3) (10 pts) DSN (Stacks)

Consider a string mathematical expression can have two kind of parenthesis '(' and '{'. The parenthesis in an expression can be imbalanced, if they are not closed in the correct order, or if there are extra starting parenthesis or extra closing parenthesis. The following table shows some examples:

Expression	Status	Expression	Status
({)}	Imbalanced due to incorrect order of).	({ }))	Imbalanced due to extra)
(( )	Imbalanced due to extra (	({ })	Balanced
{ ( ) }	Balanced		

Write a function that will take an expression in the parameter and returns 1, if the expression is balanced, otherwise returns 0. You have to use stack operations during this process. Assume the following stack definition and the functions already available to you. You may assume that the stack has enough storage to carry out the desired operations without checking.

```
void initialize(stack* s); // initializes an empty stack.
void 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.

```
// Pre-condition: e only contains the characters '(',')','{' and '}'.
int isBalanced(char *e) {
    struct stack s;
    initialize(&s);
    for(int i=0; e[i]!='\0'; i++) {

        if(e[i] == '(' || e[i] == '{')        // 2 pts
            push(&s, e[i]);                   // 1 pt

        else if(e[i] == ')') {                // 1 pt
            if(pop(&s) != '(') return 0;      // 2 pts
        }
        else if(e[i] == '}') {                // 1 pt
            if(pop(&s) != '{') return 0;      // 2 pts
        }
    }

    return isEmpty(&s) ;                      // 1 pt
}
```