//Name: Mehmet Fatih Çelik

//ID: 2385268

#include <stdio.h>

#include <stdlib.h>

struct TreeNode{

int val;

struct TreeNode \*left;

struct TreeNode \*right;

};

typedef struct TreeNode \*Tree;

struct QueueNode{

Tree val;

struct QueueNode \*next;

};

struct QueueRecord{

struct QueueNode \*front;

struct QueueNode \*rear;

int size;

};

typedef struct QueueRecord \*Queue;

Tree CreateTree(int, Tree);

void LevelOrderDisplay(Tree);

Queue createQueue();

void Enqueue(Queue, Tree);

Tree Dequeue(Queue);

int main(){

Tree root = NULL;

int value, i;

printf("Please enter a value: ");

scanf("%d",&value);

root = CreateTree(value, root);

for(i=0;i<5;i++){ // since it is written like: "takes 6 elements from the user", I specified like that.

printf("Please enter a value: ");

scanf("%d",&value);

CreateTree(value, root);

}

LevelOrderDisplay(root);

return 0;

}

Tree CreateTree(int x, Tree t){

if (t == NULL){

t = (Tree)malloc(sizeof(Tree));

if (t == NULL){

printf("Out of space!");

exit(-1);

}

else{

t->val = x;

t->left = NULL;

t->right = NULL;

}

}

else if (x < t->val)

t->left = CreateTree(x, t->left);

else if (x > t->val)

t->right = CreateTree(x, t->right);

return t;

}

void LevelOrderDisplay(Tree t){

Tree temp = t;

Queue q;

q = createQueue();

while(temp != NULL){

printf("%d ",temp->val);

if (temp->left)

Enqueue(q, temp->left);

if (temp->right);

Enqueue(q, temp->right);

temp = Dequeue(q);

}

}

Queue createQueue(){

Queue q;

q = (Queue)malloc(sizeof(Queue));

if (q == NULL){

printf("Out of space!");

exit(-1);

}

q->size = 0;

q->front = (struct QueueNode\*)malloc(sizeof(struct QueueNode));

if (q->front == NULL){

printf("Out of space!");

exit(-1);

}

q->front->next = NULL;

q->rear = q->front;

return q;

}

void Enqueue(Queue q, Tree x){

struct QueueNode \*temp;

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

if (temp == NULL){

printf("Out of space!");

exit(-1);

}

temp->next = NULL;

temp->val = x;

q->rear->next = temp;

q->rear = temp;

q->size++;

}

Tree Dequeue(Queue q){

Tree x;

struct QueueNode \*removal;

removal = q->front->next;

x = removal->val;

q->front->next = removal->next;

free(removal);

q->size--;

if (q->size == 0)

q->rear = q->front;

return x;

}