# CSI33 Data Structures

Sharon Persinger

Fall 2019

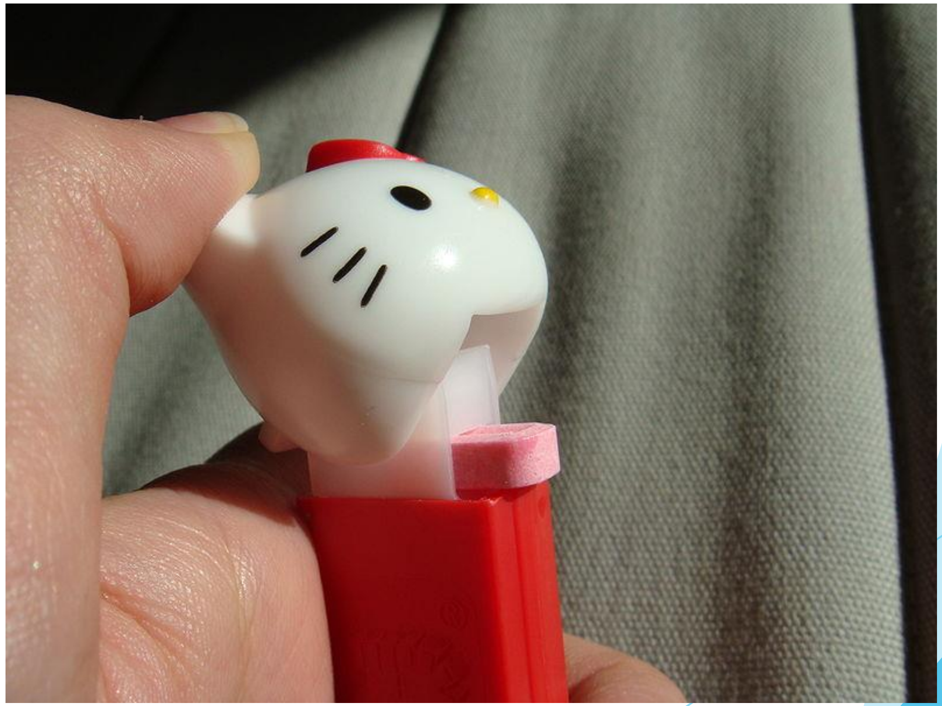
Day 9  September 25

## Stacks

Stack ADT

last in, first out data structure

LIFO



## Stacks

Stack ADT specification:

```
class Stack(object):
    #-------------------------------------------------
    def __init__(self):
    '''post: creates an empty LIFO stack'''
    #-------------------------------------------------
    def push(self, item):
    '''post: places x on top of the stack'''
    #-------------------------------------------------
def pop(self):
'''post: removes and returns the top element of the stack'''
    #-------------------------------------------------
def top(self):
 '''post: returns the top element of the stack without  removing it'''
    #-------------------------------------------------
def size(self):
'''post: returns the number of elements in the stack'''
```

## Stack implementation is straightforward

- Use a list for the items
- Use Python append function for push, and pop for pop.
- Use Python len function for size

## Application of stacks

- Undo function in software applications
- Computer operating systems -  managing function calls
  - Run-time stack – When a function is called, the values of local variables and the return address of the point at which the program was suspended are pushed on to the run-time stack.  When the function ends, this information is popped from the stack and the return address is used to find the location of the next instruction to execute.
- Analysis of syntax – mathematical expressions, programming expressions

## Evaluating arithmetic expressions

- Evaluate (7+2)*8-4+3*5
- You have to understand
  - Precedence of operations
  - Parentheses

## Reverse Polish or postfix notation

- Expresses arithmetic operations without parentheses
- Evaluate (7+2)*8-4+3*5  = 83
- 7 2 + 8 * 4 – 3 5 * +
- Use a stack and this procedure:
  1. If a number appears next in the expression, push this value on to the stack.
  2. If an operator appears next, pop two items from the top of the stack, place the operator between them, perform the operation, and push the result of the operation on to the stack.
  3. When the stack contains only one number, that is the result.

- Developed by the Polish mathematician Jan Lukasiewicz

## Changing infix notation to reverse Polish notation

- Assume the regular infix expression has a syntactically correct form.

- First break the regular infix expression into tokens – numbers, parentheses, operator symbols.

## Changing infix notation to reverse Polish notation

- Create an empty stack
- Create an empty list to represent the postfix expression
- For each token in the expression:
  - If token is a number:
    - Append it to the postfix expression
  - elif token is a left parenthesis:
    - Push it onto the stack
  - elif token is an operator:
    - while(stack is not empty and the top stack item is an operator with precedence greater that or equal to token):
      - Pop and append the operator onto the postfix expression
    - Push the token onto the stack
  - else token must be a right parenthesis
    - While the top item on the stack is not a left parenthesis:
      - Pop item from the stack and append it onto the postfix expression
    - Pop the left parenthesis
- While the stack is not empty:
  - Pop an item from the stack and append it onto the postfix expression