Stacks and Queues

Due: 5 Mar 2008

Educational objectives:

- ***Primary objectives:*** Experience implementing simple stack and queue classes, using stacks to simulate recursion, using recursive function calls.

- ***Secondary objectives:*** Implementing and using templates.

Statement of work: (i) Implement generic *stack* and *queue* classes, and (ii) use your stack class to evaluate a recursive function.

Deliverables:

- Turn in a `makefile` and all header (*.h) and cpp (*.cpp) files that are needed to build your software, as described in www.cs.fsu.edu/~asriniva/courses/DS08/HWinstructions.html. Turn in your development log too, which should be a plain ASCII text file called `LOG.txt` in your project directory.

Requirements:

- Create a subdirectory called `proj4`.
- You will need to have a `makefile` in this directory. In addition, all the header and cpp files needed to build your software must be present here, as well as the `LOG.txt` file.
- You should create the following additional files.
  - *stack.h:* This should implement a generic `stack` class with at least the following features: (i) `void push(T &)`, (ii) `void pop()`, (iii) `T &top()`, and (iv) `bool empty()`.
  - *queue.h:* This should implement a generic `queue` class with at least the following features: (i) `void push(T &)`, (ii) `void pop()`, (iii) `T &front()`, and (iv) `bool empty()`.
  - *Other files:* You may use more files.
  - *main.cpp:* This is the main program. The command: `./recurse` N_1 C_1 A_1 M_1 M_2 M_3 M_4 D_1 D_2 S_1 S_2 `Arg Op` will cause the function given below to be evaluated in two different ways, and the two answers output. (The answers should be identical if your code is correct). The function should first be evaluated recursively, and next evaluated using a stack to simulate the recursion. The function is defined as follows.

    $f(N) = 0$, if $N < N_1$

    $f(N_1) = C_1$

    $f(N) = A_1 + M_1 * f(M_2 * N / D_1 - S_1)$ *Op* $M_3 * f(M_4 * N / D_2 - S_2)$, if $N > N_1$

Here, `f(Arg)` needs to be evaluated. N_1 C_1 A_1 M_1 M_2 M_3 M_4 D_1 D_2 S_1 S_2, Arg are integers and *Op* is either `+` or `-`. The division performed is the usual integer division with truncation.

Example:

```
Prompt> ./recurse 2 3 2 1 2 0 1 3 6 0 0 18 +

13 13
```
(program output)

Sample executable: A sample executable is available at ~cop4530/spring08/solutions/proj4/recurse on `linprog`. The first person to find errors in our program will get a bonus point!

Notes:

1. You should not use the STL `list`, `vector`, `deque`, `stack`, or `queue` classes. You may use the `string` class. Please get my permission before using any other STL feature.
2. We will test your `stack` and `queue` classes on entirely different applications. So it is important for these classes to be generic and exactly as specified.

Last modified: 19 Feb 2008