Set 20 for SIZE as <const int>

// define structure

1. set structure called Stack
   i[SIZE] <int>
   top <int>
   END Stack

// define function prototypes

2. set function prototype **display_menu**(arguments: none) to display a menu.
3. set function prototype **push_item**(arguments: pointer Stack) that pushes a value onto stack.
4. set function prototype **pop_item**(arguments: pointer Stack) that removes a value from stack.
5. set function prototype **display_top**(arguments: pointer Stack) that displays the top value of stack.
6. set function prototype **check_empty**(arguments: pointer Stack) that checks if stack is empty
7. set function prototype **purge_stack**(arguments: pointer Stack) that destroys the stack.

//main

9. Set one stack variable **s**
10. Initialize s.top to -1 to create the stack.
11. Initialize an integer variable choice to zero. (choice for menu)
12. Set a char variable. (answer for another calculation)
13. call display_menu function.
14. Do
    ask the user for one of the above options using integer numbers and store the result in choice.
    While choice is less than one or choice is greater than six.
        display an error message.
        prompt the user for a valid option.
    END while.
    Switch statement.
        set case 1:
            call **push_item**(arguments: s).
        set case 2:
            call **pop_item** function (arguments: s).
        set case 3:
            call **display_top** function (arguments: s).
        set case 4:
            call **check_empty** function (arguments: s).
        set case 5:

call **purge_stack** function (arguments: s).
set case 6 to end the program.
END switch
Ask the user to perform another calculation
While answer is not Y, y, N, n
Display error message
Get input
END While
clear the screen.
while answer equals = y or Y.
END Do.

//function declarations

14. function **display_menu**(parameters: none)
display menu
End Function
15. function **push_stack** (parameters: pointer to Stack s)
Start Function.
declare an integer n
if the size of the stack is 19 (check if stack is full)
display stack is full
else
prompt user to enter a value n to be pushed
add the value to the stack
display the value added onto stack
End Function.
16. function **pop_item**(parameters: pointer to Stack s)
Start function.
if size of stack is -1 (check if stack is empty)
display stack is empty
else
display the top value that was deleted
End function.
17. function **display_top**(parameters: pointer to Stack s)
Start function.
if size of stack is -1 (check if stack is empty)
display stack is empty and no value can be displayed at top
else
display the value at the top of stack
End function.
18. function **check_empty**(parameters: pointer to Stack s)
Start function.

if size of stack is -1 (check if stack is empty)
display stack is empty
else
display stack is not empty
End function.

19. function **purge_stack**(parameters: pointer to Stack s)
Start function.
if size of stack is -1 (check if stack is empty)
display stack is empty and no values can be destroyed
else
set size of stack s to -1 (make it empty)
End function.