1. 2.26: The fractional part is truncated, but casting does not change the variable being cast.
2. 4.7: You use \\ and \” to display \ and “.
3. 4.9:

A

65

1000.34

1000

1000.34

1000

97

a

1. 1.38: Errors that are detected by the compiler are called syntax errors such as forgetting “;” at the end of line, or forgetting () for if condition.

Runtime errors cause a program to terminate abnormally. They occur while an application is running if the environment detects an operation that is impossible to carry out. Input mistakes typically cause runtime errors.

Logic errors occur when a program does not perform the way it was intended. Errors of this kind occur for many different reasons.

1. 3.10: If x=2 and y=3, the result is nothing

If x=3 and y=2, the result is “x is 3”.

If x=3 and y=3, the result is “z is 6”.

1. 3.17: Yes, they are equivalent.
2. 3.21:The results are:

0

0

1

1

1

1

1. 5.12: The output is :

sum is 14

count is 5

1. 5.15:

While loop:

long sum = 0;

while (sum != 1000)

sum = sum + 1;

Do-while loop:

long sum = 0;

do

sum = sum +1;

while (sum !=1000)

1. 5.16:

(a) n

(b) n + 1

(c) n – 5

(d) ceiling((n-5)/3)

1. 6.4: False. A value-returning function can also be invoked as statement in C++. In this case, the caller simply ignores the return value.
2. 6.9:

int function1(int n)

{

cout << n;

return 0;

}

void function2(int n, int m)

{

n += m;

function1(3); \\not 3.4

}

1. 6.12: Function overloading is that two functions have the same name but different parameter lists within one file. We can define two functions that have the same name but different parameter types. However, we cannot define two functions that have identical function names and parameter lists but different return types.
2. 6.27: You cannot overload functions that have identical parameter lists, but one is passing by value and the other one is passing by reference. That is an invalid function overloading.
3. 6.16: Inline functions are not called; rather, the compiler copies the function code in line at the point of each invocation. You can define a inline function with the keyword “inline” before the function definition.
4. 7.4:

True.

True.

True.

False.

1. 7.8: It will cause undefined behavior, which means anything could happens, but more likely, an error message should appear.
2. 7.11: False. Arrays are pointers. When an array is passed to a function, it is passing by reference, and there is no new array created.
3. 7.18: The first statement creates an array of characters of size 3 and has 3 elements ‘a’, ‘b’ and ‘c’.

The second statement creates a c-string, and has 4 elements ‘a’,’b’,’c’,and’\0’.

1. 11.9: P is a constant pointer and is initialized to have the address of x. So, you cannot change it to have the address of y later.
2. 11.10: P is pointing to an address with constant value. So, it causes error when you try to dereference
3. 11.11: 104 because int will occupy 4 space in memory.
4. 11.13:

\*p = 1

\*(p+1) = 2

p[0] = 1

p[1] = 2

1. 11.17:

i is 1

j is 2

k is 2

1. 11.15:

Dallas

D

a

l

l