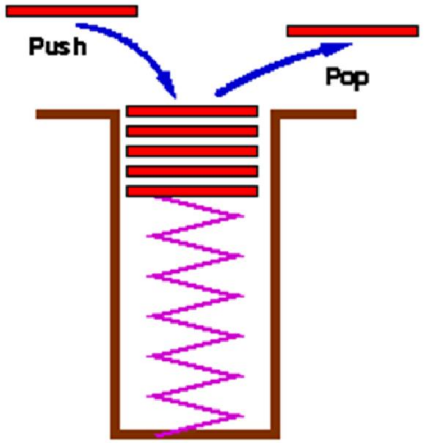
Data Structures and Algorithms

PLSD210(ii)

Stacks

Stacks

- **Stacks are a special form of collection with LIFO semantics**
- **Two methods**
  - `int push( Stack s, void *item );`
    - **add item to the top of the stack**
  - `void *pop( Stack s );`
    - **remove an item from the top of the stack**
- **Like a plate stacker**
- **Other methods**




```
int IsEmpty( Stack s );
/* Return TRUE if empty */
void *Top( Stack s );
/* Return the item at the top,
    without deleting it */
```

Stacks - Implementation

- **Arrays**
  - **Provide a stack capacity to the constructor**
  - **Flexibility limited *but* matches many real uses**
    - **Capacity limited by some constraint**
      - **Memory in your computer**
      - **Size of the plate stacker, etc**
- **`push, pop` methods**
  - **Variants of `AddToC…, DeleteFromC…`**
- **Linked list also possible**
- **Stack:**
  - ***basically a Collection with special semantics!***

Stacks - Relevance

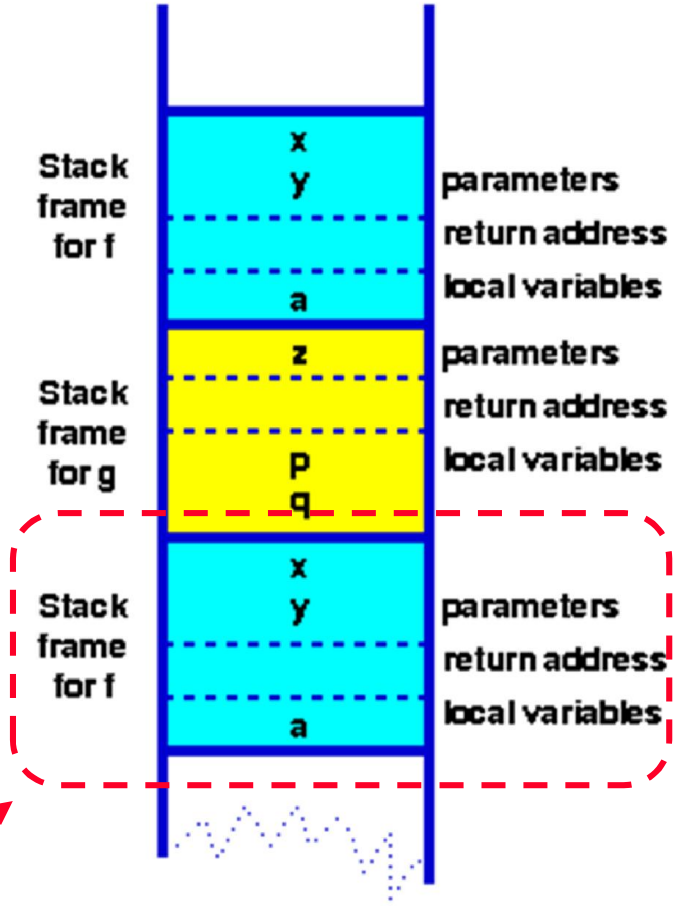
- Stacks appear in computer programs
  - Key to call / return in functions & procedures
  - Stack frame allows recursive calls
  - Call: push stack frame
  - Return: pop stack frame
- Stack frame
  - Function arguments
  - Return address
  - Local variables

Stack Frames - Functions in HLL

- **Program**

```
function f( int x, int y) {
    int a;
    if ( term_cond ) return …;
    a = ….;
    return g( a );
    }

function g( int z ) {
    int p, q;
    p = ….  ; q = ….  ;
    return f(p,q);
    }
```



Context for execution of `f`

Recursion

- **Very useful technique**
  - **Definition of mathematical functions**
  - **Definition of data structures**
    - **Recursive structures are naturally processed by recursive functions!**

Recursion

- *Very useful technique*
  - Definition of mathematical functions
  - Definition of data structures
    - Recursive structures are naturally processed by recursive functions!
- Recursively defined functions
  - factorial
  - Fibonacci
  - GCD by Euclid's algorithm
  - Fourier Transform
  - Games
    - Towers of Hanoi
    - Chess

Recursion - Example

- **Fibonacci Numbers**

Pseudo-code

```
fib( n ) = if ( n = 0 ) then 1
           else if ( n = 1 ) then 1
           else fib(n-1) + fib(n-2)
```

C

```
int fib( n ) {
     if ( n < 2 ) return 1;
     else return fib(n-1) + fib(n-2);
     }
```

Simple, elegant solution!

Recursion - Example

- **Fibonacci Numbers**

C

```
int fib( n ) {
        if (                    urn 1;
                        urn fib(n-1) + fib(n-2);
        }
```

But, in the Fibonacci case, a run-time disaster!!!!

However, many recursive functions,
eg* binary search, are simple, elegant *and efficient!