CSE 330 Jan 9th

* Student Assistant Email: [lazoa300@coyote.csusb.edu](mailto:lazoa300@coyote.csusb.edu)
  + Subject: CSE 330, Lab ^^, Monday
* Coding Stytle
  + Top of file
    - Your name
    - File name
    - Date
    - Problem description
    - Sketch of the solution
  + Comment description on every function
  + Class names must start with upper case letter; object and variables must start with lower case
  + Do not shortern names (identifies) as much as possible, IE "savings\_account" to "svngs\_acc"
  + Use underscores to separate words in a multi-word identifier
  + Indentication is four spaces!!!
* [Lab1](http://cse.csusb.edu/kay/cse330/lab1.html)
* A + b – c ---> a b + c -
* A + b \* c ---- a b c \* +
* A + b \* c – d ----> a b c \* + d -
* ( a + b ) \* (c – d) ----> ab + cd - \*
* A + (b \* (c – d)) ----> a b c d - \* +
* Stack (vector)
  + (-----) ---> (012345)
  + Push is going to the end and pop is off the top
  + Push () places a element on the top of the stack
  + Pop() takes the top element off and reveals element right under
  + Top() takes value off the top of the stack
  + #include <stack>
    - Stack<int> trays;
    - Trays.push(10);
    - 10 ------
    - Trays.push(25);
    - 10 25 ----
    - Trays.push(5);
    - 10 25 5 -----
    - To print top element
      * Cout << trays.top(); is 5
    - To remove the top
      * Trays.pop();
    - To print new top
      * Cout << trays.top(); is 25
    - Test if the stack is empty
      * If (!trays.empty())
        + Trays.pop();
  + To retrieve last of a vector (v[v.size()-1] or v.back()
  + Empty can apply to stack and vector
  + Push function for stack becomes push.back() for vector
  + Pop function for stack becomes pop.back() for vector
  + Top function for stack becomes back() for vector
  + Empty applies for both vector and stack
  + Stack is simply an adapter for a vector
* Lab 1 Analysis
  + Operators are
    - \* / are the highest precendence
    - + - are the lowser precendence
    - A + B – C
      * A gets printed right away
      * + goes into the stack
      * B gets printed right away
      * When – comes in because of equal precendence you pop the + and it prints then – goes into the stack
      * C gets printed right away
      * Then end of expression comes we empty the stack then we print and pop
      * A b + c -
    - A + B \* C
      * A gets printed
      * Read + and it goes into the stack because the stack is empty
      * B gets printed
      * \* gets read and because its higher it pushes to the top of the stack
      * C gets printed
      * Then end comes in and we pop and print and pop and print
      * A b c \* +
    - ) never goes into the stack
    - ( has the highest precendence outside. Always goes on top. Once it goes in, it has the lowest precedence inside
    - (a + b) \* (c – d)
      * ( goes into the stack right away
      * A gets printed
      * + reads and gets pushed on top of the stack
      * B gets printed
      * ) never goes into the stack. We pop until the ( comes in. So we pop the + out and then it matches the ( and they get canceled out.
      * IE so far it looks like a b +
      * Then \* gets read and gets pushed onto the stack
      * ( comes in to the stop of the stack
      * C gets printed
      * - gets read and it goes on top of the stack,
      * D gets printed
      * ) comes in. We pop the operators till the ( comes in.
      * IE a b + c d -
      * End comes in and we pop the stack. Being the \*
      * Answer is a b + c d - \*
  + Big oh notation
    - (?) function based on size of the problem
    - IE O(n) function loops on the size of N