```c
// https://www.geeksforgeeks.org/iterative-tower-of-hanoi/
// Edited to conform to C Style

#include <stdlib.h>

typedef struct Stack {
  unsigned capacity;
  int top;
  int *array;
}Stack;

struct Stack* create_stack(unsigned capacity)
{
    struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack)); // stack == self
    stack -> capacity = capacity;
    stack -> top = -1;
    stack -> array = (int*) malloc(stack -> capacity * sizeof(int));
    return stack;
}

int is_full(struct Stack* stack)
{
return (stack->top == stack->capacity - 1); // indicates if top pointer is at last index.
}

// Stack is empty when top is equal to -1
int is_empty(struct Stack* stack)
{
return (stack->top == -1);                  // Stack is empty if top is equal to -1
}

// Function to add an item to stack. It increases
// top by 1
void push(struct Stack *stack, int item)
{
    if (is_full(stack)) return;
    stack -> array[++stack->top] = item;    //preincrement top index and add item there.
}

// Function to remove an item from stack. It
// decreases top by 1
int pop(struct Stack* stack)
{
    if (is_empty(stack)) return 0;
    return stack -> array[stack->top--];    //return item at top index and postdecrement top.
}

// 03/30/24 added by Enrique
int peek(struct Stack* stack){
    if (is_empty(stack)) return 0;
    return stack -> array[stack->top];
}
```