Lab #7

Summer 2024

Requirements

In this lab, you will cover creating and maintaining a Stack ADT. You are only given a partial definition of the Stack implementation, and you must derive the remainder of the implementation from the given complexity requirements. Note that you may need to define extra helper functions or struct types to complete this lab, and that these extra definitions must go in your **lab7.c** file.

In this lab you are given the following struct definitions:

```
// This is a partial type, you must complete it in your lab7.c file.
// Note that you cannot change lab7.h
typedef struct _Info Info;
typedef struct _Stack Stack;


struct _Stack {
        Info *info;
};
```

1.1 initStack

```
// O(1)
Stack * initStack()
```

Info: This function will initialize and return a Stack. If initialization fails, the *info* pointer in the returned pointer to a stack must be set to NULL. Your grade for this function will also include your stack implementation.

1.2 getSize

```
// O(1)
int getSize(Stack *s)
```

Info: This function takes a stack pointer, and returns the number of elements on the stack.

1.3 peekStack

```
// O(1)
void * peekStack(Stack *s)
```

Info: This function takes a stack pointer, and returns the element at the top of the stack **without removing it**. It should return NULL if the stack is empty.

1.4 pushStack

```
// O(1)
int pushStack(Stack *s, void *data)
```

Info: This function takes a stack pointer, as well as a data item. It will push the item onto the top of the stack, and return 0 if insertion was successful, or 1 if it was not.

1.5 popStack

```
// O(1)
void * popStack(Stack *s)
```

Info: This function takes a stack pointer, and pops the data item from the top of the stack. It returns the item popped from the stack, or NULL if the stack is empty.

1.6 stackContains

```
// O(n)
int stackContains(Stack *s, void *data)
```

Info: This function takes a stack pointer, as well as a data item. It returns 1 if the given data exists on the stack, or 0 if it does not.

1.7 freeStack

```
// O(n)
void freeStack(Stack *s)
```

Info: This function takes a stack pointer, and frees all memory allocated to the stack. **Remember that data which have been inserted onto the stack are not considered part of the memory allocated to the stack.**

Submission Information

Submit your lab7.c file by using the mucsmake command.

Use the following submit command on Hellbender:

mucsmake <course> <assignment> <filename>

For example:

mucsmake 2050 lab7 lab7.c

Rubric: 24 points

1. Write required *initStack* function
   * 10 points
2. Write required *getSize* function
   * 1 points
3. Write required *peekStack* function
   * 2 points
4. Write required *pushStack* function
   * 2 points
5. Write required *popStack* function
   * 2 points
6. Write required *stackContains* function
   * 4 points
7. Write required *freeStack* function
   * 3 points

Notice:

1. All of your lab submissions **must** include documentation in the form of code comments to receive full points. In addition, your program is expected to have a **comment header** at the top that includes your name, pawprint, the course you are taking, and the lab that you solved. You can refer to the Lab 0 document for an example of the comment header.
2. All of your lab submissions must compile under GCC using the -Wall and -Werror flags to be considered for a grade. These flags will automatically be applied if you use the compile command.
3. Do **NOT** change the given function prototype or anything else in the provided .h file. Additional definitions/functions **must** be placed in your lab7.c file.