What does the indirection operator do? -
The * (indirection) operator dereferences the value referred to by the pointer type operand, and it is used to access the value stored in an address.

With the following code, what would be displayed if you sent *iptr to cout?

int x = 7;
int *iptr = &x; -
address of x

What are the three different uses for the * operator? -
Multiplication operator, pointer definition, indirection operator

What math operations are allowed on pointers? -
addition and subtraction

Assuming ptr is a pointer to an int, what happens when you add 4 to ptr? -
The value 16 is added to the memory address of ptr.

True or false:
Assuming numbers[], *(numbers+3) is the same as numbers[3] -
True

What is the purpose of the new operator? -
allocate memory dynamically at runtime

What happens when a program uses the new operator to allocate a block of memory, but the amount of requested memory isn't available? How do programs written with older compilers handle this? -
If new fails, it throws bad_alloc exception. In older computers, new returns value 0

What is the purpose of the delete operator? -
deallocate the memory allocated by the new operator

What circumstances can you successfully return a pointer from a function? -
1. When a pointer variable is passed into a function as a parameter
2. A pointer is indicated to dynamically allocated memory

What is the difference between a pointer to a constant and a constant pointer? -
A pointer to constant may not be used to change the value it points to.

A constant pointer may not be changed after it has been initialized.

Each byte in memory is assigned a unique ____ -
Address

The ____ operator can be used to determine a variable's address. -
&

________ variables are designed to hold addresses -
Pointer

The ________ operator can be used to work with the variable the a pointer points to -
*

Array names can be used as ____, and vice versa. -
pointers

Creating variables while a program is running is called ____ -
memory allocation

The ______ operator is used to dynamically allocate memory -
new

Under older compilers, if the new operator cannot allocated the amount of memory requested, it returns _______. -
0 or NULL

A pointer that contains the address 0 is called a(n) _____ pointer. -
null

When a program is finished with a chunk of dynamically allocated memory, it should free it with the _______ operator. -
delete

You should only use pointers with delete that were previously used with _______. -
new

Look at the following code.
double value = 29.7;
double *ptr = &value;
Write a cout statement that uses the ptr variable to display the contents of the value variable. -
cout << *ptr;

Look at the following array definition:
int set[10];
Write a statement using ptr notation that stores the value 99 in set[7]; -
*(set + 7) = 99;

Assume that tempNumbers is a pointer that points to a dynamically allocated array. Write code that releases the memory used by the array. -
delete []tempNumbers;

Write the definition of ptr, a pointer to a constant int. -
const int value;
int *ptr = &value;

Write the definition of ptr, constant pointer to an int. -
int value;
int *const ptr = &value;

T/F: Each byte of memory is assigned a unique address -
true

T/F: The * operator is used to get the address of a variable. -
false, * is used to get the value at that address if specified to a pointer variable.

T/F: Pointer variables are designed to hold addresses. -
True, by definition they are designed to hold addresses.

T/F: The & symbol is called the indirection operator. -
False, the * is an indirection operator.

T/F: The & operator dereferences a pointer. -
False, as by definition of the address operator.

T/F: When the indirection operator is used with a pointer variable, you are actually working with the value the pointer is pointing to. -
True, by definition of the indirection operator.

T/F: Array names cannot be dereferenced with the indirection operator. -
False

T/F: When you add a value to a pointer, you are actually adding that number times the size of the data type referenced by the pointer. -
True, as pointers do not work like regular variables when used in mathematical statements.

T/F: The address operator is not needed to assign an array's address to a pointer. -
True

T/F: You can change the address that an array name points to. -
False, an array points to a memory location but these are constant pointers.

T/F: Any mathematical operation, including multiplication and division, may be performed on a pointer. -
False, pointer doesn't support all arithmetic operations.

T/F: Pointers may be compared using relational operators. -
True, relational operators can be used in comparing two pointers.

T/F: When used as function parameters, reference variables are much easier to work with than pointers. -
True, as reference variable acts as an alias to original variable used as an argument.

T/F: The new operator dynamically allocates memory. -
True by definition of new operator.

T/F: A pointer variable that has not been initialized is called a null pointer. -
False, as a pointer that contains the address 0 is called a null pointer.

T/F: The address 0 is generally considered unusable. -

False, by definition of null pointers.

T/F: In using a pointer with the delete operator, it is not necessary for the pointer to have been previously used with the new operator. -
False, as delete operator is used to free memory that was allocated with new.