1. Write a class named Average that will keep track of the average of all integers passed to it. Use two members: The first one should be type std::int\_least32\_t, and used to keep track of the sum of all the numbers you’ve seen so far. The second should be of type std::int\_least8\_t, and used to keep track of how many numbers you’ve seen so far. You can divide them to find your average.

2a) Write all of the functions necessary for the following program to run:

int main()

{

Average avg{};

avg += 4;

std::cout << avg << '\n'; // 4 / 1 = 4

avg += 8;

std::cout << avg << '\n'; // (4 + 8) / 2 = 6

avg += 24;

std::cout << avg << '\n'; // (4 + 8 + 24) / 3 = 12

avg += -10;

std::cout << avg << '\n'; // (4 + 8 + 24 - 10) / 4 = 6.5

(avg += 6) += 10; // 2 calls chained together

std::cout << avg << '\n'; // (4 + 8 + 24 - 10 + 6 + 10) / 6 = 7

Average copy{ avg };

std::cout << copy << '\n';

return 0;

}

COPY

and produce the result:

4

6

12

6.5

7

7

Hint: Remember that 8 bit integers are usually chars, so std::cout treats them accordingly.

[Show Solution](javascript:void(0))

2b) Does this class need an explicit copy constructor or assignment operator?

[Show Solution](javascript:void(0))

1. Write your own integer array class named IntArray from scratch (do not use std::array or std::vector). Users should pass in the size of the array when it is created, and the array should be dynamically allocated. Use assert statements to guard against bad data. Create any constructors or overloaded operators needed to make the following program operate correctly:

#include <iostream>

IntArray fillArray()

{

IntArray a(5);

a[0] = 5;

a[1] = 8;

a[2] = 2;

a[3] = 3;

a[4] = 6;

return a;

}

int main()

{

IntArray a{ fillArray() };

std::cout << a << '\n';

auto& ref{ a }; // we're using this reference to avoid compiler self-assignment errors

a = ref;

IntArray b(1);

b = a;

std::cout << b << '\n';

return 0;

}

COPY

This program should print:

5 8 2 3 6

5 8 2 3 6

[Show Solution](javascript:void(0))