KING AND THE CRISIS

Code Explanation :

To construct a binary tree, we require the following struct and the following ‘Insert’ function as discussed in the class :

```
typedef struct tree
{
long int key;
struct tree *left;
struct tree *right;
} treenode;
```

‘Insert’ function :

```
treenode *insert(treenode *t, long int e)
{
if (t == NULL)
{
t = makenode(e);
}

else if (e < t->key)
{
t->left = insert(t->left, e);
}

else if (e > t->key)
{
t->right = insert(t->right, e);
}
return t;
}
```

In the main function, we pass every new element entered by the user to the insert function. Thus, we create the Binary Search Tree.

Now, to add to every element in the tree the sum of all the values in the tree smaller than it, we essentially perform an ‘inorder’ traversal of the created tree. We know, that the inorder traversal implies going in the left child- parent- right child order for every subtree recursively. Thus, we call the function on the sum of all the values in the left subtree, add it to the ‘parent’ and finally add the parent’s value to the right child.We do this method for every subtree recursively.We change the value of every node’s key in the same function itself.
Long int data type has been chosen in alignment with the constraints given in the question.

Code :

```
long int update(treenode *t, long int s)
{
if (t == NULL)
return s;
s = update(t->left, s);
s += t->key;
t->key = s;
s = update(t->right, s);
return s;
}
```

This code updates the values of all the nodes in the treee accordingly.

Now, to perform a level order traversal of the created tree, we apply the algorithm discussed in the class. We enque the root node, then while dequeing it, we enque its left child and right child. If wither doesn’t exist, we do not enque.We now deque the left chid and enque its children in the same manner. Thus, for every deque, we enque the node’s children. We do this till the queue becomes empty.

Time Complexity :

Time complexity of insertion : O(n^2)

Time complexity of updation : O(n)
Time complexity of level order traversal : O(n)