal of final tree contents.
Second output line will give level number and number of nodes present on that level.
If all the levels have equal number of nodes, then print "**Equal no. Of nodes on all levels**".

Sample:

Input:	**Output:**
5 2 6 1 3 -1 7 -1 -1 -1 4 -1 -1 -1 8	BST: 5 2 6 1 3 7 4 8 Level 3 has 3 nodes

Week 13:

I. Write an algorithm and a program to implement priority queue (also known as heap). Priority queue has following properties:

a) Each item has a priority associated with it.
b) Element which has heighest priority is dequeued before an element with low priority.
c) If two elements have the same priority, they are served according to their order in the queue.
d) It is always in the form of complete tree.
e) Element which has highest priority should be at root, element which has priority less than root but more than other elements comes at level 2. Hence as the level increases, priority of element decreases.

The program should implement following operations:

a) create() - create root node of priority heap
b) insert() - insert an element into the priority queue. Everytime when a node is inserted, tree should be balanced according to its priority.
c) getHighestPriority() - finds the element which has highest priority
d) deleteHeighestPriority() - delete the element which has highest priority

Input Format:
First line of input will take size of array form of tree.
Second line of input contains n space-separated integers describing priorities of n elements in the form of array. (Suppose highest priority element has largest value).
Third line of input will ask user first whether it wants to insert into/delete a node from tree or not.

Output Format:
First output line will give level order traversal of tree.
Second output line will give highest priority of element
Third output line will give level order traversal of updated tree if new node is inserted, otherwise prints priority of element which is deleted.

Sample:

Input:	**Output:**
7 4 5 1 6 7 3 8 Press: 1 to insert node 2 to delete node 3 to exit 1 9 3	Initial Priority Queue: 8 6 7 4 5 1 3 Max Priority: 8 Updated Priority Queue: 9 8 7 6 5 1 3 4

II. Given a binary tree, design an algorithm and write a program to check whether this tree is heap or not. Tree should satisfy following heap properties:
   a) Tree should be complete binary tree
   b) Every nodes value should be greater than or equal to its child node (incase of max heap)

Input Format:
The first line contains number of test cases, T.
For each test case, there will be three input lines.
First line of each test case will take size of array form of tree.
Second line of each test case contains n space-separated integers describing priorities of n elements in the form of array (level order traversal of binary tree).

Output Format:
Print "**Yes**" if given binaryb tree is heap, otherwise print "**No**".

Sample:

Input:	**Output:**
2 8 8 4 10 3 6 9 5 7 10 7 4 9 2 6 8 10 1 3 5	No Yes

III. Given a complete binary tree, design an algorithm and write a program to find Kth largest element in it. (Hint: Max heap)

Input:
First line of input will take size n of array form of tree.

Second line of input contains n space-separated integers describing elements of complete binary tree in the form of array.
Third line of input will take value of K

Output:
Output will be Kth largest element.

Sample:

Input:	**Output:**
8 4 5 1 6 7 3 8 9 3	K = 3, largest element : 7

Week 14:

Take all your friends' Facebook data (approx. 76 students) in the following format:
0, 1, 0, 1, 1, 0, 0, 0, 0, 1

Let's suppose this is record of first student in the class of 10 students. Above line shows that Roll No 1 is friend with roll no 2, 4, 5, and 10. Entry 0 between two nodes A and B shows that A and B are not Facebook friends while 1 shows A and B are friends.
Below table shows the record of a class with 4 students.

	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	0
4	1	1	0	0

There can be two cases for the above kind of representation:
a) Sparse Graph (What does this indicates? Read Graph representation.)
b) Dense Graph (What does this indicates? Read Graph representation.)

I. What do you think, which is best way to represent data of your class (Linked list representation or Adjacency list representation)? Represent the above data in both the format and find out which takes less space and by what amount?

II. Get two roll no as inputs from user and find out whether they are friends or not. If yes find their all mutual friends. Also find out those students who are not friends with both of them.