Programming Exercises 03
Stack Implementation

OBJECTIVE

To be able to write a program for the stack implementation and apply it in a program application.

INSTRUCTIONS

1. You are to work on this activity individually or by pair. There should only be one submission per person/pair.
2. You are to write a code for a program that follows the details in the PROGRAM SPECIFICATION section. There should only be one program for this PE.
3. You are to develop your program following the structured programming approach at the very least. No global variable declarations are allowed.
4. All your program files (source code files, input files, etc.) should be together with your main program file (the one that contains the main function). There should be no need to put certain files in certain folders just so your program will compile and run.
5. You only need to submit the source code file you have created. If you created more than one source code file (i.e. source code is broken down into several files) or there are other files (e.g. input files) involved, archive them in a single zip file. Name your source code file or zip file, whichever is applicable, using your surname followed by the PE # similar to this example: Rizal_03 (for individual) or Bonifacio_Rizal_03 (for pair/group).

PROGRAM SPECIFICATION

This program has two parts: stack implementation and stack application. You are to implement the stack first and then use that in implementing the program application.

Stack Implementation

Each cell of the stack should be able to hold a string. To hold the string, create an array of characters with big enough size.

The following are the expected operations for the stack:

- PUSH(*x, S*): inserts the item *x* at the top of the stack *S*
- POP(*S*): removes and returns the top element of the stack *S*
- FULL(*S*): returns true (non-zero value) if stack *S* is full, otherwise, returns false (0)
- EMPTY(*S*): returns true if stack *S* is empty, otherwise, returns false
- TOP(*S*): returns a copy of the top element of the stack *S*
- MAKENULL(*S*): makes the stack *S* empty

(Hint: For easier implementation of the stack, implement it using linked list and make it such that the first node is the top of your stack)

Stack Application

The following is the specification of the application you will implement using the stack you have created.

The application has the following main menu:

Kitchenware Washing Machine

[1] Add kitchenware
[2] Wash kitchenware
[3] Top kitchenware
[4] Wash all
[0] Exit

Enter choice:

Details of the menu items are as follows:

Add kitchenware
Once selected, the program should ask the user for a kitchenware (ex. Plate, Bowl, etc.) to be washed. The kitchenware should then be added to the stack of kitchenware to be washed.

If the stack is full, the program should not proceed with performing the process for the menu and instead display the following message:
“The STACK is FULL. Cannot add kitchenware.”

Wash kitchenware
Once selected, the program should proceed with the process of washing the kitchenware at the top of the stack. For this washing process, the kitchenware will be removed from the stack with a message following the format:
“[Kitchenware] is being washed.”

If the stack is empty, the program should not proceed with performing the process for the menu and instead display the following message:
“The STACK is EMPTY. No more kitchenware to wash.”

Top kitchenware
Once selected, the program should display what kitchenware is currently at the top of the stack. For the message, follow the format:
“[Kitchenware] is next to be washed.”

If the stack is empty, the program should not proceed with performing the process for the menu and instead display the following message:
“The STACK is EMPTY. No kitchenware to wash.”

Wash all
Once selected, the program should proceed with the process of washing all the kitchenware until the stack is empty. For each kitchenware being washed, display the same message from menu item [2]. After washing the last kitchenware, the program should inform the user that all the kitchenware has been washed.

If the stack is empty, the program should not proceed with performing the process for the menu and instead display the following message:
“The STACK is EMPTY. No kitchenware to wash.”

Exit
Once selected, the program should terminate with a message informing the user that the washing machine process has been terminated.

Note: After processing the selected main menu item, the program should loop back to the main menu. The program should end only when the user selects the main menu item ***Exit***.