Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

1. Recall that every variable in C++ has these four things: a name, a type, a value and a memory location. **1 / 1 point**

```
1   int *p;
2   p = new int;
3   *p = 0;
4
5
```

For the code above, which one of the following is NOT true for variable p?

- ○ The name of the variable is "p"
- ○ The type of the variable is a pointer to an integer, specifically the type "int *"
- ◉ The value of the variable is 0
- ○ The memory address of the variable is the value returned by the expression &p

Correct
Even though p points to a memory address containing the integer value 0, the value of the pointer is the memory address itself.

2. Which one of the following is true? **1 / 1 point**

- ◉ The address of any memory location in the stack is larger than the address of any memory location in the heap.
- ○ The "new" operator allocates memory on the stack that gets removed from the stack by the "delete"operator.
- ○ The C++ statement “int i;” allocates memory for one integer on the heap.
- ○ You should avoid using the memory address 0x0 for pointers whose value is not yet set, because memory location 0x0 is a valid location for the system to allocate to hold the contents of a variable.

Correct
The stack begins at a high memory address and works its way down, whereas the heap begins at a low memory address and works its way up.

3. Suppose we are writing the following function that is intended to return a pointer to a location in memory holding an integer value initialized to zero. **1 / 1 point**

```
1   int *allocate_an_integer() {
2       // declare variable i here
3       *i = 0;
4       return i;
5   }
6
7
```

How should variable i be declared?

- ○ int i;

- ○ int j;

  int *i = &j;
- ○ int *i;
- ◉ int *i = new int;

✓ **Correct**
The variable i should be a pointer to a memory location to an integer allocated from the heap so the memory location continues to be allocated after the function has returned.

4. Suppose we have this alternative function that returns a pointer to a memory location to an integer value of zero. **1 / 1 point**

```
1   int *allocate_an_integer() {
2       int i = 0;
3       return &i;
4   }
5
6   int main() {
7       int *j;
8       j = allocate_an_integer();
9       int k = *j;
10      return 0;
11  }
12
13
```

What value is variable k assigned and why?

- ○ Variable k is not assigned a value, because even if the compiler is set to ignore warnings and continue with compilation, the compiled program will still automatically detect that a local variable's address is being used after the function has returned, and exit to the operating system with a non-zero error code.
- ○ Assuming that the program compiles with just a warning and not an error due to the settings, the variable k will not be assigned a value, because the running program will crash the whole operating system.
- ◉ Unknown. Depending on the compiler settings, the compiler may report that a local variable address is being returned, which could be treated as a warning or as a compilation error; Or, if the program is allowed to compile, then at runtime the variable k could be assigned zero, or some other value, or the program may terminate due to a memory fault.
- ○ Variable k is certainly assigned the value zero, because the C++ runtime will automatically move the local variable to the heap and return the address of that heap variable instead.

✓ **Correct**
Correct. Variable i is allocated with a memory location on the stack. When allocate_an_integer() returns, its memory on the stack including the memory holding the value of i is freed and may be used for other purposes, and may be overwritten with a new value.

5. Suppose we declare a variable as "int i;" Which of the following expressions returns the address of the memory location containing the contents of variable i? **1 / 1 point**

- ○ *i
- ○ i.addr
- ○ i->addr
- ◉ &i

✓ **Correct**
The & operator returns the address of its operand.

6. **1 / 1 point**

```
1   int i = 0;
2   int *j = &i;
3
4
```

How many memory allocations are made on the stack and on the heap for the above code? For example, declaring an integer would count as one memory allocation.

- ○ One allocation on the stack and one allocation on the heap.
- ○ Zero allocations on the stack and one allocation on the heap.
- ◉ Two allocations on the stack and zero allocations on the heap.
- ○ One allocation on the stack and zero allocations on the heap.
- ○ Zero allocations on the stack and two allocations on the heap.

✓ **Correct**
Two allocations are made, with both to the stack: one for the integer i and one for the memory address that is the value of pointer j.

7. **1 / 1 point**

```
1   int *i = new int;
```

How many memory allocations are made on the stack and on the heap for the above code? For example, allocating space for one integer would count as one memory allocation.

- ○ One allocation on the stack and zero allocations on the heap.
- ◉ One allocation on the stack and one allocation on the heap.
- ○ Zero allocations on the stack and one allocation on the heap.
- ○ Two allocations on the stack and zero allocations on the heap.
- ○ Zero allocations on the stack and two allocations on the heap.

✓ **Correct**
The use of the new operator allocates memory on the heap that persists until it is deallocated by the delete operator, instead of on the stack which is deallocated when the current function returns.

8. **1 / 1 point**

```
1   int *i = new int;
2   *i = 0;
3   int &j = *i;
4   j++;
5
6
```

What does the last line of the above code segment do?

- ○ Increments the value of j by one, where the value of j is a local copy stored on the stack of the value of i stored on the heap.
- ○ Causes an error.
- ◉ Increments the value pointed to by variable i by one.
- ○ Increments the address pointed to by variable i by one.

Correct
Yes, j is a direct reference to the same actual integer that i points to indirectly.

9. **1 / 1 point**

```
1   int i = 0, j = 1;
2   int *ptr = &i;
3
4   i = 2;
5   *ptr = 3;
6   ptr = &j;
7   j = i;
8   *ptr = 4;
9
10
```

Enter the number of different values stored in the same address that variable i has during the execution of the code above. (Your answer should be a single integer, which is the total number of different values assigned to that address.)

3

Correct
The value of i is set to zero at line 1, then two at line 4, then three at line 5 because of line 2.

10. **1 / 1 point**

```
1   class Pair {
2       public: double a,b;
3   };
4
5   int main() {
6       Pair *p = new Pair;
7       p->a = 0.0;
8       return 0;
9   }
10
11
```

The expression p->a is equivalent to which one of the following?

- (*p).a
- *(p.a)
- p.*a
- p.a

Correct
The arrow operator -> accesses the right operand member in the class at the memory address of its left operand.