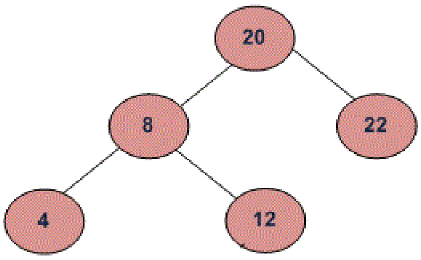
Print BST keys in the given range

Given two values k1 and k2 (where k1 < k2) and a root pointer to a Binary Search Tree. Print all the keys of tree in range k1 to k2. i.e. print all x such that k1<=x<=k2 and x is a key of given BST. Print all the keys in increasing order.

For example, if k1 = 10 and k2 = 22, then your function should print 12, 20 and 22.



Thanks to bhasker for suggesting the following solution.

Algorithm:
1) If value of root's key is greater than k1, then recursively call in left subtree.
2) If value of root's key is in range, then print the root's key.
3) If value of root's key is smaller than k2, then recursively call in right subtree.

Implementation:

C

```
#include<stdio.h>

/* A tree node structure */
struct node
{
  int data;
  struct node *left;
  struct node *right;
};

/* The functions prints all the keys which in the given range [k1..k2].
    The function assumes than k1 < k2 */
void Print(struct node *root, int k1, int k2)
{
   /* base case */
   if ( NULL == root )
      return;

   /* Since the desired o/p is sorted, recurse for left subtree first
      If root->data is greater than k1, then only we can get o/p keys
      in left subtree */
   if ( k1 < root->data )
     Print(root->left, k1, k2);

   /* if root's data lies in range, then prints root's data */
   if ( k1 <= root->data && k2 >= root->data )
     printf("%d ", root->data );

  /* If root->data is smaller than k2, then only we can get o/p keys
      in right subtree */
   if ( k2 > root->data )
     Print(root->right, k1, k2);
}

/* Utility function to create a new Binary Tree node */
struct node* newNode(int data)
{
  struct node *temp = new struct node;
  temp->data = data;
  temp->left = NULL;
  temp->right = NULL;

  return temp;
}

/* Driver function to test above functions */
int main()
{
  struct node *root = new struct node;
  int k1 = 10, k2 = 25;

  /* Constructing tree given in the above figure */
  root = newNode(20);
  root->left = newNode(8);
  root->right = newNode(22);
  root->left->left = newNode(4);
  root->left->right = newNode(12);

  Print(root, k1, k2);

  getchar();
  return 0;
}
```

Java

```java
// Java program to print BST in given range

// A binary tree node
class Node {

    int data;
    Node left, right;

    Node(int d) {
        data = d;
        left = right = null;
    }
}

class BinaryTree {

    static Node root;

    /* The functions prints all the keys which in the given range [k1..k2].
     The function assumes than k1 < k2 */
    void Print(Node node, int k1, int k2) {

        /* base case */
        if (node == null) {
            return;
        }

        /* Since the desired o/p is sorted, recurse for left subtree first
         If root->data is greater than k1, then only we can get o/p keys
         in left subtree */
        if (k1 < node.data) {
            Print(node.left, k1, k2);
        }

        /* if root's data lies in range, then prints root's data */
        if (k1 <= node.data && k2 >= node.data) {
            System.out.print(node.data + " ");
        }

        /* If root->data is smaller than k2, then only we can get o/p keys
         in right subtree */
        if (k2 > node.data) {
            Print(node.right, k1, k2);
        }
    }


    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();
        int k1 = 10, k2 = 25;
        tree.root = new Node(20);
        tree.root.left = new Node(8);
        tree.root.right = new Node(22);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(12);

        tree.Print(root, k1, k2);
    }
}

// This code has been contributed by Mayank Jaiswal
```

Output:

```
12                                  20                                  22
```

Time Complexity: O(n) where n is the total number of keys in tree.