3) (10 pts) DSN (Stacks)

You are playing a scoring game that uses a LIFO approach for keeping track of scores. Here are how the scores are managed. You are given a character array (string) of moves where each index represents some rule for managing the score. You must go through the array in index order to properly manage the score.

Here are the rules for managing the score:

- If the move is some character representing an integer (0-9 both inclusive), record the integer itself.
- If the move is the character '+'. You will need to retrieve the last 2 scores recorded and compute the sum. After computing the sum, you will need to add this sum to the recorded list of scores.

Once all moves have been processed, you will need to compute the total sum of all the scores and return this value. For example, if the string passed to the function is "25+3++1", then after processing the first plus sign the corresponding stack of values from bottom to top would be [2, 5, 7]. After processing the second plus sign the corresponding stack of values from bottom to top would be [2, 5, 7, 3, 10]. After processing the string completely, the stack would store [2, 5, 7, 3, 10, 13, 1]. The sum of these values, 41, should be returned. Complete the following function definition that simulates this scoring game. You may assume that the string header file is included. The provided functions and stack structure are here to assist you with completing this function. **Note: There exists a solution that doesn't use the stack and this or any such solution will get full credit, if correctly implemented.** The parameter represents the character array of moves, and is guaranteed to be valid. Namely, the string will consist solely of digits and plus signs, and if the string has any plus signs, they will only appear in an index 2 or greater (meaning that there will be two previous scores to add.)

This code is fairly long, so go ahead and write your code on the following page. The structs and functions you may use are listed on this page, below:

```
typedef struct node_s {
      int data;
      struct node_s * next;
} node_t;
typedef struct {
      node_t * top;
} stack_t;

// Initializes a stack to be empty.
void init(stact_t* s);

// Pushes data onto the stack pointed to by s.
void push(stack_t* s, int data);

// Removes and returns the integer at the top of the stack pointed to by s.
int pop(stack_t* s);

// Returns 1 if and only if the stack pointed to by s is empty. Returns 0
// otherwise.
int empty(stack_t* s);
```