CSE 12 Week Three Lecture Two
(Tuesday’s discussion: Q&A on hw3)

intopost example:
00 2 4 2 6 4 //
priorities
((27 + 452) * 2 – 16 ^ 2) / 7 <ret>

What:_________________

What:__________

stack 1

stack 2

eval example:

What:__________

What:__________

stack 1

stack 2

To represent a calculator word for an operator, you need all of the following all together in one long (word):
1. _________: to distinguish an operator from a number
2. _________: -> to display to the user
3. _________: -> used in “intopost” logic
4. _________: -> which function to execute
Inside eval, you will use the “functions” array to evaluate each part of the expression.

```c
long eval (…) {
    long op1, op2, word; // for the operands and operator
    …
    result = ________________________________;
    // word -> originated from a stack…it’s the operator
    …
}
```

You are NOT going to have a switch/case statement
```c
switch (operator) {
    case ‘+’: … NO!!!
    case ‘-‘: … NO!!!
    …
}
```

---

ungetting characters:

What: intopost and decin will need to put characters back into ____________ when they read a character not processed directly by the function.

Why: So that the character can be processed by the processing function. For example, __________ must read the entire number.

When:
- decin (provided) reads a __________ for __________ to process.
- intopost (you do) reads a __________ for __________ to process.
Obtaining the priority from the index:

<table>
<thead>
<tr>
<th>index</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(</td>
</tr>
<tr>
<td>1</td>
<td>)</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td>5</td>
<td>/</td>
</tr>
<tr>
<td>6</td>
<td>^</td>
</tr>
<tr>
<td>7</td>
<td>blank</td>
</tr>
<tr>
<td>8</td>
<td>!</td>
</tr>
</tbody>
</table>

The “setupword” function:
What:  Your constructor function to create the long to be used as the operator to be able to store on your stacks - the calculator word.
How:

`+` results in:
binary:

hexadecimal:
To get started with some code:

```c
long setupword (char character) {
}
```

Note: __________________ are magic numbers
- grader takes off
Better:
#define _____________________
#define _____________________

Accessor methods on the calculator word:
What: ________________, ________________,
___________ extract what ____________ creates.

Example use of ________ can be found in “eval” notes above.

Please keep in mind:

Only the following goes on the stacks in “intopost”:
- return values from:
  o __________
  o __________
  o __________
  o __________

NO EXCEPTIONS!!! If you forget, you will have errors!!
List object:
What: - a container object made up of ____________

Node objects:
What:
- the ____________ of a List
  - one Node per item _________________

List type:
head pointer: points to the ________________.
end pointer: points to the ________________.

Node type:
pre, next: points to the ________________.

Central theme to CSE 12 introduced in hw5
- Characteristic of all containers for the rest of the course
  - “________________________”

Polymorphic:
- The container is going to change its behavior based on
  ________________.

Generic:
- The container will be able to store ________________.
- The container will not know ________________.
- The object ________________ when inserted into a container.
- The object ________________ when removed from the container.

Ability:
- The container can ______________ the objects it holds without knowing
  ________________.

Constraints: __________________________________________
  - the constraints will differ in C, Java, C++
  - we will examine those constraints
    o Why: Without conforming, solution will ________.

What part of your program should allocate the objects to store in a generic container?
1. Driver program/main method allocates objects.
   a. Container stores ________________
or
2. Container allocates objects
   a. Container gets ________________
   b. Creates and stores ________________.
Efficiency:
What: A concept describing how good or bad a particular algorithm is.
Why: We want to save time and resources.
   - Time example: How long does it take for a computer to produce a result?
     o Real time systems have real-time requirements.
     o Users don’t want to wait too long for results.
   - Resource example: Customers billed by CPU cycle usage.

How do we measure:
   - Select an aspect of our algorithm to identify as a unit of work.
   - Then count it.

How do we compare efficiencies: Big-O (Wikipedia)
   - In computer science, big O notation is used to classify algorithms by how they respond (e.g., in their processing time or working space requirements) to changes in input size.
     - Not precise.
       o Constants are ignored
       o Lower order polynomial terms are ignored.

\[ O(3n^3 + 5n^2 + 45n + 2) = \] ____________

\( n = \) number of items in your data set

Best Efficiency: _______
What: Amount of work performed is ____________________
______________________________.