CS212 – Lab 3 – Implementing a stack with an array

A. Implement the following functions for manipulating a stack (without actually declaring the stack – you will do that later), based on an array of characters of size STACKSZ, where STACKSZ is a #define symbol. The name of the stack is your choice. Pick a name. It will be declared in step B.
   a. Push – adds one element, a character, to the top of the stack (i.e.; it accepts a character as input) and returns an int:
      i. -1 if the stack is already full, if this occurs that character is lost
      ii. 0 if the item was successfully added
   b. Pop – removes the top element (if it exists) and returns:
      i. NULL if there are no elements (stack is empty)
      ii. The element itself if there was one
      iii. The next element of the stack becomes the new top of the stack
   c. Top – does NOT remove anything. Returns a char with the value:
      i. NULL if there are no elements (stack is empty)
      ii. The topmost element if there is one
   d. isFull – does NOT remove anything. Returns an int:
      i. -1 if there are no elements (the stack is empty)
      ii. 1 if the stack is full
      iii. 0 in all other cases
B. Using the above functions, write a program that:
   a. Declares the stack with a size of STACKSZ characters
   b. Uses your functions above to do the following:
      i. Accept input characters, 1 at a time until the symbol ^ is input
      ii. Push each character onto the stack
      iii. Output the stack (top to bottom – i.e.; LIFO) after all inputs have been received

Notes

1. For this, and future programs, you will add the statement:
   #include "tstdata.h"
   this file will contain the values of any global constants you may need.
   for example, for this program, it will contain:

   #define STACKSZ nnn

   Where nnn is the size of the stack at runtime. For testing your program yourself, you can create this file yourself in the same directory as your program.

2. Your main program should have NO access to the stack except through the above functions. The stack array & value of the index to the top ARE global, so all your functions can get at them, but the main code must never look at or use those values.
3. If the stack is full and more data keeps coming, your program should ignore the data and reply, "The stack is full. Enter the '^' character to stop.".
4. Repeat the above message if more input (other than "^" keeps coming).