Nataly Garcia (811-251-176)
Madison Bardwell
Siddie Pennewill

1 Functions, Pointers, and Tricky Declarations Activity

1. The output of the code is:
   44
   22
   In the first function call, it is pass-by-value meaning a copy of the value is created. The second function call, or line 16, is a pass-by-reference.

2. &x is used because the address of x is being used as a parameter rather than the value. In the function itself, a pointer is pointed towards the address and uses the dereference operator to modify the value of x.

3.

Declaration	Meaning
int x;	x is an int
int * x;	x is a pointer to an int
char ** x;	x is a pointer to a pointer to a char
int * x [5];	x is an array of 5 pointers to ints
int (* x) [5];	x is an array of 5 pointers to arrays of 5 pointers to ints
int (* x [5]) [5];	x is an array of 5 pointers to an array of 5 ints
int * (* x [5]) [5];	x is an array of 5 pointers to an array of 5 pointers to ints
int x();	x is a function that returns an int
int x(int);	x is a function with an int parameter that returns an int
int * x();	x is a function that returns a pointer to an int
int * x(int *);	x is a function with a pointer to an int that returns a pointer to an int
int (* x)();	x is a pointer to a function with no parameters that returns an int
int ** (* x) (int **);	x is a pointer to a function with a pointer to a pointer to an int that returns a pointer to a pointer to an int

2 Const Pointers Activity

1. (a) Both statements of the code are valid.
   (b) The first statement is valid. The second statement is not valid because the int of the pointer is constant, meaning the value cannot be changed. The third statement is valid.

Nataly Garcia (811-251-176)
Madison Bardwell
Siddie Pennewill

(c) The first statement is valid. The second statement is invalid since the pointer is pointer to a constant int value. The third statement is also invalid because the pointer is constant.

2. (a) All statements of the code are valid.
   (b) The first two statements are valid. The third statement is not valid because it is a constant pointer, meaning the pointer cannot be moved.
   (c) The first statement is not valid because arrays cannot be declared to one another; the memory address of each variable are compared instead of the elements. The second statement is invalid because the a is a reference to an array, which cannot be changed. The third statement is valid.

3. (a) The first statement is valid. The second statement is invalid because the function foo returns a constant int. The third statement is invalid because for the same reason that foo returns a constant int.
   (b) The first three statements are valid. The fourth statement is invalid because it returns a constant int. The fifth statement is valid.
   (c) The first statement is valid. The second statement is invalid because the function returns a constant pointer. The third statement is invalid because it attempts to modify a constant value. The fourth statement is not valid because the function returns a constant pointer to an int. The fifth statement is valid.