```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>  // For INT_MAX

// Structure for BST Node
struct Node {
      int data;
      struct Node *left, *right;
};

// Function to create a new node
struct Node* newNode(int data) {
      struct Node* node;

      node = (struct Node*)malloc(sizeof(struct Node));
      node->data = data;
      node->left = NULL;
      node->right = NULL;

      return node;
}

// Function to insert a node into BST
struct Node* insert(struct Node* root, int key) {
      // If tree is empty, return a new node
      if (root == NULL)
      return newNode(key);

      // Recur down the tree
      if (key < root->data)
      root->left = insert(root->left, key);
      else
      root->right = insert(root->right, key);

      return root;
}

// Function to check if BST contains a dead end
int checkDeadEnd(struct Node* root, int min, int max) {
      // Base condition: empty subtree
      if (root == NULL)
      return 0;

      // If min == max, no further nodes can be inserted -> dead end
      if (min == max)
      return 1;
```

```
	// Recur for left and right subtrees
	return checkDeadEnd(root->left, min, root->data - 1) ||
		checkDeadEnd(root->right, root->data + 1, max);
}

// Wrapper function to check for dead end
int containsDeadEnd(struct Node* root) {
	return checkDeadEnd(root, 1, INT_MAX);
}

// Driver Code
int main() {
	struct Node* root1;
	struct Node* root2;

	root1 = NULL;
	root2 = NULL;

	// ---------- Test Case 1: BST with a dead end ----------
root1 = NULL;
root1 = insert(root1, 8);
root1 = insert(root1, 5);
root1 = insert(root1, 2);
root1 = insert(root1, 1);  // Critical for dead end
root1 = insert(root1, 3);
root1 = insert(root1, 7);
root1 = insert(root1, 11);


	printf("Test Case 1:\n");
	if (containsDeadEnd(root1))
	printf("BST contains a dead end\n");
	else
	printf("BST does not contain a dead end\n");

	// ---------- Test Case 2: BST without a dead end ----------
	// Tree: 10, 5, 1, 7, 40, 50
	root2 = insert(root2, 10);
	root2 = insert(root2, 5);
	root2 = insert(root2, 1);
	root2 = insert(root2, 7);
	root2 = insert(root2, 40);
	root2 = insert(root2, 50);

	printf("\nTest Case 2:\n");
```

```
      if (containsDeadEnd(root2))
      printf("BST contains a dead end\n");
      else
      printf("BST does not contain a dead end\n");

      return 0;
}

## OUTPUT :

Test Case 1:
BST does not contain a dead end

Test Case 2:
BST does not contain a dead end

--------------------------------
Process exited after 0.04176 seconds with return value 0
```