Name: Jude Onyia

Student ID: V00947095

Course: ECE 596C

Due Date: June 5, 2020

Assignment 2: Non – Programming Exercise

8.1)

x?	lvalue
static_cast<std::vector<int>&&>(x)?	Rvalue (xvalue)
x.begin()?	Rvalue (xvalue)
++i?	lvalue
*i?	lvalue
*i += 5?	lvalue
x[0]?	lvalue
++a?	lvalue
a++?	Rvalue (prvalue)
func1(x)?	Rvalue (xvalue)
y = func1(x)?	lvalue

8.24)

Widget b(a);	Copy Constructor	Required with certainty (cannot be elided)
Widget c = a;	Copy Constructor	Required with certainty (cannot be elided)
Widget d(std::move(c));	Move Constructor	Required with certainty (cannot be elided)
Widget e = std::move(d);	Move Constructor	Required with certainty (cannot be elided)
Widget f(make_widget_1());	Move Constructor	Guaranteed to be elided
Widget g(make_widget_2(true));	Move Constructor	May be required (depending on if it can be elided)
c = a;	Copy Assignment	Required with certainty (cannot be elided)
b = std::move(c);	Move Assignment	Required with certainty (cannot be elided)
a = make_widget_1();	Move Assignment	Guaranteed to be elided
a = make_widget_2(true);	Move Assignment	May be required (depending on if it can be elided)
func_1(a);	Copy Constructor	Required with certainty (cannot be elided)
func_1(std::move(a));	Move Constructor	Required with certainty (cannot be elided)
func_1(make_widget_1());	Move Constructor	Guaranteed to be elided
func_2(std::move(b));	Move Constructor	Required with certainty (cannot be elided)

8.25)

Lines marked with ???:	Line of code where a temporary object is created:	Description and explanation:
Line 67: z = x + y;	*Line 53:* counter(x)	This temporary object is created within operator+. It is needed to store the result of x+=y without changing the value of x.
	Line 67: x+y	This temporary object is created in the main function. It is required in order to hold the return of operator+ before it can be assigned to counter z.
Line 68: z = z + z;	*Line 53:* counter(x)	This temporary object is created within operator+. It is needed to store the result of x+=y without changing the value of x.
	Line 68: z+z	This temporary object is created in the main function. It is required in order to hold the return of operator+ before it can be assigned to counter z.
Line 69: y = ++z;	N/A	This line does not need a temporary object because the pre-fixed operator++ returns an lvalue reference to the counter z.
Line 70: z = y++;	*Line 70:* y++	This temporary object is created in the main function. It is required in order to hold the return value of operator++(int) before it can be assigned to counter z.
Line 71: x = z;	N/A	This line does not need a temporary object because the operator= returns an lvalue reference.

8.26)

Some advantages of array-based implementation of a stack are elements stored contiguously in memory, less overhead, and it is more cache friendly. The disadvantage of array-based implementation is it cannot guarantee that each push will take constant time. This is due to the situation where the capacity of the array is full, in this case it would have to copy its entire content to a bigger section of memory before it can push that object.

An advantage of node-based implementation of a stack is that the previously discussed capacity exceeding problem is not evident here. This guarantees the push operation will be done in constant time. Another advantage is that references to objects in the stack are stable (always valid) since the objects they refer to are not forced to be copied or moved somewhere else. Disadvantages of node-based implementation include not storing elements contiguously in memory, per-element overhead, and requires more space than array-based implementation due to the overhead.