```
#include<stdio.h>
#include<stdlib.h>
# define MAX 100
# define INFINITY_MAX 10000
struct Node
{
   int data;
   struct Node* left,  *right;
};

/* This function traverse the  binary search  tree and
  stores its nodes pointers in array  arr[] */
void storeBSTNodes(struct Node* root, struct Node* arr[], int *i)
{
   // Base case
   if (root==NULL)
      return;
    storeBSTNodes(root->left, arr,i);
   arr[(*i)++] = root;
   storeBSTNodes(root->right, arr,i);
}

/* Recursive function to construct balanced binary search tree from a sorted array */
struct Node* ArraytoBalancedBST( struct Node* arr[], int start, int end)
{
  struct  Node *root;
   if (start > end)
      return NULL;

   /* Get the middle element and make it root */
   int mid = (start + end)/2;
   root = arr[mid];

   /* Using index in Inorder traversal, construct
     left and right subtress */
   root->left  = ArraytoBalancedBST(arr, start, mid-1);
   root->right = ArraytoBalancedBST(arr, mid+1, end);

   return root;
}

// This functions converts an unbalanced BST to
// a balanced BST
struct Node* buildTree(struct Node* root)
{
   Node *arr[MAX];
   int n=0;
   storeBSTNodes(root,arr,&n);
```

```
   return (ArraytoBalancedBST(arr,0,n-1));

}

// Utility function to create a new node
struct Node* newNode(int data)
{
   struct  Node* node = (struct  Node*)malloc(sizeof(struct Node));
   node->data = data;
   node->left = node->right = NULL;
   return (node);
}

/* Function to do preorder traversal of tree */
void preOrder(struct Node* node)
{
   if (node == NULL)
      return;

   printf("%d ", node->data);
   preOrder(node->left);
   preOrder(node->right);
}

// Driver program
int main()
{
   struct Node* root = newNode(10);
   root->left = newNode(8);
   root->left->left = newNode(7);
   root->left->left->left = newNode(6);
   root->left->left->left->left = newNode(5);
   printf("Preorder traversal of  BST is : \n");
   preOrder(root);
   root = buildTree(root);
   printf("\n Preorder traversal of balanced BST is : \n");
   preOrder(root);

   return 0;
}
```

Output:
Preorder traversal of BST is:
10 8 7 6 5
Preorder traversal of balanced BST is:
7 5 6 8 10

Time Complexity: O(n)

Space Complexity: O(n)