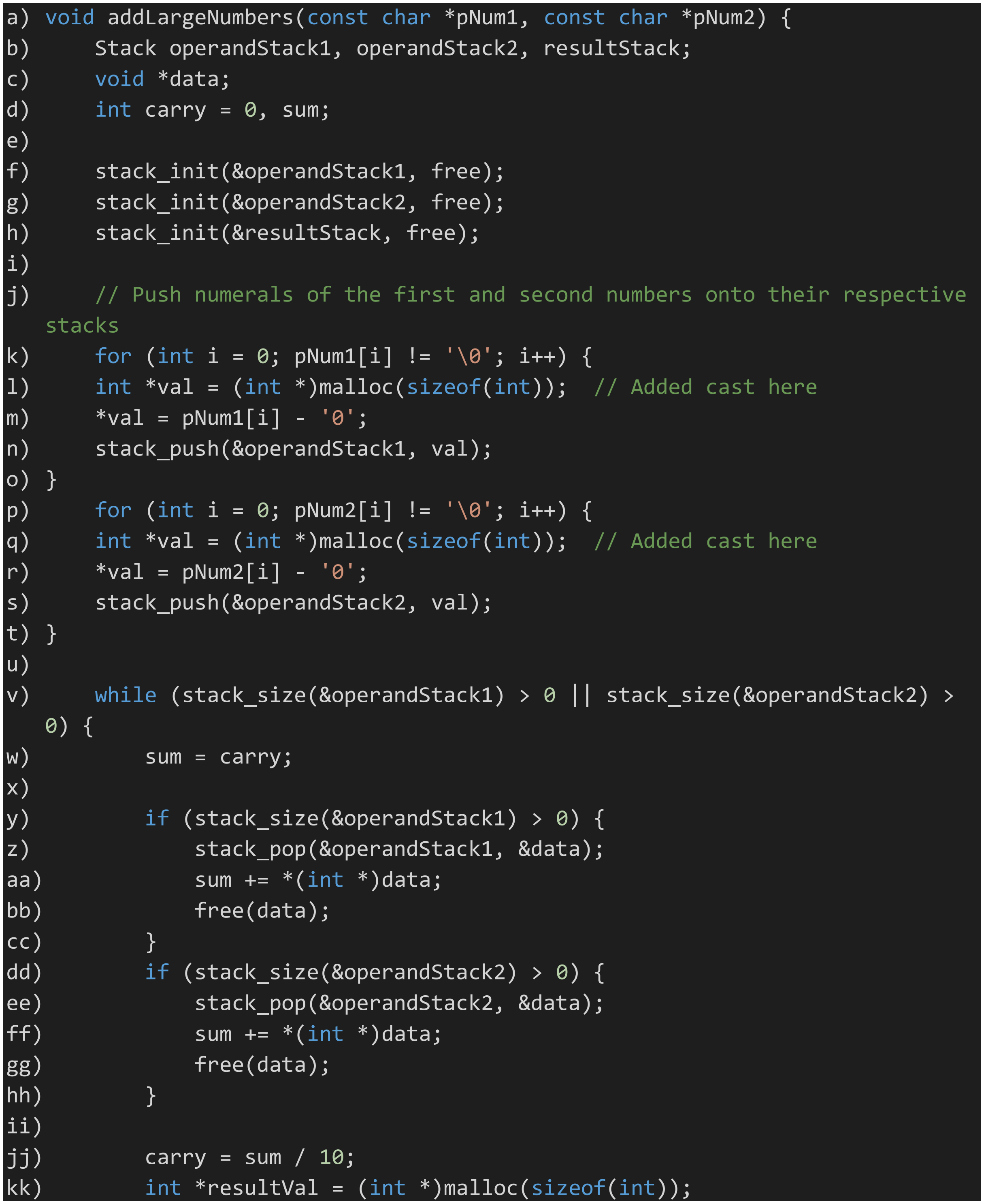
(6 point) Implement the addLargeNumbers function with the following prototype: void addLargeNumbers(const char *pNum1, const char *pNum2); This function should output the result of adding the two numbers passed in as strings. Here is an example call to this function with the expected output: /* Sample call to addLargeNumbers */ addLargeNumbers("592", "3784"); /* Expected output */ 4376



```
a) void addLargeNumbers(const char *pNum1, const char *pNum2) {
b)     Stack operandStack1, operandStack2, resultStack;
c)     void *data;
d)     int carry = 0, sum;
e)
f)     stack_init(&operandStack1, free);
g)     stack_init(&operandStack2, free);
h)     stack_init(&resultStack, free);
i)
j)     // Push numerals of the first and second numbers onto their respective stacks
k)     for (int i = 0; pNum1[i] != '\0'; i++) {
l)     int *val = (int *)malloc(sizeof(int));  // Added cast here
m)     *val = pNum1[i] - '0';
n)     stack_push(&operandStack1, val);
o) }
p)     for (int i = 0; pNum2[i] != '\0'; i++) {
q)     int *val = (int *)malloc(sizeof(int));  // Added cast here
r)     *val = pNum2[i] - '0';
s)     stack_push(&operandStack2, val);
t) }
u)
v)     while (stack_size(&operandStack1) > 0 || stack_size(&operandStack2) > 0) {
w)         sum = carry;
x)
y)         if (stack_size(&operandStack1) > 0) {
z)             stack_pop(&operandStack1, &data);
aa)            sum += *(int *)data;
bb)            free(data);
cc)        }
dd)        if (stack_size(&operandStack2) > 0) {
ee)            stack_pop(&operandStack2, &data);
ff)            sum += *(int *)data;
gg)            free(data);
hh)        }
ii)
jj)        carry = sum / 10;
kk)        int *resultVal = (int *)malloc(sizeof(int));
```

```c
        *resultVal = sum % 10;
        stack_push(&resultStack, resultVal);
    }
    if (carry != 0) {
        int *resultVal = (int *)malloc(sizeof(int));
        *resultVal = carry;
        stack_push(&resultStack, resultVal);
    }

    // Pop and display the result
    while (stack_size(&resultStack) > 0) {
        stack_pop(&resultStack, &data);
        printf("%d", *(int *)data);
        free(data);
    }
    printf("\n");

    stack_destroy(&operandStack1);
    stack_destroy(&operandStack2);
    stack_destroy(&resultStack);
}
```

(3 points) Implement a test program that demonstrates adding at least three pairs of large numbers (numbers larger than can be represented by a long).

```c
int main() {
    // Sample call
    addLargeNumbers("592", "3784"); // Expected output: 4376

    // Test cases for numbers larger than can be represented by a long
    addLargeNumbers("12345678901234567890", "98765432109876543210");// Expected output: 111111111111111111100
    addLargeNumbers("918273645812634851283648 51", "918273645812634851283648 51");
    addLargeNumbers("123456789012345678900123456 7890", "987654321098765432109870654 3210");

    return 0;
}
```

Source file:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"
#include "stack.h"

//Prototype
void addLargeNumbers(const char *pNum1, const char *pNum2);

//Main program
int main() {
    // Sample call
    addLargeNumbers("592", "3784"); // Expected output: 4376

    // Test cases for numbers larger than can be represented by a long
    addLargeNumbers("12345678901234567890", "98765432109876543210");// Expected
output: 111111111111111111100
    addLargeNumbers("91827364581263485128364851", "91827364581263485128364851");
    addLargeNumbers("1234567890123456789001234567890",
"9876543210987654321098706543210");

    return 0;
}
void addLargeNumbers(const char *pNum1, const char *pNum2) {
    Stack operandStack1, operandStack2, resultStack;
    void *data;
    int carry = 0, sum;

    stack_init(&operandStack1, free);
    stack_init(&operandStack2, free);
    stack_init(&resultStack, free);

    // Push numerals of the first and second numbers onto their respective stacks
    for (int i = 0; pNum1[i] != '\0'; i++) {
    int *val = (int *)malloc(sizeof(int));  // Added cast here
    *val = pNum1[i] - '0';
    stack_push(&operandStack1, val);
}
    for (int i = 0; pNum2[i] != '\0'; i++) {
    int *val = (int *)malloc(sizeof(int));  // Added cast here
    *val = pNum2[i] - '0';
    stack_push(&operandStack2, val);
}

    while (stack_size(&operandStack1) > 0 || stack_size(&operandStack2) > 0) {
        sum = carry;

        if (stack_size(&operandStack1) > 0) {
            stack_pop(&operandStack1, &data);
            sum += *(int *)data;
            free(data);
        }
        if (stack_size(&operandStack2) > 0) {
            stack_pop(&operandStack2, &data);
            sum += *(int *)data;
            free(data);
        }

        carry = sum / 10;
        int *resultVal = (int *)malloc(sizeof(int));
        *resultVal = sum % 10;
        stack_push(&resultStack, resultVal);
    }
    if (carry != 0) {
        int *resultVal = (int *)malloc(sizeof(int));
        *resultVal = carry;
        stack_push(&resultStack, resultVal);
    }

    // Pop and display the result
    while (stack_size(&resultStack) > 0) {
        stack_pop(&resultStack, &data);
        printf("%d", *(int *)data);
        free(data);
    }
    printf("\n");

    stack_destroy(&operandStack1);
    stack_destroy(&operandStack2);
    stack_destroy(&resultStack);
}
```

Output:

```
4376

366490028597211402711138110

183654729162526970256729702
```

111111111011111111101111111100

Stack.h

```
/*
 * stack.h
 */
#ifndef STACK_H
#define STACK_H

#include <stdlib.h>

#include "list.h"

/*
 * Implement stacks as linked lists.
 */
typedef List Stack;

/*
 * Public Interface
 */

// Initialize the stack
void stack_init(Stack *stack, void (*destroy)(void *data));

// Destroy the stack
void stack_destroy(Stack *stack);

// Push an element onto the stack
int stack_push(Stack *stack, const void *data);

// Pop an element off the stack
int stack_pop(Stack *stack, void **data);

// Get the element at the top of the stack without removing it
void *stack_peek(Stack *stack);

// Get the current size of the stack
int stack_size(Stack *stack);

#endif
```

Stack.c

```c
/*
 * stack.c
 */
#include <stdlib.h>

#include "list.h"
#include "stack.h"

// Initialize the stack
void stack_init(Stack *stack, void (*destroy)(void *data)) {
    list_init(stack, destroy);
}

// Destroy the stack
void stack_destroy(Stack *stack) {
    list_destroy(stack);
}

// Push data onto the stack
int stack_push(Stack *stack, const void *data) {
    // Insert the data at the beginning of the list
    return list_ins_next(stack, NULL, data);
}

// Pop data off the stack
int stack_pop(Stack *stack, void **data) {
    // Remove the first element from the list
    return list_rem_next(stack, NULL, data);
}

// Peek at the top of the stack
void *stack_peek(Stack *stack) {
    // Return the data at the beginning of the list if it's not empty
    return (stack->head == NULL ? NULL : stack->head->data);
}

// Return the size of the stack
int stack_size(Stack *stack) {
    return list_size(stack);
}
```