Lab 6:

Goals

Understand how to construct an executable program from multiple source files. Demonstrate profiency in building functions that manipulate arrays. Understand the use of the *for loop* in processing arrays.

Background

Building large programs

- All large "real world" programs are built from multiple .c component files.
- One or more .h files must contain function prototypes for the functions that are defined in one module but referenced from another.
- Suppose some array manipulation functions are defined in module array.c and the main() functions that calls them is defined in module main.c. Then to build the a.out executable it suffices to enter the command:

```
gcc -g -Wall main.c array.c
```

Array inner products

- When two arrays have the same number of elements we can define an operation called the *inner product.*
- The inner product is the sum of the component-wise products.
- Suppose array a[] = {1, 4, 3, 5} and array b[] = {2, -1, 1, 4}.
- Then the inner product of the two arrays is:

  1 * 2 + 4 * -1 + 3 * 1 + 4 * 5 = 2 - 4 + 3 + 20 = 21

The *for* loop

```
for (init-expression; continue-condition; update-expression)
{
    loop-body
}
```

The *init-expression* is executed one time
The *continue-condition* is evaluated each iteration of the loop *before* the *loop-body* is executed.
The *update-expression* is executed after the *loop-body* is executed.
The *loop-body* is executed if and only if the *continue* condition is true.

Example:

The loop construct:

```
for (ndx = 0; ndx < count; ndx = ndx + 1)
{

     do - stuff

}
```

is 100% equivalent to:

```
ndx = 0;
while (ndx < count)
{
     do - stuff
     ndx = ndx + 1;
}
```

Assignment:

Create a file called lab6.c that contains two functions described below. Both functions *should* use the *for()* loop.

NEITHER FUNCTION SHOULD EVER CALL (f)scanf() or (f)printf()!!

```
int inner_product(
int a[],      // input array one
int b[],      // input array two
int count)    // number of values in both arrays
```

The inner_product() function should compute and *return* the inner product of the two arrays.

```
void reverse(
int a[],      // input array
int count)    // number of values in the array
```

The reverse() function should reverse the elements of the values in the array. If initial values in a[] are {4, 3, 7, 2}, then after reverse(a, 4) has been called the array should contain the values {2, 7, 3, 4}. Your reverse function *may not use an auxilliary third array*. The reverse() function must swap pairs of elements as described in class. The key to a correct solution is to identifying the pairs that must be swapped!!

Create a header file called *lab6.h* that contains prototypes for both functions.

Create a file called main.c that contains the main() function and use it to test your reverse() and inner_product() functions. You are free to either read in data or use statically initialized arrays in your tests. Your main() *will not be used* in evaluating your submission.

.

Turn In Work

Show your TA that you completed the assignment. Then turn in your lab6.c and lab6.h files using the command:

sendlab.101.*section_number* 6 lab6.c lab6.h