1. Write a program that takes the number of elements and the numbers themselves in an integer array that holds a maximum of 50 elements. The program then prompts for an integer to be searched in the array using a binary search. Make sure to include the following steps:
   1. Call selection (or bubble) sort routine before the binary search. Sort must be implemented in its own function and not in main.
   2. Implement a function for binary search. The sorted array is passed to the search routine which returns the location of the sought value, or -1 if the value is not in the array.
   3. Implement a function that computes and returns the mean of array.
   4. The program should output the size of the array entered, the array as entered by the user, the sorted array, the integer being searched for, the location of that integer in the sorted array (or an appropriate message if it is not in the array), and the mean of the data set.
2. Write a program to build a binary tree for the given elements (i.e., integers) and give traversal functions: inorder, preorder, postorder.
3. Finding the maximum and minimum element in BST.
4. Write a program for removing the duplicates in array using BST.
5. Input graph as a set of edges, with each edge as a pair ***v*** and ***w***, where ***v*** and ***w*** are integers.
   1. Write a program to read such an input and construct the corresponding adjacency matrix for both undirected and directed graphs.
   2. Implement a function that can print all the neighbours of a vertex ***v*** given the adjacency matrix of a graph.
   3. Implement a function for Depth First Search when the graph is represented as adjacency matrix.
6. Input graph as a set of edges, with each edge as a pair ***v*** and ***w***, where ***v*** and ***w*** are integers.
   1. Write a program to read such an input and construct the corresponding adjacency list data structures for both undirected and directed graphs.
   2. Implement a function that can print all the neighbours of a vertex ***v*** given the adjacency lists of a graph.
   3. Implement a function for Depth First Search when the graph is represented as adjacency list.