**CODE:-**

#include <stdio.h>

#include <stdlib.h>

struct treenode

{

struct treenode \*lchild;

int info;

struct treenode \*rchild;

};

struct listnode

{

int info;

struct listnode \*next;

} \*postptr = NULL, \*inptr = NULL;

struct listnode \*create\_linked\_list(struct listnode \*start, int n)

{

struct listnode \*temp, \*p;

start = NULL;

if (n == 0)

return start;

for (int i = 0; i < n; i++)

{

temp = (struct listnode \*)malloc(sizeof(struct listnode));

scanf("%d", &temp->info);

temp->next = NULL;

if (start == NULL)

start = temp;

else

{

p = start;

while (p->next != NULL)

p = p->next;

p->next = temp;

}

}

return start;

}

struct treenode \*construct\_in\_post(struct listnode \*inptr, struct listnode \*postptr, int num)

{

struct treenode \*temp;

struct listnode \*q, \*ptr;

int i, j;

if (num == 0)

return NULL;

ptr = postptr;

for (i = 1; i < num; i++)

ptr = ptr->next;

temp = (struct treenode \*)malloc(sizeof(struct treenode));

temp->info = ptr->info;

temp->lchild = NULL;

temp->rchild = NULL;

if (num == 1)

return temp;

q = inptr;

for (i = 0; q->info != ptr->info; i++)

q = q->next;

temp->lchild = construct\_in\_post(inptr, postptr, i);

for (j = 1; j <= i; j++)

postptr = postptr->next;

temp->rchild = construct\_in\_post(q->next, postptr, num - i - 1);

return temp;

}

int height(struct treenode \*ptr)

{

int h\_left, h\_right;

if (ptr == NULL)

return 0;

h\_left = height(ptr->lchild);

h\_right = height(ptr->rchild);

if (h\_left > h\_right)

return 1 + h\_left;

else

return 1 + h\_right;

}

int finddepth(struct treenode \*ptr, int skey)

{

if (ptr == NULL)

return -1;

int depth = -1;

if ((ptr->info == skey) || (depth = finddepth(ptr->lchild, skey) >= 0) || (depth = finddepth(ptr->rchild, skey) >= 0))

return depth+1;

return depth;

}

void displaylr(struct treenode \*ptr, int level)

{

if (ptr == NULL)

return;

if (level == 1)

printf("%d ", ptr->info);

else if (level > 1)

{

displaylr(ptr->lchild, level - 1);

displaylr(ptr->rchild, level - 1);

}

}

void displayrl(struct treenode \*ptr, int level)

{

if (ptr == NULL)

return;

if (level == 1)

printf("%d ", ptr->info);

else if (level > 1)

{

displayrl(ptr->rchild, level - 1);

displayrl(ptr->lchild, level - 1);

}

}

void levelorder(struct treenode \*ptr)

{

int h = height(ptr);

for (int i = 1; i <= h; i++)

displaylr(ptr, i);

printf("\n");

}

void spiralorder(struct treenode \*ptr)

{

int h = height(ptr);

for (int i = 1; i <= h; i++)

{

if (i % 2 == 1)

displaylr(ptr, i);

else

displayrl(ptr, i);

}

printf("\n");

}

int main()

{

struct listnode \*start1 = NULL, \*start2 = NULL;

struct treenode \*root1;

int num, ch, skey;

printf("Enter the number of nodes.\n");

scanf("%d", &num);

printf("Enter the inorder list.\n");

start1 = create\_linked\_list(start1, num);

printf("Enter the postorder list.\n");

start2 = create\_linked\_list(start2, num);

root1 = construct\_in\_post(start1, start2, num);

while (1)

{

printf("Enter 1 to find height of the tree.\n");

printf("Enter 2 to find depth of the given node.\n");

printf("Enter 3 to perform level order traversal.\n");

printf("Enter 4 to perform spiral order traversal.\n");

printf("Enter 5 to exit.\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("The height of the tree is %d.\n", height(root1));

break;

case 2:

printf("Enter the value of the node.\n");

scanf("%d", &skey);

if (finddepth(root1, skey) == -1)

printf("Tree is empty or node is not found.\n");

else

printf("The depth of the node in the tree is %d.\n", finddepth(root1, skey));

break;

case 3:

levelorder(root1);

break;

case 4:

spiralorder(root1);

break;

case 5:

exit(1);

default:

printf("Erroneous input.\n");

}

}

return 0;

}

**OUTPUT:-**



