3) (10 pts) DSN (Stacks)

A word is considered a palindrome if the reverse of the word is the same as the original word. For example: the word "test" is not a palindrome as its reverse "tset" is not the same as "test". On the other hand, the word "racecar" is a palindrome as its reverse is the same as "racecar". Some other examples of palindromes are "hannah", "level", "madam", and "yay."

Write a function that will take a string in the parameter and returns 1, if the string is a palindrome, otherwise returns 0. **You have to use stack operations during this process.** (Credit isn't awarded for correctly solving the problem, but for utilizing the stack in doing so.)

Assume the following stack definition and the functions already available to you. The stack will be extended automatically if it gets full (so you, don't have to worry about it). The top of the stack is controlled by your push and pop operation as usual stack operations.

```
void initialize(stack* s); // initializes an empty stack.
int push(stack* s, char value); //pushes the char value to the stack
int isEmpty(stack* s); // Returns 1 if the stack is empty, 0 otherwise.
char pop(stack* s); // pops and returns character at the top of the stack.
char peek(stack* s); // returns character at the top of the stack.
```

Note: pop and peek return 'I' if the stack s is empty.

```
int isPalindrome(char *str) {

    struct stack s;
    initialize(&s);
    int len = strlen(str);

    for (int i=0; i<len/2; i++)          // Loop can go to len
        push(&s, str[i]);

    for (int i=len-len/2; i<len; i++)    // if first goes to len
        if (pop(&s) != str[i])           // this one must start
            return 0;                    // at 0.

    return 1;                            // ok to check for empty

}
```

Note: The second for-loop could be written like so: `for (int i=(len+1)/2; i<len; i++)`

Grading: 2 pts push first half,
2 pts correct # of pushes (note: if string length is odd, they can push the middle element or not; that doesn't affect credit here, but it could affect the next 2 points below)
2 pts pop off second half (give 1 pt if off by 1) (if string length is odd and they pushed the middle element, they must also pop it here in order to get all 2 points)
1 pt return 0 as soon as error is spotted
2 pts indexing and # of pops is accurate (give 1 pt if off by 1)
1 pt return 1 at end (note stack should be empty via # of pushes/pops so no need to check)