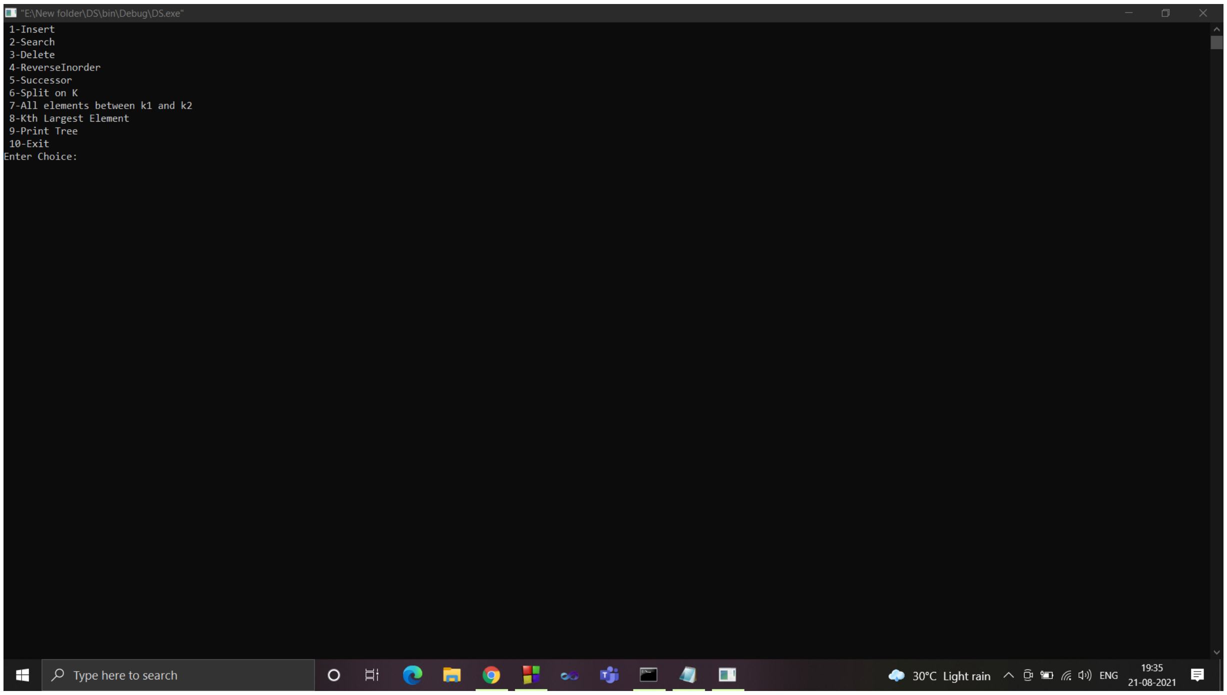
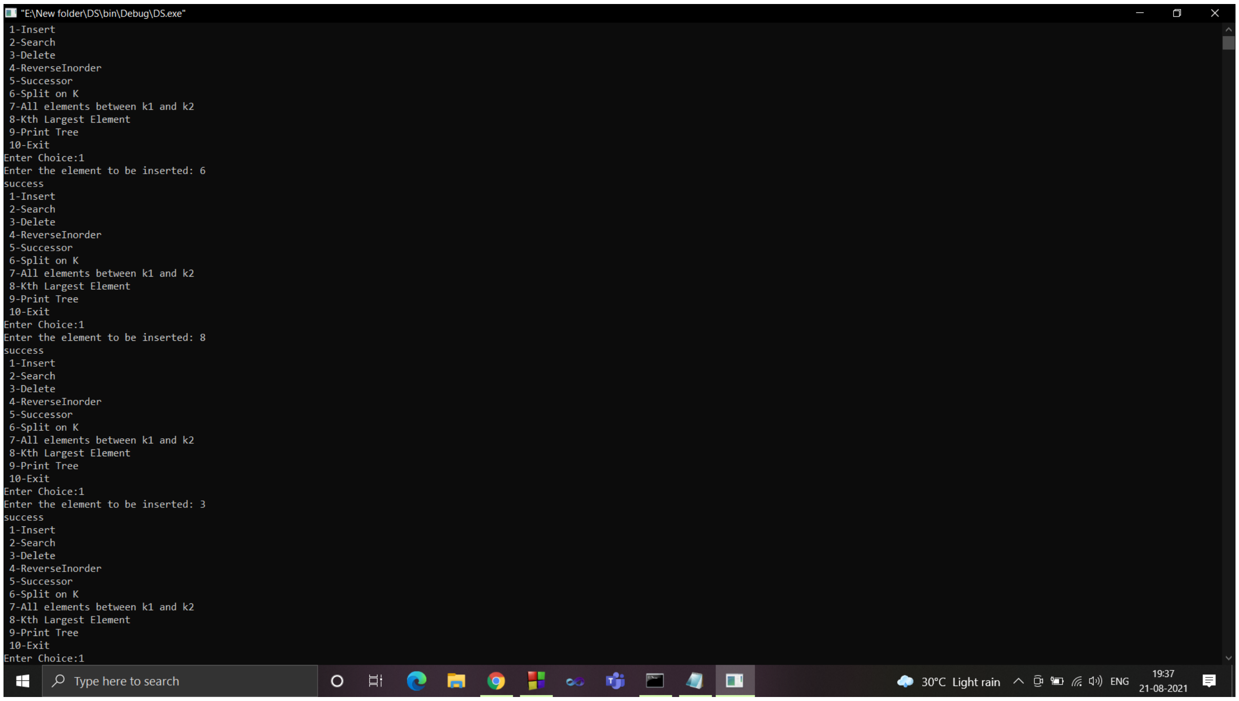
Logic of each functions:

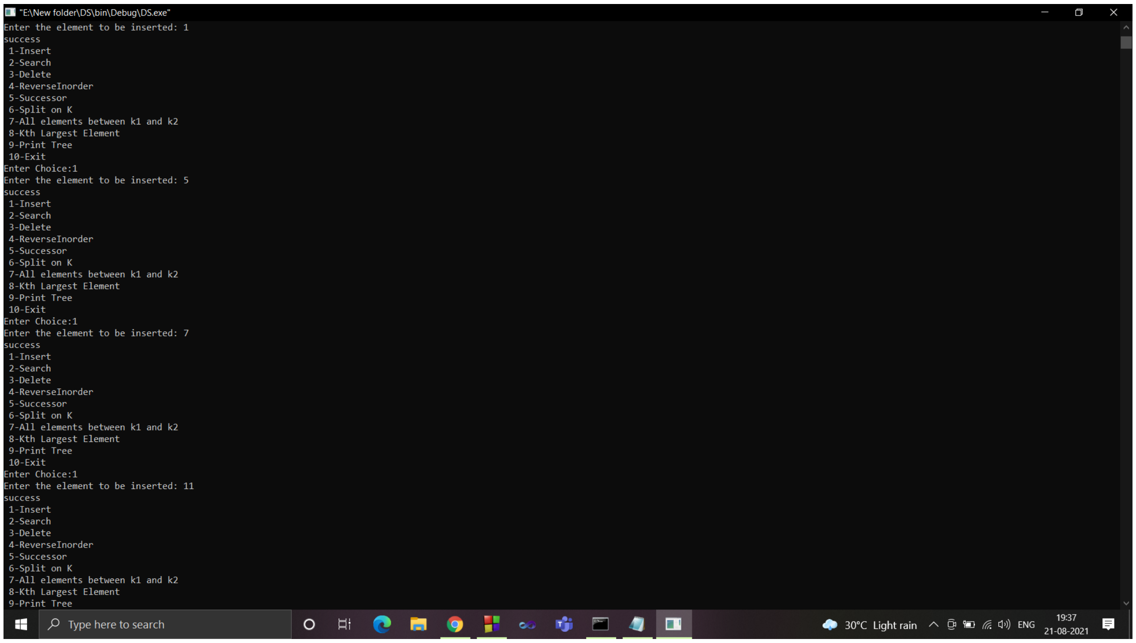
insertnode()
First I'm finding whether key to be inserted is already present in Binary Search tree or not.
If already present I'm just throwing an exception.
Else I'm traversing the tree to find the position of the node to be iserted.
Once the position is found the node is created and inserted and the parameters of the node is set accordingly.



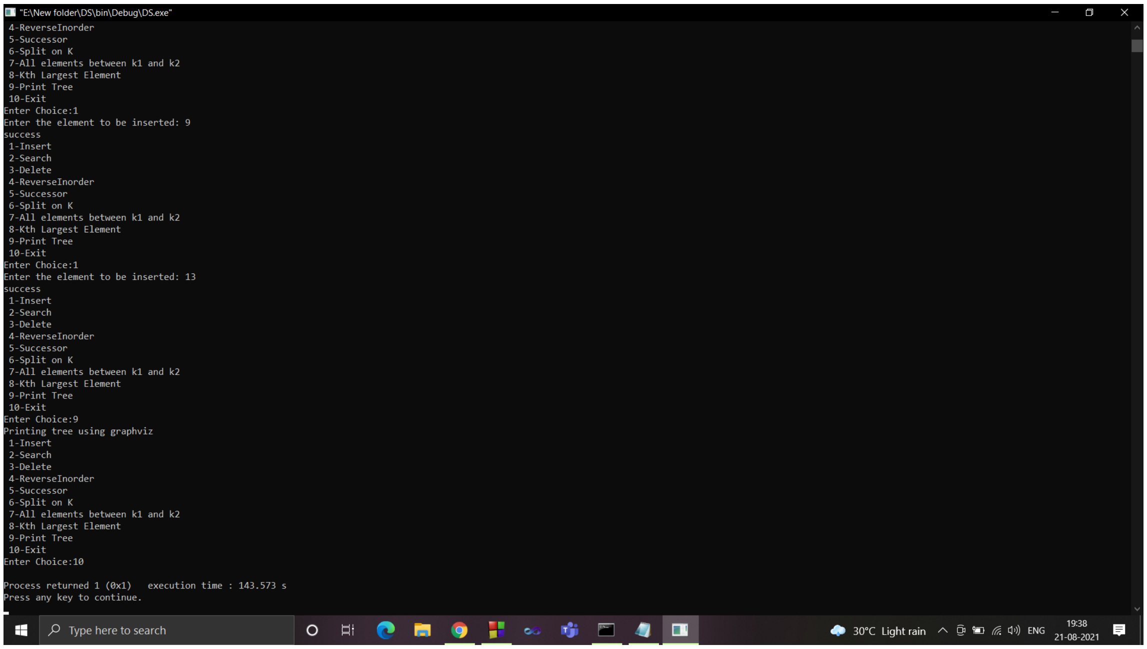
```
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:
```



```
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 6
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 8
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 3
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
```



```
Enter the element to be inserted: 1
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 5
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 7
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 11
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
```



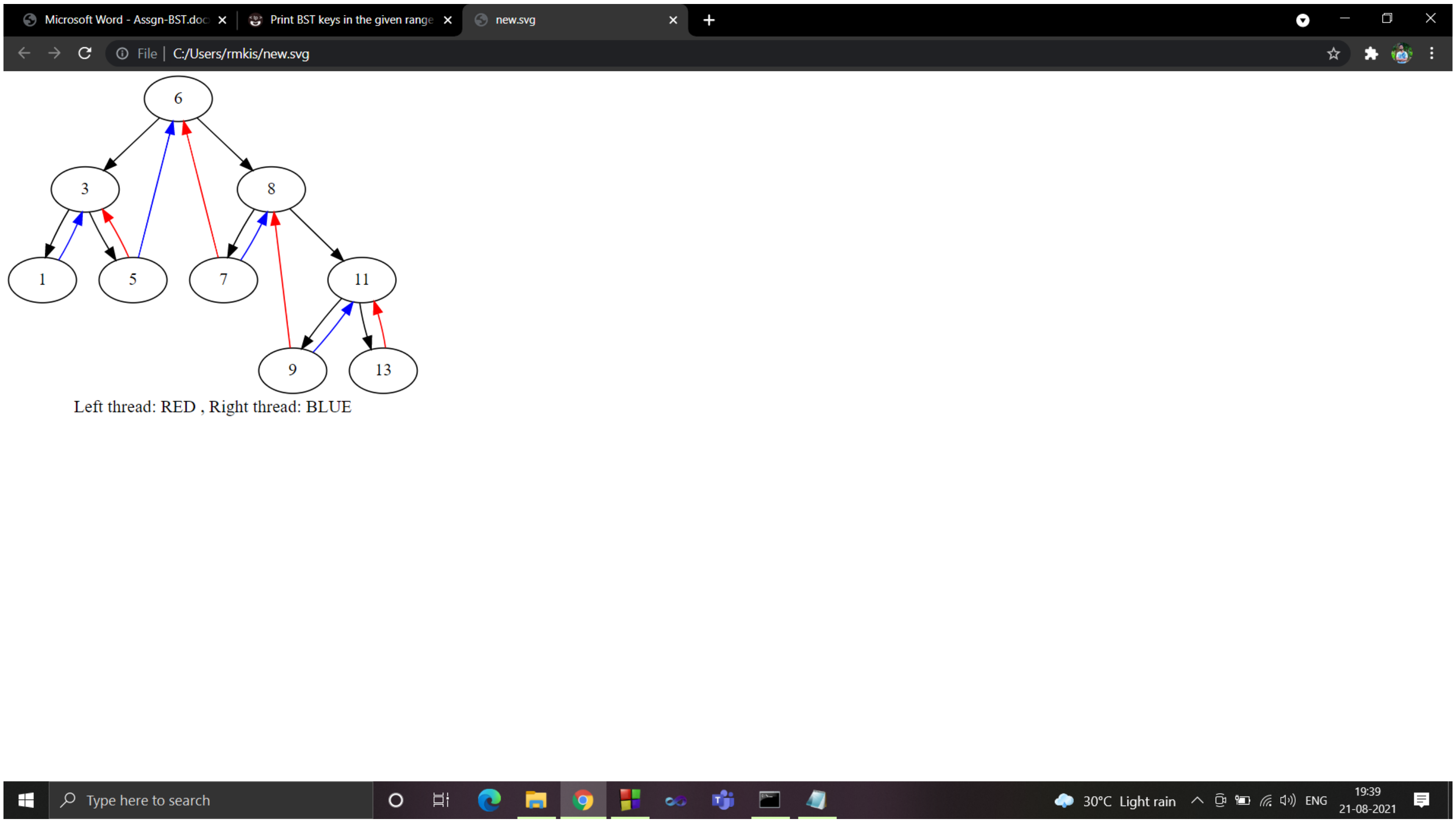
```
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 9
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:1
Enter the element to be inserted: 13
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:9
Printing tree using graphviz
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:10

Process returned 1 (0x1)   execution time : 143.573 s
Press any key to continue.
```

All the elements have been inserted

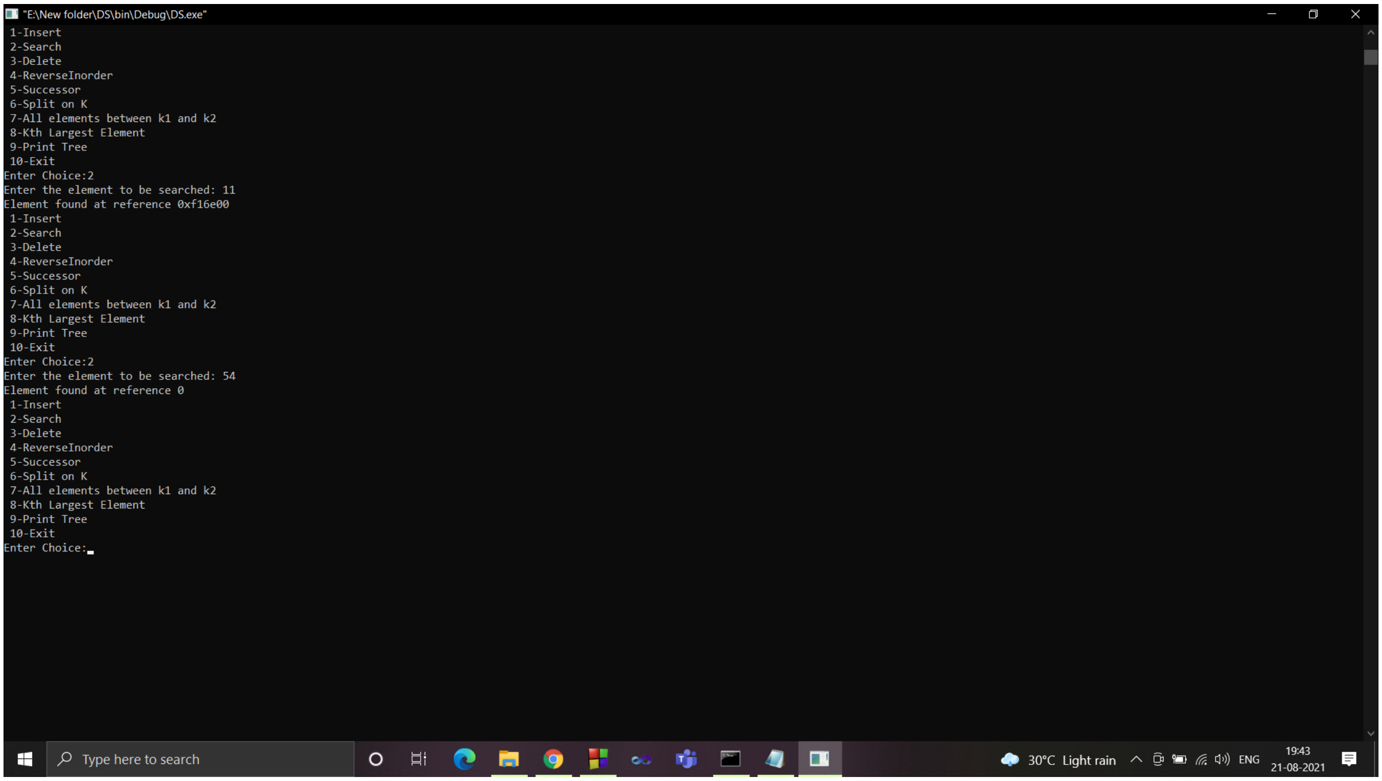
The elements are inserted in order : 6,8,3,1,5,7,11,9,13.

The visualization of tree after insertion is shown below.


Microsoft Word - Assgn-BST.doc
Print BST keys in the given range
new.svg
File | C:/Users/rmkis/new.svg
6
3
8
1
5
7
11
9
13
Left thread: RED , Right thread: BLUE
Type here to search
30°C Light rain
ENG
19:39
21-08-2021

searchkey()
Here I'm traversing through left or right subtree to find whether the node is present in the tree or not after finding key != root->data.
It returns boolean value so that it can be used in both insert and selete functions.

searchfun()
This function is same as searchkey functions where it differs is that it returns reference of the node where the key is present.



```
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:2
Enter the element to be searched: 11
Element found at reference 0xf16e00
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:2
Enter the element to be searched: 54
Element found at reference 0
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:_
```

deletenode()

In this function first I'm checking whether the element is present is BST or not.

If present then there occurs three cases in which the element has to be deleted.

CASE 1:

If the element to be deleted is a leaf

Then the element can be directly deleted and the parent of the element element becomes leaf and acquires properties of leaf

CASE 2:

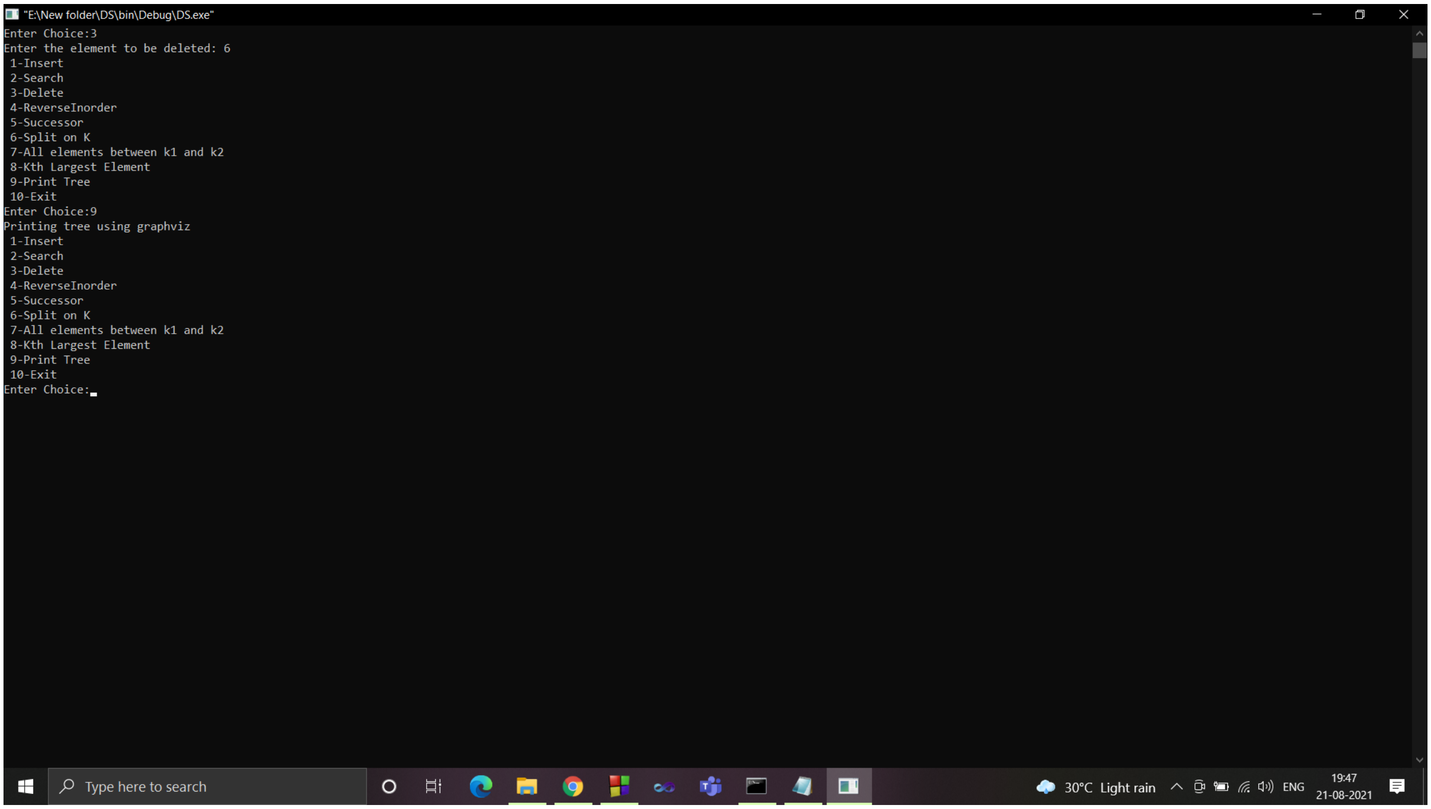
If the element to be deleted has one children.

Then we make the parent node to point the node which deleted node was pointing.

CASE 3:

If the element to be deleted has two children

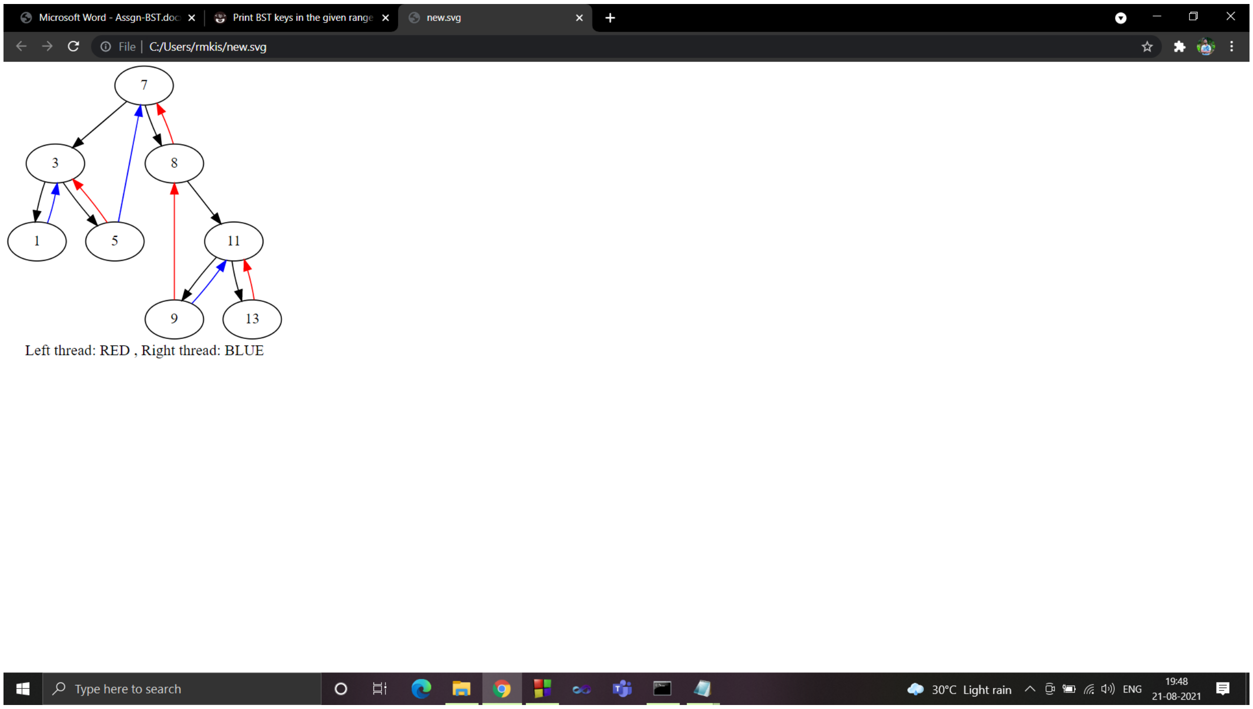
Then the deleted node must be replaced with inorder successor of the element.



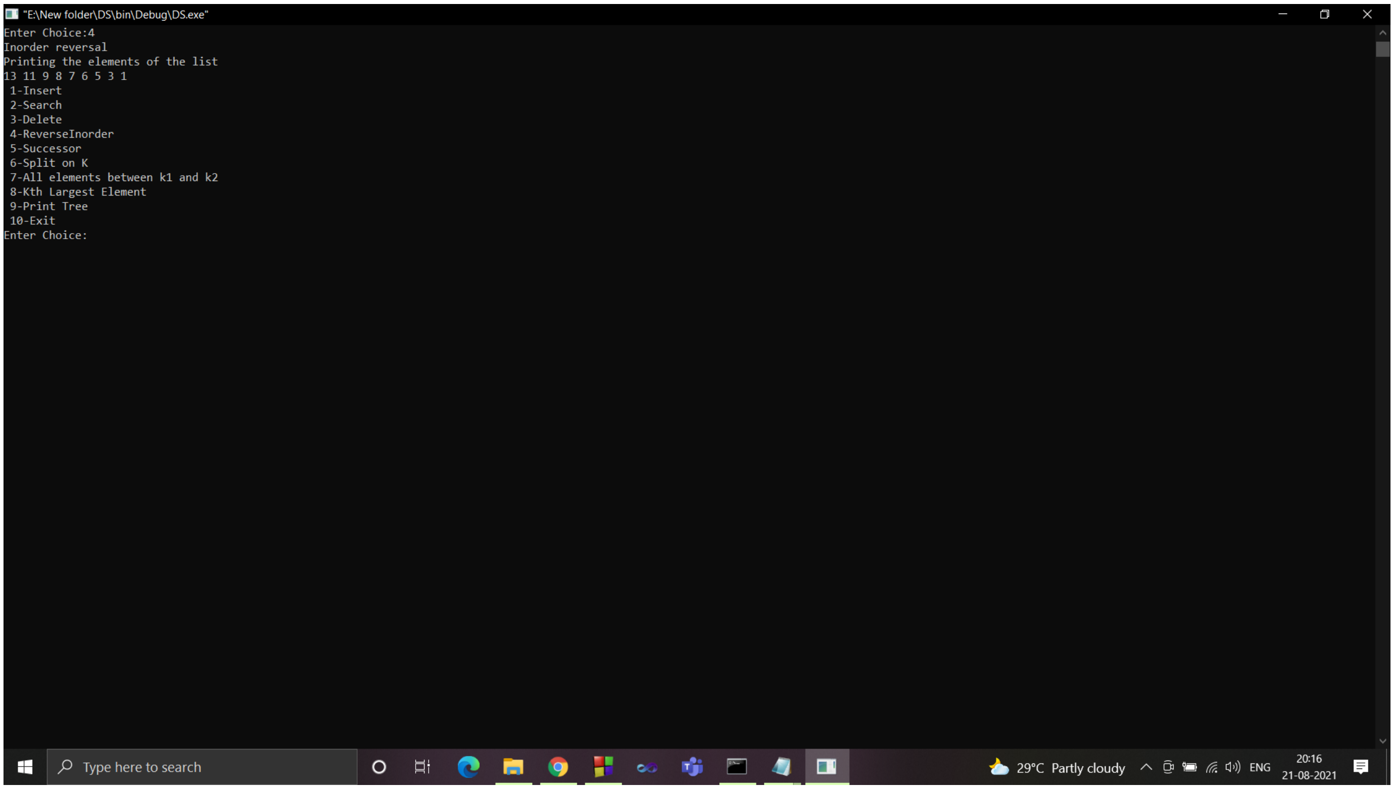
```
Enter Choice:3
Enter the element to be deleted: 6
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:9
Printing tree using graphviz
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:_
```

Element 6 is deleted from the original tree

The visualization of tree after deletion is shown below.


Microsoft Word - Assgn-BST.doc
Print BST keys in the given range
new.svg
File | C:/Users/rmkis/new.svg
7
3
8
1
5
11
9
13
Left thread: RED , Right thread: BLUE
Type here to search
30°C Light rain
ENG
19:48
21-08-2021

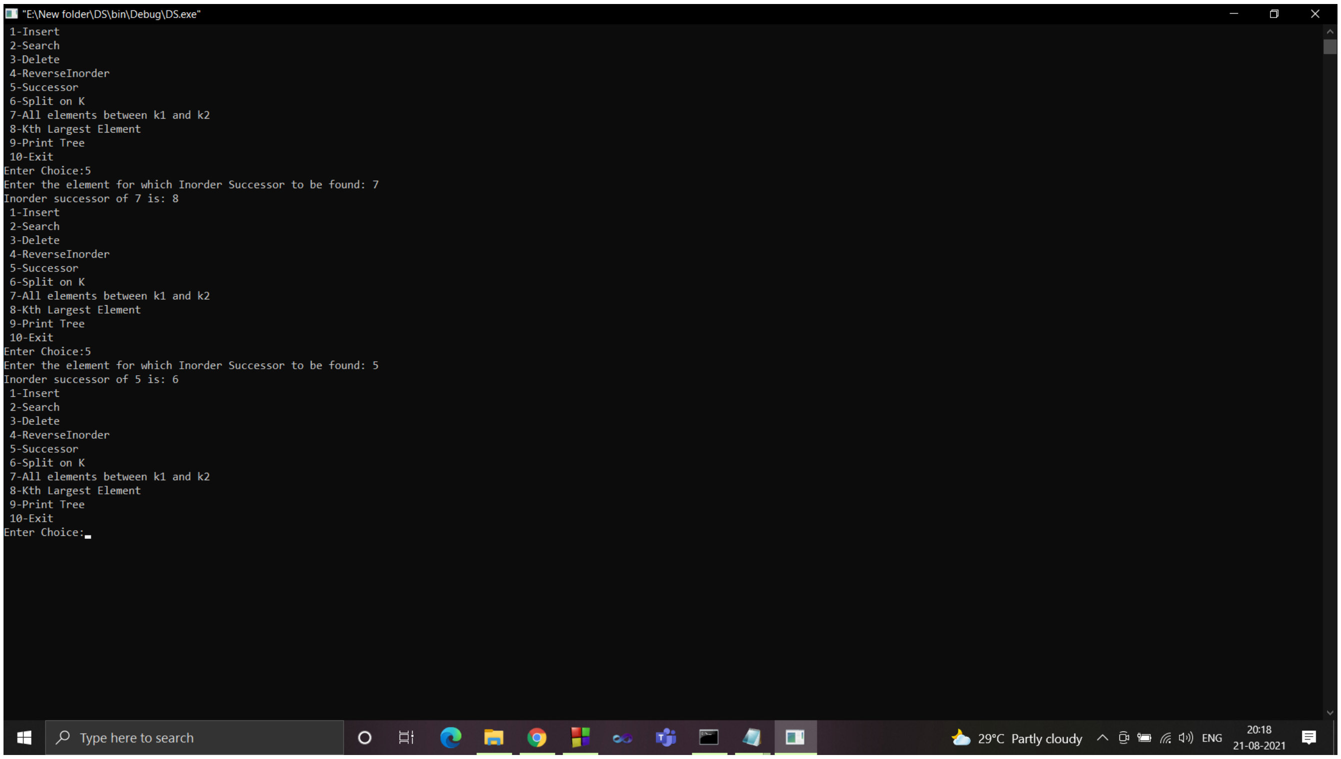
reverseinorder()
Here we are traversing to to extreme right to find the max element.
Then we are finding inorder predecessor of each element and adding them to linked list.



```
Enter Choice:4
Inorder reversal
Printing the elements of the list
13 11 9 8 7 6 5 3 1
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:
```

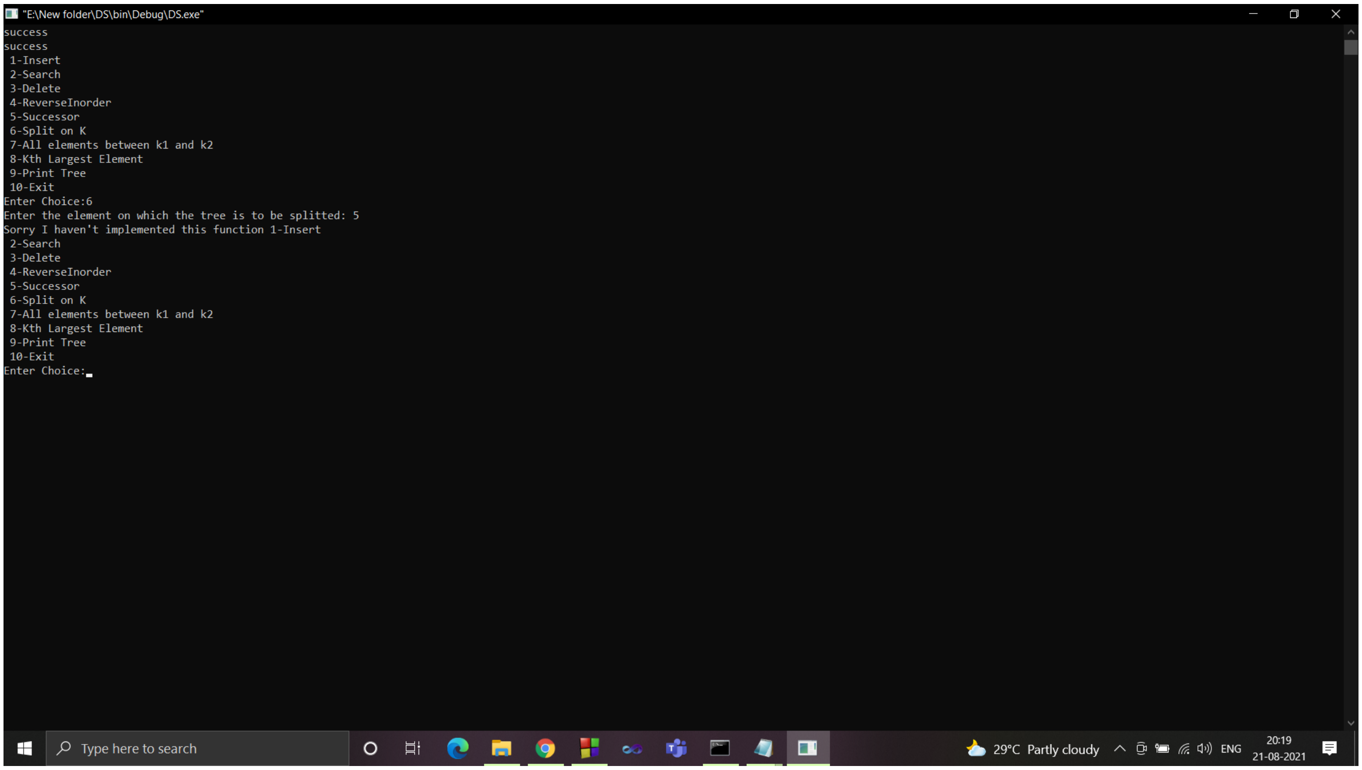
inS()
inS() function finds the inordr successor with the help of threading

inP()
inP() function finds the inorder predecessor with the help of threading



```
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:5
Enter the element for which Inorder Successor to be found: 7
Inorder successor of 7 is: 8
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:5
Enter the element for which Inorder Successor to be found: 5
Inorder successor of 5 is: 6
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:_
```

I haven't implemented split function.

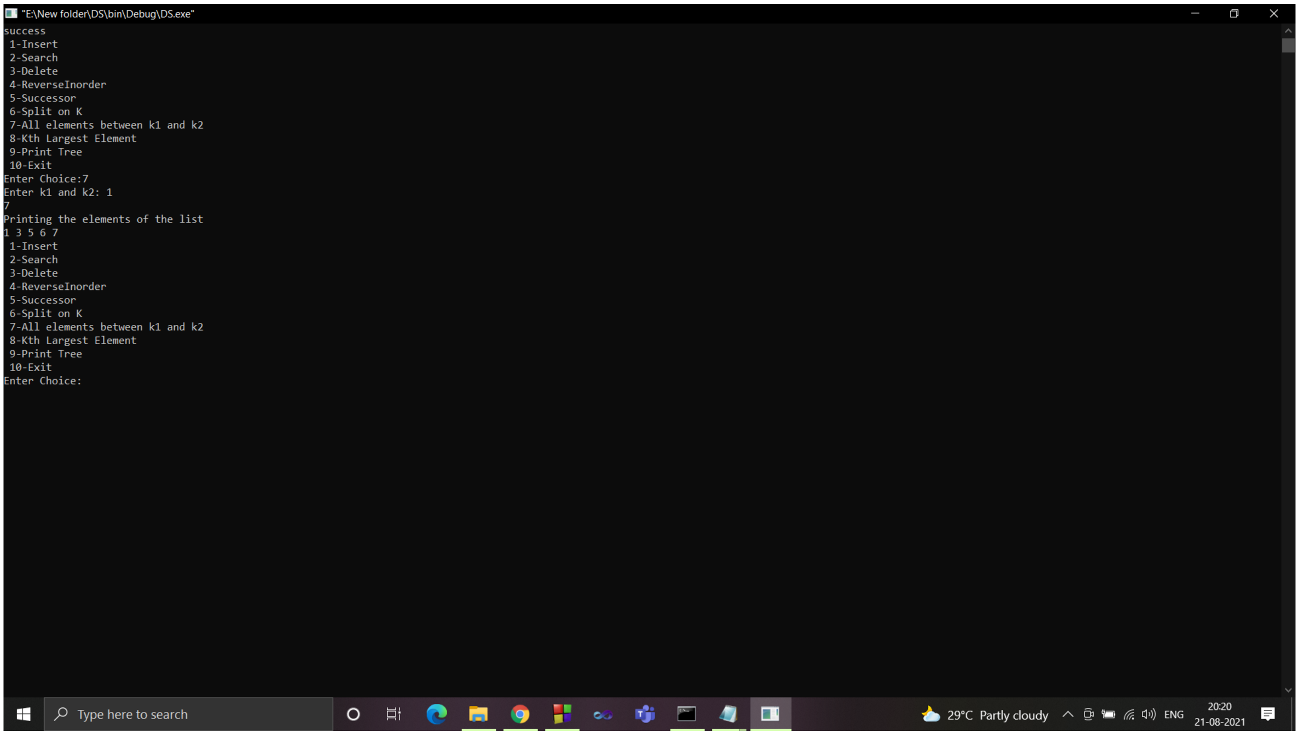


```
success
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:6
Enter the element on which the tree is to be splitted: 5
Sorry I haven't implemented this function 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:_
```

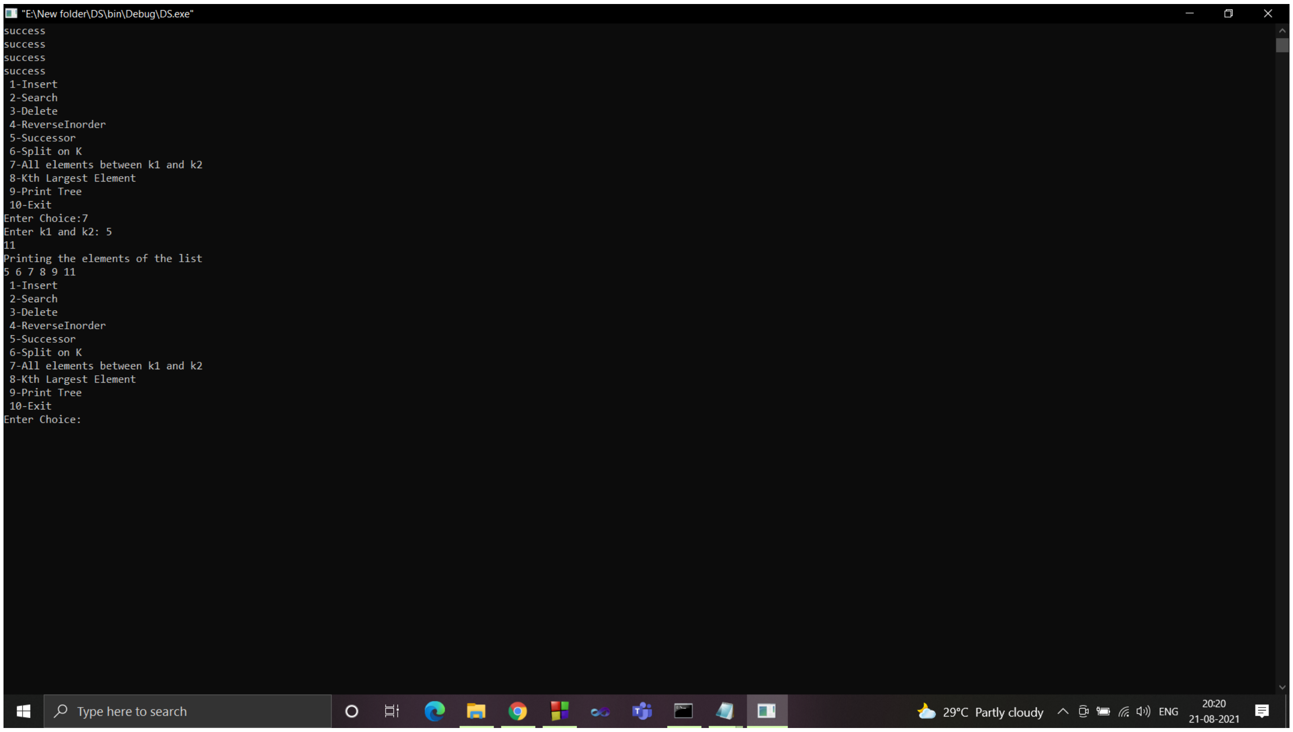
allelementsbetween()
Here we are finding whether the root is greater than k1 if so we traverse right subtree
If not we traverse from left subtree using recursion.

Once the closest value greater than k1 is reached, the element is added to linked list
and we keep on traversing to find all elements so the running time is O(K)
where k is number of elements between k1 and k2.

The elements of the list is printed .

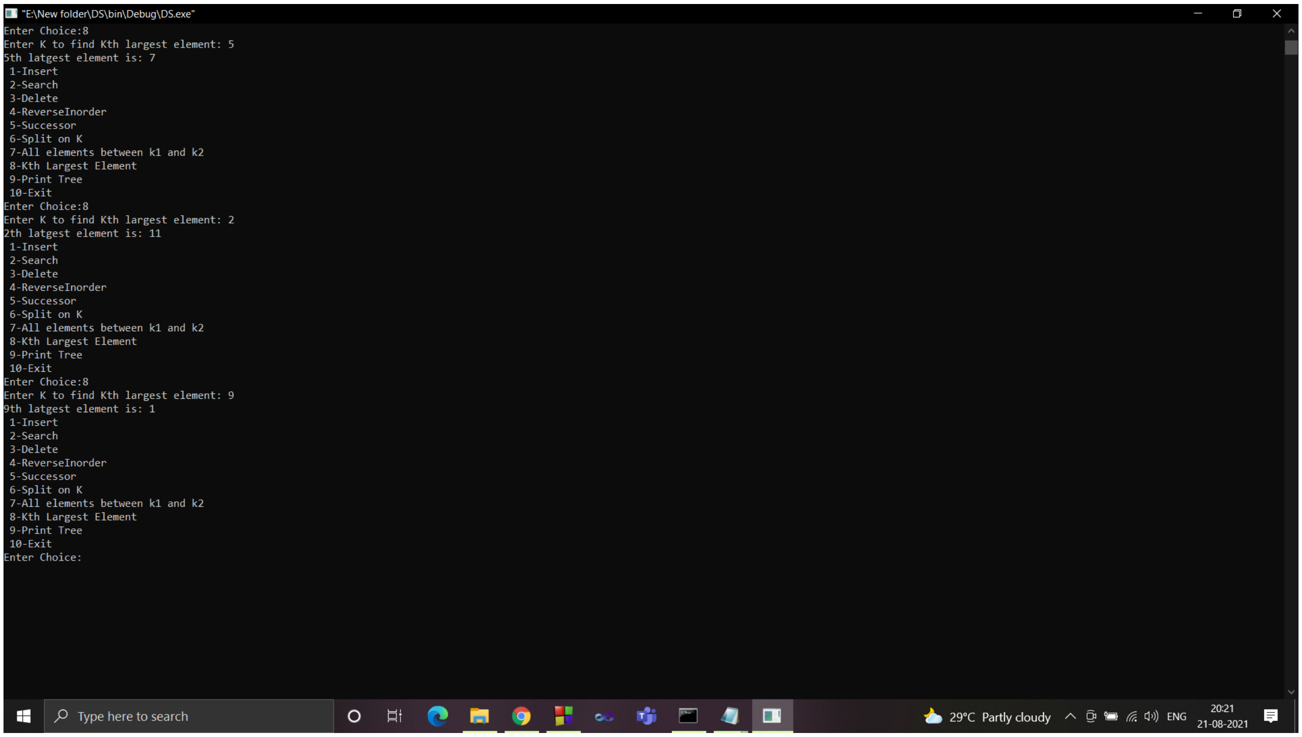


```
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:7
Enter k1 and k2: 1
7
Printing the elements of the list
1 3 5 6 7
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:
```



```
success
success
success
success
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:7
Enter k1 and k2: 5
11
Printing the elements of the list
5 6 7 8 9 11
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:
```

Kthlargest()
Here while inserting the node we are maintaining a count variable as rcount
It keeps count of each element as kth largest while inserting.
And in this function we just traverse to find the rcount which is equal to K.



```
Enter Choice:8
Enter K to find Kth largest element: 5
5th latgest element is: 7
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:8
Enter K to find Kth largest element: 2
2th latgest element is: 11
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:8
Enter K to find Kth largest element: 9
9th latgest element is: 1
 1-Insert
 2-Search
 3-Delete
 4-ReverseInorder
 5-Successor
 6-Split on K
 7-All elements between k1 and k2
 8-Kth Largest Element
 9-Print Tree
 10-Exit
Enter Choice:
```

createlnode()
creates node of like linked list.

main()
In main function we create objects of each class and call functions accordingly.
And free out the dynamic space allotted.

To use Graphviz Install it on your system
change the system environment variables to location of graphviz
Open command line and execute the command
dot.exe-Tsvg inputfile -o outputfile