1. main.c

```
#include "header.h"


int main() {
   BST tree;
   initBST(&tree);


   int choice;
   char month[10];


   while (1) {
      printf("\nMenu:\n");
      printf("1. Insert a node\n");
      printf("2. Remove a node\n");
      printf("3. Traverse the tree\n");
      printf("4. Destroy the tree\n");
      printf("5. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);


      switch (choice) {
         case 1:
            printf("Enter month to insert: ");
            scanf("%s", month);
            insertNode(&tree, month);
            break;
         case 2:
            printf("Enter month to remove: ");
            scanf("%s", month);
            removeNode(&tree, month);
            break;
            case 3:
                printf("In-order traversal of the tree:\n");
                traverse(tree.root);
                break;
            case 4:
                destroyTree(tree.root);
                tree.root = NULL;
                printf("Tree destroyed.\n");
                break;
            case 5:
                destroyTree(tree.root);
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

2.header.h

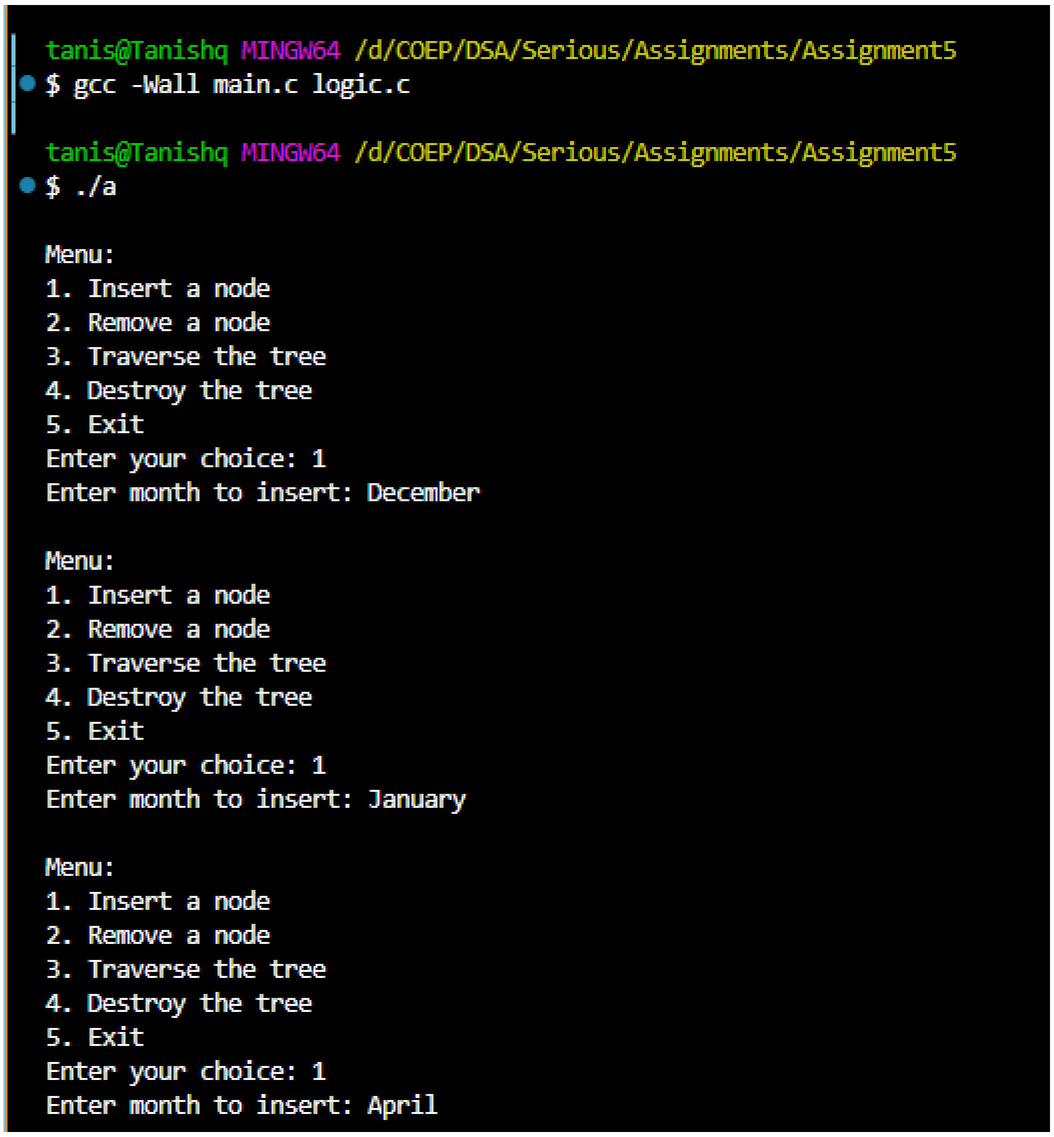
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Node {
    char month[10];
    struct Node* left;
    struct Node* right;
    struct Node* parent;
} Node;

typedef struct {
    Node* root;
} BST;

void initBST(BST* tree);
Node* createNode(char* month);
void insertNode(BST* tree, char* month);
Node* searchNode(BST* tree, char* month);
void removeNode(BST* tree, char* month);
void traverse(Node* node);
void destroyTree(Node* node);
```

3.logic.c

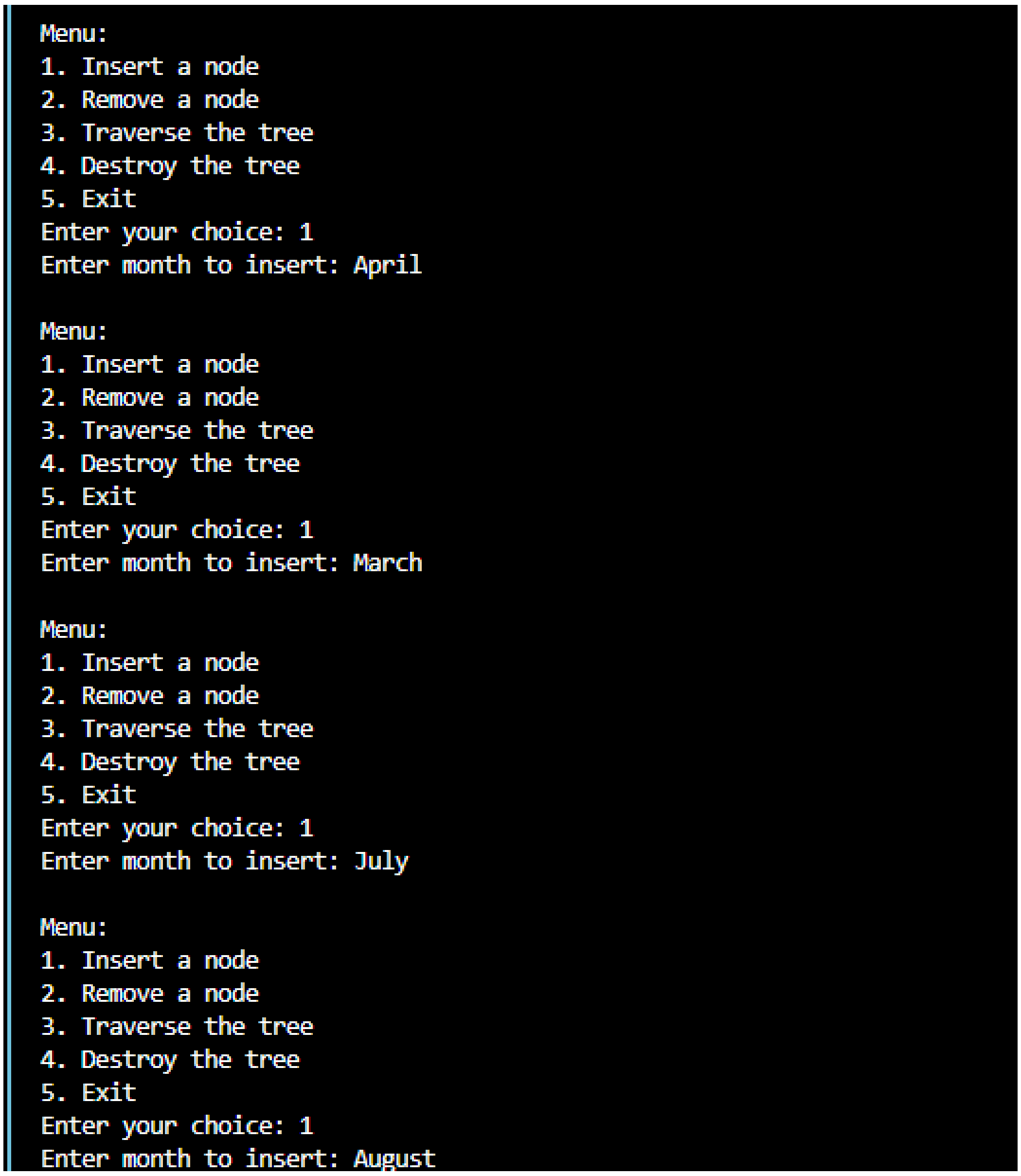
```
#include "header.h"

void initBST(BST* tree){
   tree->root = NULL;
}

Node* createNode(char* month){
   Node* newNode = (Node*)malloc(sizeof(Node));
   strcpy(newNode->month, month);
   newNode->left = newNode->right = newNode->parent = NULL;
   return newNode;
}

void insertNode(BST* tree, char* month) {
   Node* newNode = createNode(month);
   Node* current = tree->root;
   Node* parent = NULL;

   // Traverse the tree to find the correct parent for the new node
   while (current != NULL) {
      parent = current;

      // Compare alphabetically and move left or right
      if (strcmp(month, current->month) < 0)
         current = current->left;
      else
         current = current->right;
   }

   // Attach the new node to the appropriate parent
```

```
    newNode->parent = parent;
    if (parent == NULL) {
        // Tree was empty; new node is the root
        tree->root = newNode;
    } else if (strcmp(month, parent->month) < 0) {
        parent->left = newNode;
    } else {
        parent->right = newNode;
    }
}

Node* searchNode(BST* tree, char* month){
    Node* current = tree->root;
    while (current != NULL && strcmp(month, current->month) != 0) {
        if (strcmp(month, current->month) < 0)
            current = current->left;
        else
            current = current->right;
    }
    return current;
}

void removeNode(BST* tree, char* month){
    Node* node = searchNode(tree, month);
    if (node == NULL) {
        printf("Node with month %s not found.\n", month);
        return;
    }

    Node* y;
    Node* x;
    if (node->left == NULL || node->right == NULL)
        y = node;
    else {
        y = node->right;
        while (y->left != NULL)
            y = y->left;
    }

    if (y->left != NULL)
        x = y->left;
    else
        x = y->right;

    if (x != NULL)
        x->parent = y->parent;

    if (y->parent == NULL)
        tree->root = x;
    else if (y == y->parent->left)
        y->parent->left = x;
    else
        y->parent->right = x;

    if (y != node)
        strcpy(node->month, y->month);

    free(y);
}

void traverse(Node* node){
    if (node == NULL)
        return;
    Node* current = node;
    Node* pre;
    while (current != NULL) {
        if (current->left == NULL) {
            printf("%s ", current->month);
            current = current->right;
        } else {
            pre = current->left;
            while (pre->right != NULL && pre->right != current)
                pre = pre->right;
            if (pre->right == NULL) {
                pre->right = current;
                current = current->left;
            } else {
                pre->right = NULL;
                printf("%s ", current->month);
                current = current->right;
            }
        }
    }
    printf("\n");
}
void destroyTree(Node* node){
    if (node == NULL)
        return;
    destroyTree(node->left);
    destroyTree(node->right);
    free(node);
}
```

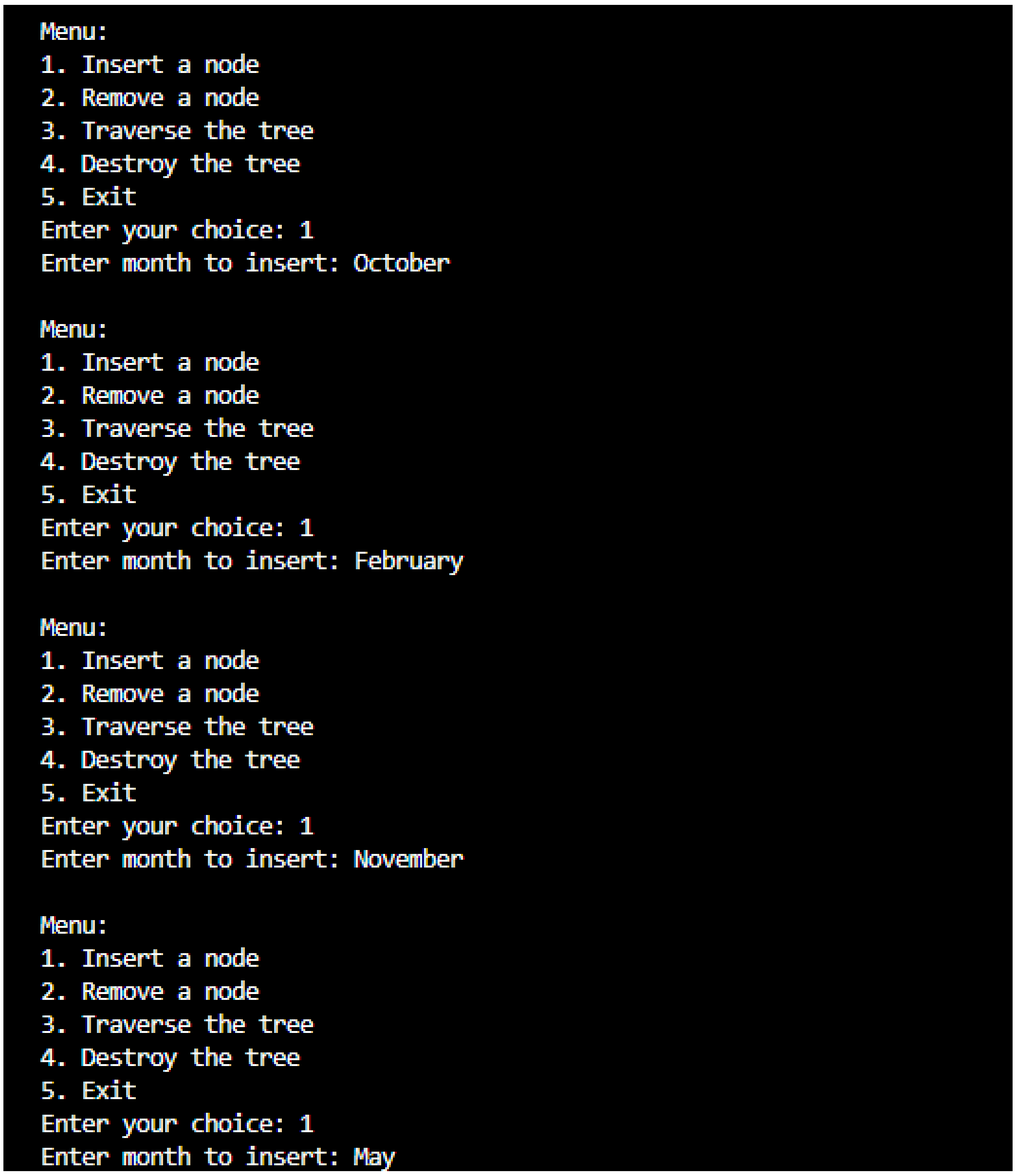
Output:



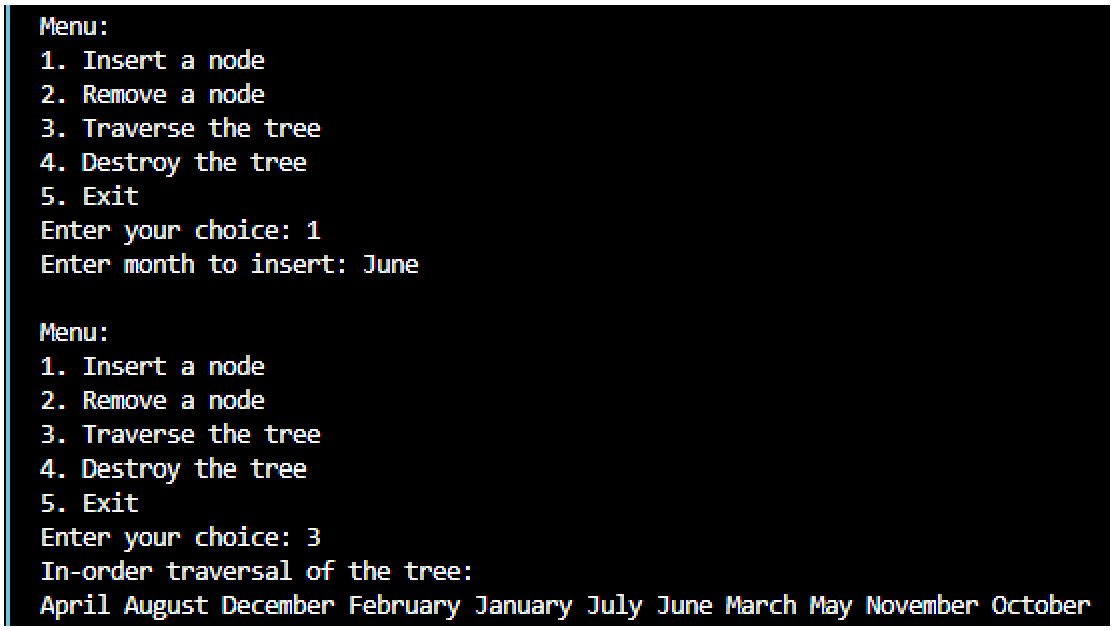
```
tanis@Tanishq MINGW64 /d/COEP/DSA/Serious/Assignments/Assignment5
$ gcc -Wall main.c logic.c

tanis@Tanishq MINGW64 /d/COEP/DSA/Serious/Assignments/Assignment5
$ ./a

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: December

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: January

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: April
```



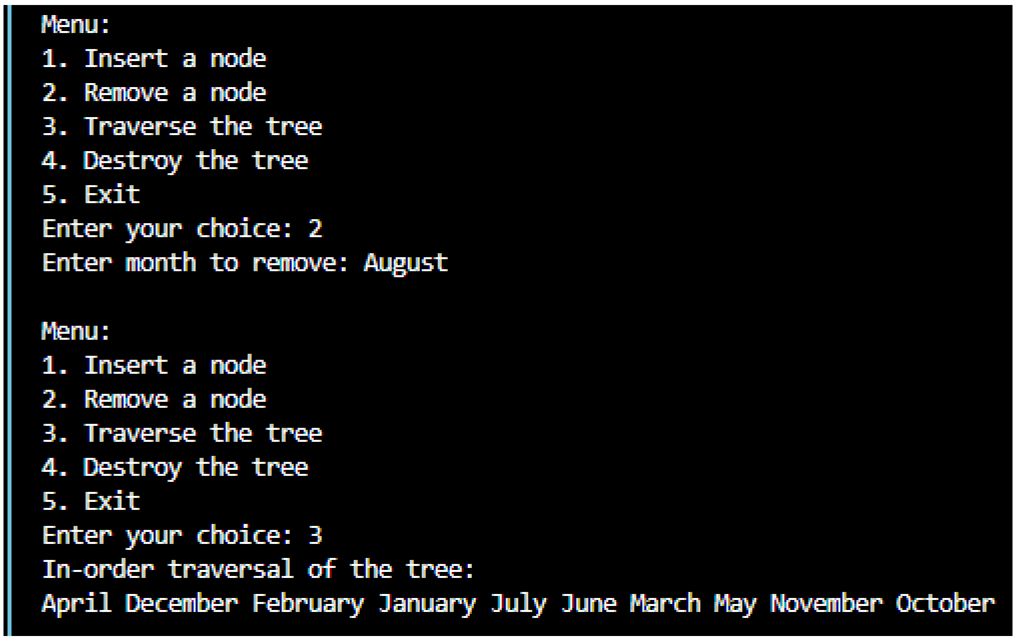
```
Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: April

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: March

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: July

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: August
```



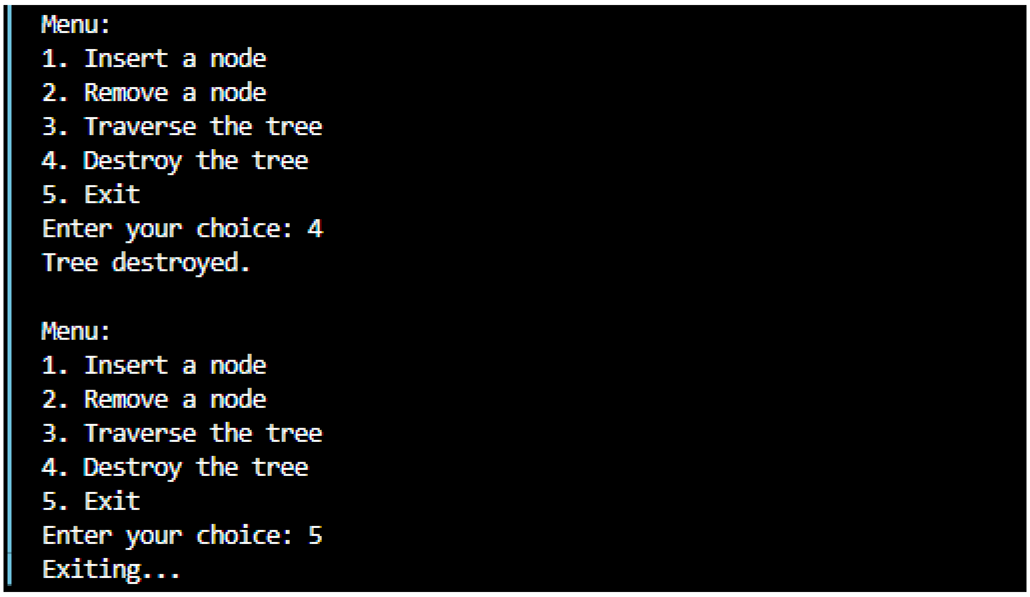
```
Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: October

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: February

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: November

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: May
```



```
Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 1
Enter month to insert: June

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 3
In-order traversal of the tree:
April August December February January July June March May November October
```



```
Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 2
Enter month to remove: August

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 3
In-order traversal of the tree:
April December February January July June March May November October
```



```
Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 4
Tree destroyed.

Menu:
1. Insert a node
2. Remove a node
3. Traverse the tree
4. Destroy the tree
5. Exit
Enter your choice: 5
Exiting...
```