Homework #4

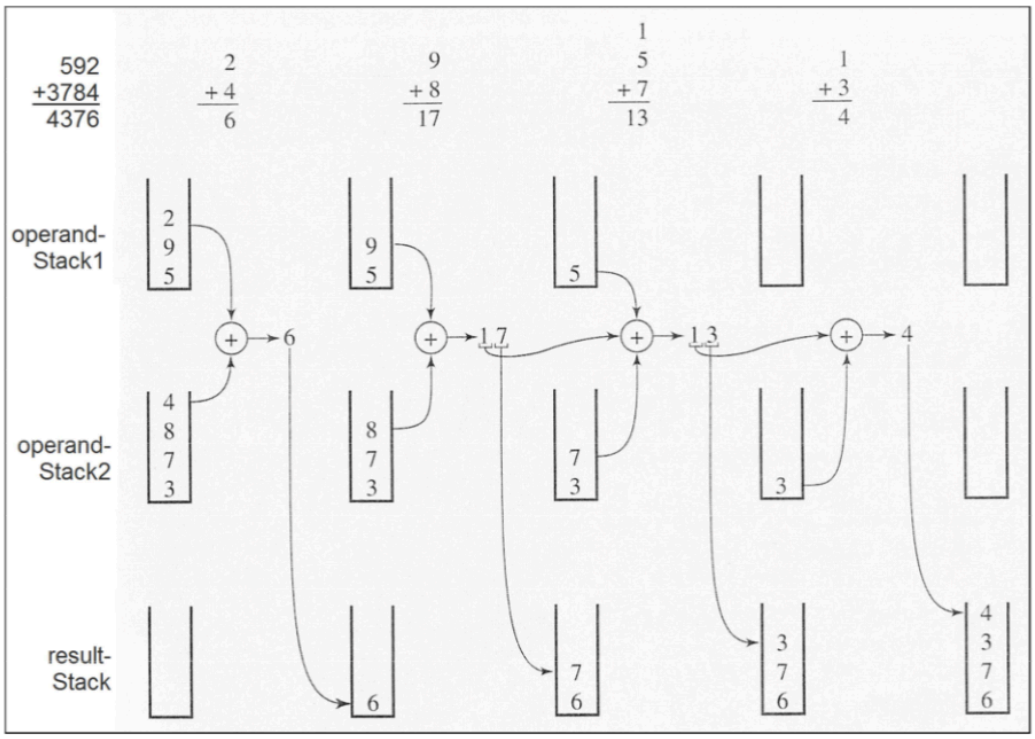
In this assignment you will implement an algorithm that uses stacks to add numbers of any size.

The largest magnitude of integers is limited. We are not able to add 18,274,364,583,929,273,748,525 and 8,129,498,165,026,350,236 because integer variables cannot hold such large values, let alone their sum.

The problem can be solved if we treat these numbers as strings of numerals, store the numbers corresponding to these numerals on two stacks, and then perform addition by popping out the stacks. The pseudocode for this algorithm is as follows:

addLargeNumbers (number1, number2)
read the numerals of the first number and store them on one stack
read the numerals of the second number and store them on another stack
var result :=0
while at least one stack is not empty
pop a numeral from each nonempty stack and add them to result
push the unit part of addition onto a new stack called the result stack
push result onto the result stack if it is not zero
pop numbers from the result stack and display them

The following diagram shows an example of adding numbers 592 and 3,784:



a) **(6 points)** 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

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

void addLargeNumbers(const char *pNum1, const char *pNum2) {
    // initialize stacks
    Stack stack1, stack2, resultStack;
    stack_init(&stack1, free);
    stack_init(&stack2, free);
    stack_init(&resultStack, free);

    // Read the numerals of the first number and store them on one stack
    for (int i = 0; i < strlen(pNum1); i++) {        // Read from last char of the string
        char *digit = (char *)malloc(sizeof(char));
        *digit = pNum1[i];
        stack_push(&stack1, digit);
    }

    // Read the numerals of the second number and store them on another stack
    for (int i = 0; i < strlen(pNum2); i++) {
        char *digit = (char *)malloc(sizeof(char));
        *digit = pNum2[i];
        stack_push(&stack2, digit);
    }

    // Initialize carry variable
    int carry = 0;

    // While at least one stack is not empty
    while (stack_size(&stack1) > 0 || stack_size(&stack2) > 0 || carry > 0) {
        int num1_digit = 0, num2_digit = 0;

        // Pop a numeral from each nonempty stack
        if (stack_size(&stack1) > 0) {
            char *top;
            stack_pop(&stack1, (void **)&top);
            num1_digit = *top - '0';                        // Convert from char to int
            free(top);
        }

        if (stack_size(&stack2)> 0) {
            char *top;
            stack_pop(&stack2, (void **)&top);
            num2_digit = *top - '0';                        // Convert from char to int
            free(top);
        }

        // Add them to result
        int result = num1_digit + num2_digit + carry;
        carry = result / 10;                                // Carry is either 1 or 0 (truncated)
        result %= 10;                                       // Result is one's digit

        /* Push the unit part of addition onto resultStack */
        char *result_digit = (char *)malloc(sizeof(char));
        *result_digit = (result % 10) + '0';                // Convert from int back to char
        stack_push(&resultStack, result_digit);
    }

    // Pop numbers from resultStack and display them
    printf("%s + %s = ", pNum1, pNum2);
    while (list_size(&resultStack) > 0) {
        char *top;
        stack_pop(&resultStack, (void **)&top);
        printf("%c", *top);
        free(top);
    }
    printf("\n");

    // Destroy stacks
    stack_destroy(&stack1);
    stack_destroy(&stack2);
    stack_destroy(&resultStack);
}
```

```
int main() {
    addLargeNumbers("592", "3784");
    // addLargeNumbers("213091240876702312301 3", "2184120123123311231223 1");
    return 0;
}
```

```
~/Desktop/DSA/hw4 main* ❯ /Users/jeffylee/Desktop/DSA/hw4/hw4
592 + 3784 = 4376
```

b) **(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).

```
77  int main() {
78      addLargeNumbers("592", "3784");
79      addLargeNumbers("21309124087670231230213", "21841201231233112312231");
80      addLargeNumbers("9223372036854775809", "9223372036854775809");
81      addLargeNumbers("88223372036854775809", "12223372036854775809");
82      return 0;
83  }
```

```
~/Desktop/DSA/hw4 main* ❯ /Users/jeffylee/Desktop/DSA/hw4/hw4
592 + 3784 = 4376
21309124087670231230213 + 21841201231233112312231 = 43150325318903343542444
9223372036854775809 + 9223372036854775809 = 18446744073709551618
88223372036854775809 + 12223372036854775809 = 100446744073709551618
~/Desktop/DSA/hw4 main* ❯
```