Goal: Call a method that assigns a variable (a long) to zero; that variable exists elsewhere:

```c
void func (__________) {
    __________;
}
main () {
    long lll;      // __________ variable
        func (__________);
}
```

<table>
<thead>
<tr>
<th>address</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

black is before, blue is after

Result: original “lp” in main is __________. Goal was __________.
Goal: Call a method that assigns a variable (a long pointer) to zero; that variable exists elsewhere:

```c
void func (__________) {
    __________;
}

main () {
    long * lp;       // __________ variable

    func (__________);  
}
```

<table>
<thead>
<tr>
<th>address</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

black is before, blue is after

Result: original “lp” in main is __________. Goal was __________.
Goal: Call a method that assigns a variable (a long pointer) to zero, that variable exists elsewhere:

```c
void func (__________) {
    __________;
}
main () {
    long * lp;
    func (__________);
}
```

<table>
<thead>
<tr>
<th>address</th>
<th>value</th>
<th>black is before, blue is after</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```c
void func (__________) {  
    __________;
}  
main () {  
    long * lp;
    func (__________);
}
```

<table>
<thead>
<tr>
<th>address</th>
<th>value</th>
<th>black is before, blue is after</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stack example:

A hw3 array-based stack of 10 items:

<table>
<thead>
<tr>
<th>count</th>
<th>size</th>
<th>sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

newStack (10):

Push 25

Push 50

Push 75

Top: sends back: 75

Pop: sends back: 75

Push 100

Push of 25 to a stack called mainStack in Java:

in C:
The delete_Stack method:

What: ________________________________

Responsibilities: ____________________________

In main: delete_Stack (&mainStack);

---

Function pointers:

What: ________ of a function

What: The address into the _________ section where the

__________________________

Example:

    long push (Stack * this_Stack, long item) { .... }

    void someFunc (Stack * this_Stack, long item) {
      long (*fp) (Stack *, long) = push;

      push (thisStack, item); /* call in hw3 */
      *fp (this_Stack, item); /* same as line above */
      fp (this_Stack, item); /* same as line above */
    }
How to read a declaration:

* pointer to …
[] array of …
() function returning …

Start reading the declaration at the name or the inner-most set of parenthesis. Loop of…
- reading forward until you have to go back:
  o grouping parenthesis
  o end of your declaration
- until you’ve accounted for each symbol.

Illegal declarations:
1. returning an array
   a. Why: _________________
2. return a function
   a. Why: _________________
3. an array of functions
   a. Why: _________________

returning a function pointer is __________ since they are __________.

NULL = 0
Matches __________ types in C and C++.
Data hiding in C:
In your .h files (Header files) is where you list _________ information.
In your .c files (Code) is where you list _________ or _________ information.

What goes in a .h file:

Ans:

_______________________________________________

Reasons:

1. ____________________________________________
2. ____________________________________________

Compiler Behavior

- What is a compiler?
  - Ans:
    o A programming tool that ________________________.
    o For us: C, Java, C++, to eventually get byte code, native machine code.

- What do we want a compiler to do for us?
  - Ans:
    o _________________
    o _________________

Compiler errors:

1) Lexical errors: Input is not recognized as legal in terms of basic symbols.
   - First errors the compiler identifies
     a. ______________________
     b. ______________________
     c. ______________________
     d. ______________________
     e. English: _______________
2) Parse errors or Syntax errors: Group of tokens is legal, but the order is unexpected.
   a. ______________________
   b. English: ________________

3) Semantic error: Structure is legal, but your statement doesn’t make sense.
   a. ______________________
   b. ______________________
   c. ______________________
   d. English: __________________

4) Warning: The compiler will do as you ask, but not necessarily as you want.
   - in CSE 12, we will compile without warnings
   a. ______________________
   b. ______________________
   c. ______________________
   d. ______________________
   e. English: __________________

5) Advice: All errors beyond the first are based on the compiler making a guess at what your meant:
   a. Guess was good: Error message is _____.
   b. Guess was wrong: Error message is ______.
   c. Therefore…correct the first few messages and compile again
   d. Therefore…don’t even look at 20th message

Run-Time Errors:
   What: Your program asked to do the impossible.
   Ex: if (10/0)…
   - access memory that is out of range.
   - result: _________ in C/C++, ___________ in Java
   - The “core” file:
     o A run-time image of your program in execution just before you asked to do the impossible.
     o Often very large files (consume entire quota)
     o What good is it?
       • The ____________ can read it
         o gdb a.out core
         o where … tell the line of problem.
Side note:
Desire to remove core file, xxx.o: student types (note the extra unwanted space after the * before the .o):
   rm core * .o
* is expanded by UNIX to match all files in current directory, and they will all be removed.
   make clean // safely remove core, exe, .class, .o
   make backup // copies source into a backup directory

Logic Errors: The program does what you ask, not what you want.
   - Most of your time will be spent here.
     Ex: Output is wrong.
     Ex: infinite loop.
     Ex: extra output of display.

Driver program:
What: Unit testing: Test each function/method individually

   hw1-2: driver1, driver2
   hw3: driver
   hw4: no driver (executable is calc, an end user application)
   hw5: driver1, driver2
   hw6-7: Driver1, Driver2
   hw8-9: Driver

Who is the end user of the driver/Driver programs?

Ans: ____________________

Therefore: All input doesn’t need to be checked, because ____________________.
Purpose: To test your methods before ____________________.

Therefore: The driver is disposable…you can throw it away once you are done testing (not in CSE 12).