Stack: What problems does it solve?

- Stacks are used to avoid recursion, a stack can replace the implicit/actual stack of functions called recursively.

- Stacks are used to evaluate arithmetic expressions, to implement compilers, to implement interpreters:
  - The Java Virtual Machine (JVM) is a stack-based machine.
  - Postscript is a stack-based language.
  - Stacks are used to evaluate arithmetic expressions in many languages.

- Small set of operations: LIFO or last in is first out access:
  - Operations: push, pop, top, create, clear, size.
  - More in postscript, e.g., swap, dup, rotate, ...
Simple stack example

- `tstack` is a templated class, stores any type of value that can be assigned (like `tvector`)
  - Implemented simply using a vector, what does pop do?

```cpp
tstack<int> s;
s.push(2);
s.push(3);
s.push(1);
cout << s.size() << endl;
cout << s.top() << endl;
s.pop();
cout << s.top() << endl;
int val;
s.pop(val);
cout << val << endl;
```
Templated class, .h ok, .cpp ugly

- **See tstack.h for example**

  ```cpp
template <class Type>
  class tstack
  {
    public:
      tstack( );                   // construct empty stack
      const Type & top( ) const;   // return top element
      bool  isEmpty( ) const;     // return true iff empty
      int   size( ) const;        // # elements

      void push( const Type & item ); // push item
  }
  ```

- **But look at part of stack.cpp, class is templated (ugly?)**

  ```cpp
template <class Type>
  bool tstack<Type>::isEmpty() const
  {
    return myElements.size() == 0;
  }
  ```
Template class: implementation notes

- A templated function or class isn’t code, per se, but template (or pattern) for generating the “real” code
  - The templated class or function is instantiated when an object is created, or a function called
  - The template code is *instantiated* for a particular type
    - `tvector<int> a;  // creates code int vector`
    - `sort(a.begin(), a.end());  // create function`

- Since not really code, header declaration needs access to .cpp implementation at compile time
  - Typically use `#include “foo.cpp”` in `foo.h`, then client code gets both .h and .cpp
  - Ok because not code, otherwise would cause problems at link time with duplicate function/class definitions
Postfix, prefix, and infix notation

- Postfix notation used in some HP calculators
  - No parentheses needed, precedence rules still respected
  
  \[3 \ 5 \ + \ \ 4 \ 2 \ * \ 7 \ + \ 3 \ - \ \ 9 \ 7 \ + \ *\]

  - Read expression
    - For number/operand: push
    - For operator: pop, pop, operate, push

- See `postfix.cpp` for example code, key ideas:
  - Read character by character, check state of expression
  - Note: `putback` character on stream, only last one read

- What about prefix and infix notations, advantages?
Prefix notation in action

- Scheme/LISP and other functional languages tend to use a prefix notation

(define (square x) (* x x))

(define (expt b n)
  (if (= n 0)
      1
      (* b (expt b (- n 1)))))
Postfix notation in action

- Practical example of use of stack abstraction
- Put operator after operands in expression
  - Use stack to evaluate
    - operand: push onto stack
    - operator: pop operands push result
- PostScript is a stack language mostly used for printing
  - drawing an X with two equivalent sets of code

```plaintext
%! 200 200 moveto
100 100 rlineto
200 300 moveto
100 -100 rlineto
stroke showpage

%! 100 -100 200 300 100 100 200 200
moveto rlineto moveto rlineto
stroke showpage
```
Queue: another linear ADT

- **FIFO: first in, first out, used in many applications**
  - Scheduling jobs/processes on a computer
  - Tenting policy?
  - Computer simulations

- **Common operations (as used in tqueue.h/tqueue.cpp)**
  - Add to back, remove from front
    - Called `enqueue`, `dequeue`, like `s.push()` and `s.pop()`
    - Analog of `top()` is `front()`

- **Also used in level-order tree traversal, similar to pre-order without recursion but using stack**
  - See code in `treelevel.cpp`
Stack and Queue implementations

- Different implementations of queue (and stack) aren’t really interesting from an algorithmic standpoint
  - Complexity is the same, performance may change (why?)
  - Use vector or linked list, any sequential structure

- Linked list is easy for stack, where to add/remove nodes?

- Linked list is easy for queue, where to add/remove nodes?
  - Use circular linked list, why?

- Vector for queue is tricky, need ring buffer implementation, add but wrap-around if possible before growing
  - Tricky to get right (see tqueue.h, tqueue.cpp)
Reasoning about tree traversals

tqueue<
if (t != 0) q.enqueue(t);
while (! q.isEmpty()) {
    t = q.front();
    q.dequeue();
    if (t->left != 0)
        q.enqueue(t->left);
    if (t->right != 0)
        q.enqueue(t->right);
    cout << t->info << endl;
}  
tstack<
while (t != 0 ||
    ! s.isEmpty()) {
    while (t != 0) {
        s.push(t);
        t = t->left;
    }
    s.pop(t);
    cout << t->info << endl;
    t = t->right;
}
Using linear data structures

- **We’ve studied vectors, stacks, queues, which to use?**
  - It depends on the application
  - Vector is multipurpose, why not always use it?
    - Make it clear to programmer what’s being done
    - Other reasons?

- **Other linear ADTs exist**
  - List: add-to-front, add-to-back, insert anywhere, iterate
    - Alternative: create, head, tail, Lisp or Tapestry Clist<. . .>
    - Linked-list nodes are concrete implementation
  - Deque: add-to-front, add-to-back, random access
    - Why is this “better” than a vector?
    - How to implement?
Jaron Lanier is a computer scientist, composer, visual artist, and author. He coined the term ‘Virtual Reality’ ... he co-developed the first implementations of virtual reality applications in surgical simulation, vehicle interior prototyping, virtual sets for television production, and assorted other areas

"What's the difference between a bug and a variation or an imperfection? If you think about it, if you make a small change to a program, it can result in an enormous change in what the program does. If nature worked that way, the universe would crash all the time."

Lanier has no academic degrees