CS102 - Lab 10 - 21/03/2024

```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 10
//PROGRAM 1
//Date : 21/03/2024

/*
1.Given a Binary tree, write a C program to invert the tree, i.e., swap the left
and right children of every node and display the nodes in the preorder traversal.
*/

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

void invert(struct Node* node) {
    if (node == NULL)
        return;
    else {
        struct Node* temp;
        invert(node->left);
        invert(node->right);
```

```
        temp = node->left;
        node->left = node->right;
        node->right = temp;
    }
}

void preOrder(struct Node* root) {
    if (root == NULL)
        return;
    printf("%d ", root->data);
    preOrder(root->left);
    preOrder(root->right);
}

void inOrder(struct Node* root) {
    if (root == NULL)
        return;
    inOrder(root->left);
    printf("%d ", root->data);
    inOrder(root->right);
}


int main() {
    struct Node* root = createNode(50);

    root->left = createNode(17);
    root->right = createNode(72);

    root->left->left = createNode(12);
    root->left->right = createNode(23);

    root->left->left->left = createNode(9);
    root->left->left->right = createNode(14);

    root->left->right->left = createNode(19);

    root->right->left = createNode(54);
    root->right->right = createNode(76);
```

```c
    root->right->left->right = createNode(67);

    printf("Original tree (pre-order traversal): ");
    preOrder(root);
    printf("\n");

    printf("Original tree (in-order traversal): ");
    inOrder(root);
    printf("\n");

    invert(root);

    printf("\n");
    printf("Inverted tree (pre-order traversal): ");
    preOrder(root);
    printf("\n");

    printf("Inverted tree (in-order traversal): ");
    inOrder(root);
    printf("\n");

    return 0;
}
```

OUTPUT:

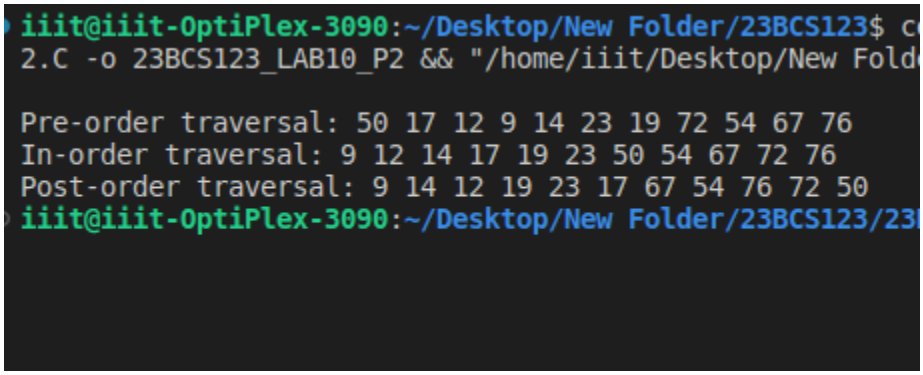
```
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123$ cd "/home/iiit/Desk
1.C -o 23BCS123_LAB10_P1 && "/home/iiit/Desktop/New Folder/23BCS123/23BCS1
Original tree (pre-order traversal): 50 17 12 9 14 23 19 72 54 67 76
Original tree (in-order traversal): 9 12 14 17 19 23 50 54 67 72 76

Inverted tree (pre-order traversal): 50 72 76 54 67 17 23 19 12 14 9
Inverted tree (in-order traversal): 76 72 67 54 50 23 19 17 14 12 9
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123/23BCS123_LAB10$
```

```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 10
//PROGRAM 2
//Date : 21/03/2024

/*
2.Write a C program to implement various traversals of a Binary tree
such as Inorder, postorder and preorder traversals.
*/

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

struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}

void preOrder(struct Node* root) {
   if (root == NULL)
       return;
   printf("%d ", root->data);
   preOrder(root->left);
   preOrder(root->right);
}

void inOrder(struct Node* root) {
   if (root == NULL)
       return;
```

```
    inOrder(root->left);
    printf("%d ", root->data);
    inOrder(root->right);
}

void postOrder(struct Node* root) {
    if (root == NULL)
        return;
    postOrder(root->left);
    postOrder(root->right);
    printf("%d ", root->data);
}

int main() {
    struct Node* root = createNode(50);

    root->left = createNode(17);
    root->right = createNode(72);

    root->left->left = createNode(12);
    root->left->right = createNode(23);

    root->left->left->left = createNode(9);
    root->left->left->right = createNode(14);

    root->left->right->left = createNode(19);

    root->right->left = createNode(54);
    root->right->right = createNode(76);

    root->right->left->right = createNode(67);


    printf("\nPre-order traversal: ");
    preOrder(root);
    printf("\n");

    printf("In-order traversal: ");
    inOrder(root);
    printf("\n");
```

```
    printf("Post-order traversal: ");
    postOrder(root);
    printf("\n");

    return 0;
}
```

OUTPUT:



```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 10
//PROGRAM 3
//Date : 21/03/2024

/*
3.Given a doubly linked list with an even number of elements, write
a C program swap every two adjacent nodes and print the resultant linked
list.
*/
#include <stdio.h>
#include <stdlib.h>

typedef struct node {
   int data;
   struct node *prev;
   struct node *next;
} node;

void traverse(node *head) {
   printf("\nDoubly Linked List:\n");
   node *current = head;
   int count = 1;
   while (current != NULL) {
       printf("  Node_%d: %d\n", count, current->data);
       current = current->next;
       count++;
   }
}

node* create_node(int item) {
   node *new_node = (node*)malloc(sizeof(node));
   if (new_node == NULL) {
       printf("Memory allocation failed\n");
       exit(EXIT_FAILURE);
```

```
    }
    new_node->data = item;
    new_node->prev = NULL;
    new_node->next = NULL;
    return new_node;
}

void insert_end(node **head, int item) {
    node *new_node = create_node(item);

    if (*head == NULL) {
        *head = new_node;
        return;
    }
    node *p = *head;
    while (p->next != NULL) {
        p = p->next;
    }
    p->next = new_node;
    new_node->prev = p;
}

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void swap_adjacent_nodes(node *head) {
    node *current = head;

    while (current != NULL && current->next != NULL) {
        swap(&(current->data), &(current->next->data));
        current = current->next->next;
    }
}

int main() {
    node *head = NULL;
    int choice, item;
```

```
    insert_end(&head, 1);
    insert_end(&head, 2);
    insert_end(&head, 3);
    insert_end(&head, 4);
    insert_end(&head, 5);
    insert_end(&head, 6);
    traverse(head);

    while(1) {
        printf("1.Swap adjacent nodes 2.Exit || Enter your choice:");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                swap_adjacent_nodes(head);
                printf("Adjacent nodes swapped.\n");
                traverse(head);
                break;
            case 2:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

OUTPUT:

```
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123$ cd "/home/
3.C -o 23BCS123_LAB10_P3 && "/home/iiit/Desktop/New Folder/23BCS1

Doubly Linked List:
  Node_1: 1
  Node_2: 2
  Node_3: 3
  Node_4: 4
  Node_5: 5
  Node_6: 6
1.Swap adjacent nodes 2.Exit || Enter your choice:1
Adjacent nodes swapped.

Doubly Linked List:
  Node_1: 2
  Node_2: 1
  Node_3: 4
  Node_4: 3
  Node_5: 6
  Node_6: 5
1.Swap adjacent nodes 2.Exit || Enter your choice:1
Adjacent nodes swapped.

Doubly Linked List:
  Node_1: 1
  Node_2: 2
  Node_3: 3
  Node_4: 4
  Node_5: 5
  Node_6: 6
1.Swap adjacent nodes 2.Exit || Enter your choice:
```