3. Modify the `stackADT2.c` file of Section 19.4 so that it uses `assert` to test for errors instead of using `if` statements. (Note that the `terminate` function is no longer necessary and can be removed.)

Section 24.2 W 4. (a) Write a "wrapper" function named `try_math_fcn` that calls a math function (assumed to have a `double` argument and return a `double` value) and then checks whether the call succeeded. Here's how we might use `try_math_fcn`:

```
y = try_math_fcn(sqrt, x, "Error in call of sqrt");
```

If the call `sqrt(x)` is successful, `try_math_fcn` returns the value computed by `sqrt`. If the call fails, `try_math_fcn` calls `perror` to print the message `Error in call of sqrt`, then calls `exit` to terminate the program.

(b) Write a macro that has the same effect as `try_math_fcn` but builds the error message from the function's name:

```
y = TRY_MATH_FCN(sqrt, x);
```

If the call of `sqrt` fails, the message will be `Error in call of sqrt`. *Hint:* Have `TRY_MATH_FCN` call `try_math_fcn`.

Section 24.4 W 5. In the `inventory.c` program (see Section 16.3), the `main` function has a `for` loop that prompts the user to enter an operation code, reads the code, and then calls either `insert`, `search`, `update`, or `print`. Add a call of `setjmp` to `main` in such a way that a subsequent call of `longjmp` will return to the `for` loop. (After the `longjmp`, the user will be prompted for an operation code, and the program will continue normally.) `setjmp` will need a `jmp_buf` variable; where should it be declared?