**Design:** Code and output

**Code:**

class Tree:

    def \_\_init\_\_(self, key):

        self.left = None

        self.right = None

        self.val = key

def searchDLS(root, h, t):

    h -= 1

    if(h == -1):

        return 0

    if(root == None):

        return 0

    if(t == root.val):

        return root.val

    l = searchDLS(root.left, h, t)

    if (l != 0):

        return l

    r = searchDLS(root.right, h, t)

    if (r != 0):

        return r

    if(l == 0):

        return r

    else:

        return l

root = Tree(1)

root.left = Tree(2)

root.right = Tree(3)

root.left.left = Tree(4)

root.left.right = Tree(5)

root.right.left = Tree(6)

root.right.right = Tree(7)

h = 2

target = int(input("Enter target element : "))

x = searchDLS(root, h, target)

if(x == 0):

    x = -1

    print("Target element not found")

else:

    print("Target element found :", target)

Text

Description automatically generated

**Design:** Code and output

**Code:**

from queue import PriorityQueue

v = 14

graph = [[] for i in range(v)]

def bestfs(actual\_Src, target, n):

    visited = [False]\*n

    pq = PriorityQueue()

    pq.put((0, actual\_Src))

    visited[actual\_Src] = True

    while pq.empty() == False:

        u = pq.get ()[1]

        print(u, end = " ")

        if u == target:

            break

        for v, c in graph[u]:

            if visited [v] == False :

                visited[v] = True

                pq.put ((c, v))

    print()

def addedge (x, y, cost):

    graph[x].append ( (y, cost))

    graph[y]. append ( (x, cost))

addedge(0, 1, 3)

addedge(0, 2, 6)

addedge(0, 3, 5)

addedge(1, 4, 9)

addedge(1, 5, 8)

addedge(2, 6, 12)

addedge(2, 7, 14)

addedge(3, 8, 7)

addedge(8, 9, 5)

addedge(8, 10, 6)

addedge(9, 11, 1)

addedge(9, 12, 10)

addedge(9, 13, 2)

source= 0

target = 10

bestfs(source, target, v)

Text

Description automatically generated