

Distributed: March 18, 2010**Due:** April 8, 2010**Synopsis**

Write code to implement a stack in C. The stack must be implemented as a linked list. Each node in the stack contains a single char and a pointer to the next node.

Detailed information

Create a stack:

```
node * stack = null;
```

Write the following functions:

`push(char a);` pushes a single char to stack.

`char pop();` pops stack and returns the popped value. If stack is empty, return the char `'!'`. Do not exit.

`printStack();` steps through stack from top to bottom, printing out each char. Put a blank after each char in the output. The whole stack should go on one line of output.

`int isEmpty();` returns 1 if stack is empty, returns 0 if stack is not empty.

`void main ();` is the main function. After each push or pop instruction, print the entire stack.

```
    if (!isEmpty()) printStack();
    push ('a');
    push ('b');
    x = pop();
    push ('c');
    x = pop();
    x = pop();
    push ('d');
    if (!isEmpty()) printStack();
```

Declare all variables appropriately.

Thoroughly test your program so that at demo time, you will be able to write a new `main()` function with any order of stack operations.

Running the program

At the console, to run the C compiler:

`cc prog.c`

To execute the compiled program:

`a.out`

If `a.out` doesn't work, use `./a.out`

Turn in:

Hardcopy of code.

Screen shot for above order of push and pops.

Also, schedule a code demo and walk through.

Grade based on:

Meets specs: 80%

Coding style: 10%

Elegance: 10%

Extra credit: for an added 10%.

Use your stack routines to check for properly nested parentheses:

```
while (more input) {  
    if the next char is a '(', push it.  
    else if the next char is ')',  
        pop the stack;  
        if the popped char is a '(' continue,  
        else report error and quit  
    if the next char is anything else, ignore it and continue.  
}
```

At the end of the input, if the stack is empty report 'success', else report 'fail'.

Comment on development:

You might want to think about implementing the stack as a 1D array of chars first. Once you get that debugged and you understand what is happening, implement the final version using pointers.