**Amrita Vishwa Vidyapeetham**

**Department of Computer Science and Engineering**

**Lab-11**

**--------------------------------------------------------------------------------------------------------------------------**

**Sub Code:** 19CSE212 **Sub Title:** Data Structures

**Roll No:** CB.EN.U4CSE19453 **Name:** R.ABHINAV

**Lab-**11  **Date:** 29-04-2021

**Suitable Data Structure is :** BST

**Code:**

class node():  
 def \_\_init\_\_(self, e):  
 self.element = e  
 self.leftchild = *None* self.rightchild = *None* self.root = *None*class BST():  
 def \_\_init\_\_(self):  
 self.sz = *0* self.root = *None* self.ht = *0* self.inorderNodes = []  
   
 def insert(self, v, e):  
 if self.root == *None*:  
 self.root = node(e)  
 else:  
 if v.element < e:  
 if v.rightchild == *None*:   
 v.rightchild = node(e)   
 else:  
 self.insert(v.rightchild, e)  
 else:  
 if v.leftchild == *None*:   
 v.leftchild = node(e)   
 else:   
 self.insert(v.leftchild, e)  
 self.sz += *1* def inorderTraverse(self, v):  
 if v == *None*:  
 return *None* self.inorderTraverse(v.leftchild)  
 self.inorderNodes.append(v.element)  
 self.inorderTraverse(v.rightchild)  
   
 def minmax(self):  
 self.inorderTraverse(self.root)  
  
 length = len(self.inorderNodes)  
 i = *0* j = length-*1* while i < j:  
 print(self.inorderNodes[i], end='->')  
 print(self.inorderNodes[j], end='->')  
 i += *1* j -= *1* if i == j:  
 print(self.inorderNodes[i], end='->')  
   
 def build(self, arr):  
 for i in arr:  
 self.insert(self.root, i)  
  
  
def main():  
 size = int(input())  
 arr = list(map(int, input().split()))  
 bst = BST()  
 bst.build(arr)  
 size = int(input())  
 for \_ in range(size):  
 command = input()  
 if command == 'MINMAX':  
 bst.minmax()  
 else:  
 bst.inorderTraverse(bst.root)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()