Problem 7.c

```c
/**
 * Implementation for recursive traversal of a Simple Binary Tree
 * Author: Mithusayel Murmu
 */

#include <stdio.h>
#include <stdlib.h>

/* Typedefs for BST and BSTNode */
typedef struct _BSTNode BSTNode;
typedef struct _BSTree BSTree;

struct _BSTNode { int data; BSTNode *left, *right; };
struct _BSTree { int size; BSTNode *root; };

BSTree * bst_create() {
    BSTree *bst = (BSTree *) malloc(sizeof(BSTree));
    bst->size = 0; bst->root = NULL;
    return bst;
}

/**
 * Recursive definition for node insertion in BST
 * @node:   Pointer to BST's node pointer (typically the root)
 * @data:   The data/value to insert in the tree
 */
void bst_insert_node(BSTNode **node, int data) {
    if (*node == NULL) {
        BSTNode *_node = (BSTNode *) malloc(sizeof(BSTNode));
        _node->left = _node->right = NULL;
        _node->data = data;
        *node = _node;

        return;
    }

    // Redirect left
    if (data < (*node)->data)
        bst_insert_node(&(*node)->left, data);
    else
        bst_insert_node(&(*node)->right, data);
}

/**
 * Recursive definition for inorder traversal of the BST
 * @node:       Pointer to the node to start traversing from (root)
 * @callback:   Pointer to the callback function to process results
 */
void bst_traverse_in(BSTNode *node, void (*callback)(int)) {
    if (node == NULL) return;

    bst_traverse_in(node->left, callback);
    callback(node->data);
    bst_traverse_in(node->right, callback);
}

void bst_destroy_nodes(BSTNode **node) {
    if (*node == NULL) return;

    bst_destroy_nodes(&(*node)->left);
    bst_destroy_nodes(&(*node)->right);
    free(*node); *node = NULL;
}

void bst_insert(BSTree *bst, int data) {
    bst_insert_node(&bst->root, data);
    bst->size++;
}

void bst_destroy(BSTree *bst) {
    if (bst == NULL) return;
    bst_destroy_nodes(&bst->root); free(bst);
}

void print_utility(int data) { printf("%d ", data); }

#define scand(n) scanf("%d", &n)

int main(int argc, char const *argv[]) {
```

```
80      int N, num;
81      BSTree *bst = bst_create();
82
83      printf("Number of elements to be inserted: ");
84      scand(N);
85      printf("Enter %d space separated integers: ", N);
86
87      while (N--) {
88          scand(num);
89          bst_insert(bst, num);
90      }
91
92      printf("\nPrinting while traversal:\n");
93      bst_traverse_in(bst->root, print_utility);
94      printf("\n"); bst_destroy(bst);
95
96      return 0;
97  }
98
```