Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

(Autonomous Institute Affiliated to University of Mumbai)

Name: Jhaveri Varun Nimitt

UID: 2023800042

Batch: CSE A Batch C

Experiment No.:5

Aim: Binary Tree Creation

Problem:

1-Creation of binary tree given preorder and Inorder

2- display intermediate output using any one traversal

Solution:

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

struct Node {
   int data;
   struct Node *l, *r;
};

struct Node* nodenew(int data) {
   struct Node* node = (struct Node*)malloc(sizeof(struct Node));
   node->data = data;
   node->l = node->r = NULL;
   return node;
}

void preorder(struct Node* root) {
   if (root == NULL)
       return;
   printf("%d ", root->data);
   preorder(root->l);
   preorder(root->r);
}

int getindex(int* arr, int start, int end, int value) {
   for (int i = start; i <= end; i++) {
       if (arr[i] == value)
           return i;
   }
   return -1;
}

struct Node* posttreemaker(int* inorder, int* postorder, int start, int end, int* postgetindex) {
   if (start > end)
       return NULL;

   struct Node* root = nodenew(postorder[*postgetindex]);
   (*postgetindex)--;

   if (start == end)
       return root;

   int ingetindex = getindex(inorder, start, end, root->data);
   root->r = posttreemaker(inorder, postorder, ingetindex + 1, end, postgetindex);
```

```
    root->l = posttreemaker(inorder, postorder, start, ingetindex - 1, postgetindex);

    return root;
}

struct Node* pretreemaker(int* inorder, int* preorder, int start, int end, int* preindex) {
    if (start > end)
        return NULL;

    struct Node* root = nodenew(preorder[*preindex]);
    (*preindex)++;

    if (start == end)
        return root;

    int ingetindex = getindex(inorder, start, end, root->data);
    root->l = pretreemaker(inorder, preorder, start, ingetindex - 1, preindex);
    root->r = pretreemaker(inorder, preorder, ingetindex + 1, end, preindex);

    return root;
}

void inorder(struct Node* root) {
    if (root == NULL)
        return;
    inorder(root->l);
    printf("%d ", root->data);
    inorder(root->r);
}

bool is_skewed(struct Node* root) {
    if (root == NULL)
        return true;
    if (root->l && root->r)
        return false;
    return is_skewed(root->l) && is_skewed(root->r);
}

bool validate_input(int* inorder, int* order, int n) {
    int found[100] = {0};
    for (int i = 0; i < n; i++) {
        found[inorder[i]]++;
        found[order[i]]++;
        if (found[inorder[i]] > 1 || found[order[i]] > 1)
            return false;  // dupes
    }
    return true;
```

```
}

int main() {
   int n;
   printf("total node: ");
   scanf("%d", &n);

   int inorderArr[n], preorderArr[n], postorderArr[n];

   printf("Enter inorder sequence: ");
   for (int i = 0; i < n; i++)
       scanf("%d", &inorderArr[i]);

   int choice;
   printf("1. preorder+inorder\n 2. postorder+inorder\n");
   scanf("%d", &choice);

   struct Node* root2 = NULL;

   if (choice == 1) {
       printf("Enter preorder sequence: ");
       for (int i = 0; i < n; i++)
           scanf("%d", &preorderArr[i]);

       if (!validate_input(inorderArr, preorderArr, n)) {
           printf("invalid input check again plz.\n");
           return 1;
       }

       int preindex = 0;
       root2 = pretreemaker(inorderArr, preorderArr, 0, n - 1, &preindex);

   } else if (choice == 2) {
       printf("Enter postorder sequence: ");
       for (int i = 0; i < n; i++)
           scanf("%d", &postorderArr[i]);

       if (!validate_input(inorderArr, postorderArr, n)) {
           printf("invalid input check again plz.\n");
           return 1;
       }

       int postgetindex = n - 1;
       root2 = posttreemaker(inorderArr, postorderArr, 0, n - 1, &postgetindex);

   } else {
       printf("somethign went wrong.\n");
```

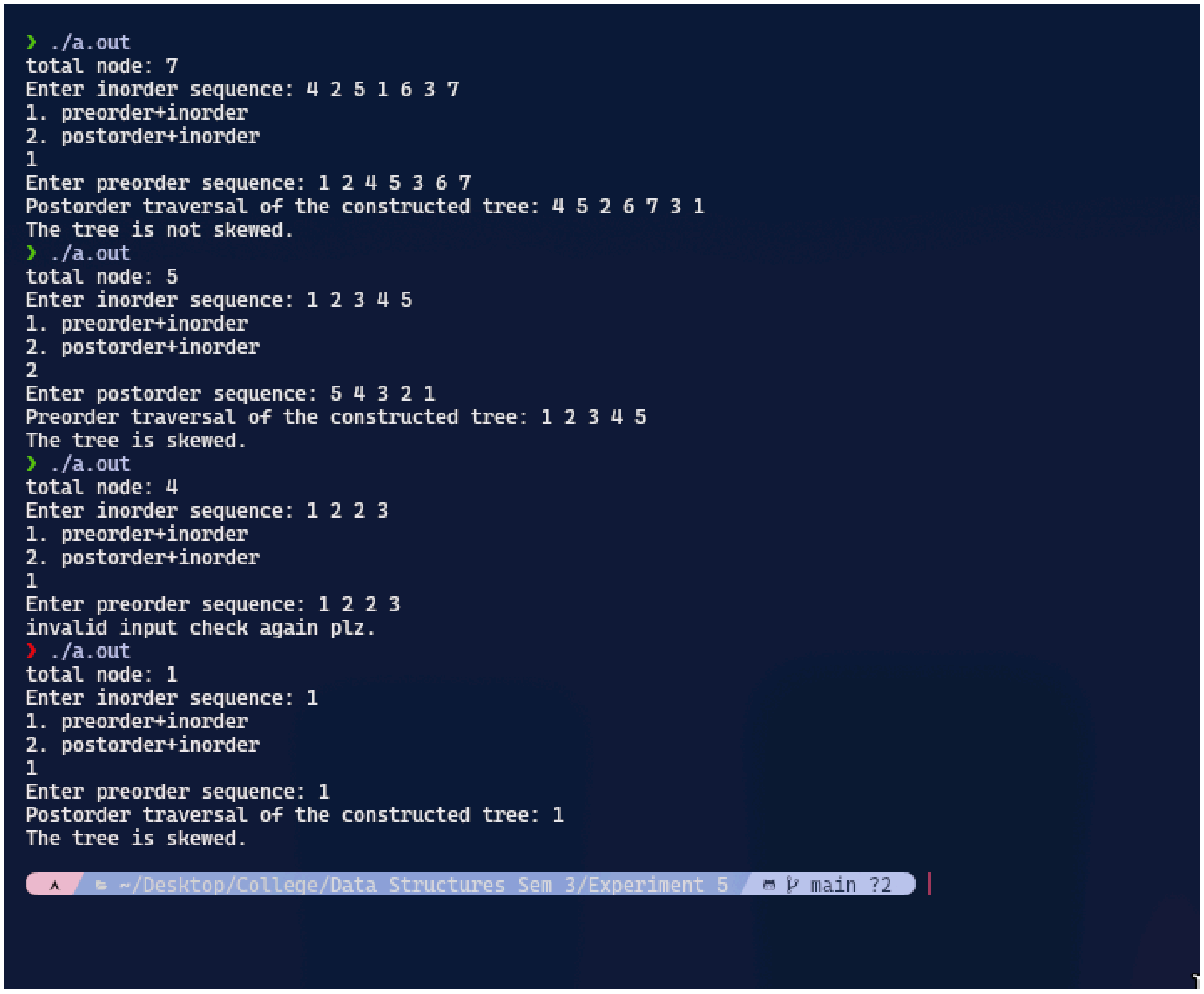
```
        return 1;
    }

    printf("Inorder traversal of the constructed tree: ");
    inorder(root2);
    printf("\n");

    if (is_skewed(root2)) {
        printf("The tree is skewed.\n");
    } else {
        printf("The tree is not skewed.\n");
    }

    return 0;
}
```

OUTPUT:



```
❯ ./a.out
total node: 7
Enter inorder sequence: 4 2 5 1 6 3 7
1. preorder+inorder
2. postorder+inorder
1
Enter preorder sequence: 1 2 4 5 3 6 7
Postorder traversal of the constructed tree: 4 5 2 6 7 3 1
The tree is not skewed.
❯ ./a.out
total node: 5
Enter inorder sequence: 1 2 3 4 5
1. preorder+inorder
2. postorder+inorder
2
Enter postorder sequence: 5 4 3 2 1
Preorder traversal of the constructed tree: 1 2 3 4 5
The tree is skewed.
❯ ./a.out
total node: 4
Enter inorder sequence: 1 2 2 3
1. preorder+inorder
2. postorder+inorder
1
Enter preorder sequence: 1 2 2 3
invalid input check again plz.
❯ ./a.out
total node: 1
Enter inorder sequence: 1
1. preorder+inorder
2. postorder+inorder
1
Enter preorder sequence: 1
Postorder traversal of the constructed tree: 1
The tree is skewed.

 ~/Desktop/College/Data Structures Sem 3/Experiment 5  main ?2
```

top to bottom cases :

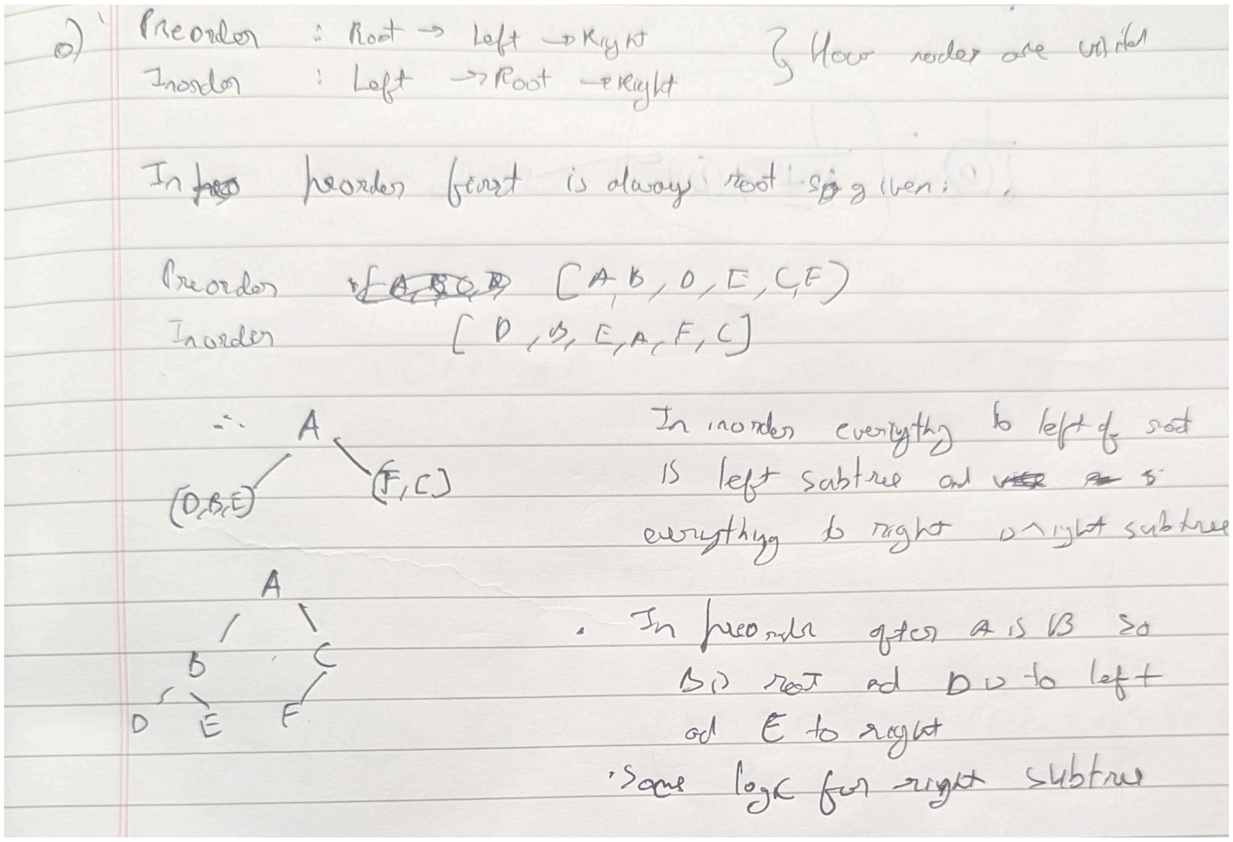
1. balanced tree
2. right skewed tree
3. duplicate error showcase
4. single node tree

Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

(Autonomous Institute Affiliated to University of Mumbai)

Handwritten assignment :



a) Preorder : Root → Left → Right } How order are visited
Inorder : Left → Root → Right

In ~~pre~~ preorder first is always root so given:

Preorder ~~[A,B,C,D]~~ [A, B, D, E, C, F]
Inorder [D, B, E, A, F, C]

$$\begin{tikzcd} & A \arrow[dl, dash] \arrow[dr, dash] & \\ (D,B,E) & & (F,C) \end{tikzcd}$$

In inorder everything to left of root is left subtree and ~~vice~~ ~~versa~~ everything to right is right subtree

$$\begin{tikzcd} & & A \arrow[dl, dash] \arrow[dr, dash] & \\ & B \arrow[dl, dash] \arrow[d, dash] & & C \arrow[dl, dash] \\ D & E & F & \end{tikzcd}$$

- In preorder after A is B so B is root and D is to left and E to right
- Same logic for right subtree