What's new in C++ exercises: ( [gonenni@yahoo.com](mailto:gonenni@yahoo.com) )

1) make the changes needed to make the following code compile:

class Coord

{

int x;

int y;

};

void demo()

{

vector<Coord> v1 = { {0,0},{1,1} };

//cout << v1[0].getX();

}

1b) Create a class called MyData that has 1 member called data, which is an array of ints

Create global or member begin() and end() functions for your class

Create an instance of MyData called myData

Use a range based loop (similar to for (int x: y) ) to iterate over the data in myData without explicitly using its ‘data’ member from main

2) Given

template<class MyType> void myswap(MyType& left, MyType& right )

{

}

Implement the function so that it does not use another occurence of the word

MyType, and without using the standard library

2b) Given

struct Struct1 { int a; int b; };

struct Struct2 { const char \*a; const char\* b; };

template<class MyType> void swapMembers(MyType& value)

{

}

int main()

{

Struct1 s1 = { 5, 6 };

Struct2 s2 = { "Five", "Six" };

swapMembers (s1);

swapMembers (s2);

std::cout << "s1 a = " << s1.a << " b = " << s1.b << std::endl;

std::cout << "s2 a = " << s2.a << " b = " << s2.b << std::endl;

}

SwapMembers is supposed to swap the values of the members a and b.

Implement the swapMembers function so that it workds for Struct1 and for Struct2

3) Given

#include <iostream>

struct A

{

void start()

{

std::cout << "A.Start\n";

}

};

struct B

{

void start()

{

std::cout << "B.Start\n";

}

};

#ifdef SYSTEM1

A createObject()

{

return A();

}

#else

B createObject()

{

return B();

}

#endif

#ifdef SYSTEM1

void doStart(A objToStart)

{

objToStart.start();

}

#else

void doStart(B objToStart)

{

objToStart.start();

}

#endif

void startObject()

{

#ifdef SYSTEM1

A temp = createObject();

doStart(temp);

#else

B temp = createObject();

doStart(temp);

#endif

}

int main()

{

startObject();

}

Rewrite the code so you can remove the #ifndef-#else-#endif in the startOBject function,

and also remove the #ifndef-#else-#endif from the doStart functions, so you have just one doStart.

3b)

template<class MyType, class MyType2 > void add(MyType a, MyType2 b, ??? sum)

{

sum = a + b;

}

#include <iostream>

int main()

{

int sum1;

double sum2;

add(5, 3, sum1);

add(5, 3.9, sum2);

add(3.9, 5, sum2);

}

Complete the code so that it is correct for both sum1 and sum2

4) Given :

class A{};

class B : public A {};

template <typename T1, typename T2>

class ParentChildPair

{

T1 parent;

T2 child;

};

Write the ParentChildPair class so it only compiles when the first type

is a parent of the second type

ParentChildPair<A, B> pair; // Should compile because A is a parent of B

ParentChildPair<B, A> pair2; // Should not compile because A is a parent of B

5) Create a C-Style array with a fixed size that is the 6th fibonnacci number

( but don't use a hard coded value ;)

6) Create a class Person that has string name, int id

Create a constructor for Person that accepts name and id

Create a class Employee that extends it and adds string employer

without adding any constructors to Employee, make the following line compile:

Employee e("Gonen",1);

Next, add a default constructor to Person, without writing it

without adding any constructors to Employee, make the following line NOT compile:

Employee e2;

7) Given

int numbers[] = { 5,3,1,100,102 };

Use find\_if with a lambda to find and print the first even number in the list

8) Write a function that accepts a vector of numbers and a lambda ( or function pointer ) and returns the result of running the lambda on the vector

call it once with a lambda that calculates the average of the numbers

call it once with a lambda that calculates the sum of the numbers

8b) Write a function called my\_foreach that accepts a vector of numbers and a lambda that accepts one int.

make my\_foreach call the lambda on each member of the passed vector

in main, define a vector of numbers data={1,2,3,4,5}

1. from main call my\_foreach with the vectors and a lambda that will make foreach display the numbers with a comma as follows: 1,2,3,4,5

2. from main call my\_foreach with the vectors and a lambda that will make foreach display the numbers with a '|' as follows: 1|2|3|4|5

3. declare a global variable int sum = 0;

from main call my\_foreach with a lambda that will store the sum of all the vector's numbers in the variable sum

( check your results ofcourse )

9) In main create local variables a,b, and sum

define and execute a local lambda that puts the sum of a+b into sum

9b) Try to modify my\_foreach from question 8b) so that you can change sum from a global variable to a local variable, and still get the right sum.

10) For each of the following say if they are Lvalue or RValue references

Try first without coding.

If you want to verify, you can test your hypothesis in code

X foo() { X x(5, "Joe"); return x; }

int a(7);

int b = ++a; // what is ++a ?

int c = b--; // what is b-- ?

X x = X(6, "Jane"); // what is X(6, "Jane") ?

int\* v = new int[10];

int d = v[7]; // what is v[7] ?

int e = a+b; // what is a+b ?

X f = foo(); // what is foo() ?

11) Create a class called Val which is a simple wrapper for an int

Give the class an int datamember called x

Give the class a constructor which accepts an int and puts it in x

Give the class a copy constructor

Create an operator+ which supports addition between two vals

Val a1(10);

Val a2(2);

Val a3 = a1 + a2;

cout << " Value of a3.x = " << a3.x; // should print 12

Create a direct reference to a1+a2 and change the value of x of the temporary

e.g. ( will (a1+a2).x = 9 work ? )

12) Create a class called MemBuf

Give the class a member called buffer which is an array of 1000 int

Create a constructor for the class which accepts an int value, and puts it at buffer[0]

Create copy and move constructors and assignment operators for the class

Create a global function ( not in MemBuf ) called createMemBuf which creates and returns a MemBuf with the value 42 at buffer[0]

In main create a MemBuf variable called memBuf

and use the createMemBuf to initialize/assign it

Run your code and make sure that the move constructor or move assignment were called

if they were not called , get them to run.

13) Create a global int counter called globalCounter

Have main create a thread that increments globalCounter 100000 times with a tight for loop

Have main also increment globalCounter with a tight for 100000 loop

print globalCounter at the end of main, and see that it is not 200000

Add a global mutex called mymutex

Before the increments, Add thread synchronization using

std::lock\_guard<std::mutex> guard(mymutex)

Check that the final result of globalCounter is now 200000 as it should be

14) Write a template function called minElement

Implement the function to use a Variadic Args template expansion to find the smallest element in compile time

Check that your function works

See if you can add constexpr to your function to allocate an array as follows:

int buffer[minElement(10,20,5,100)];

15) Create a tuple variable called myTuple with <int,double,int \*>

populate your tuple with values

create a lambda funciton that accepts a myTuple, swaps the values of the int and double, and returns it

call your lambda on myTuple and see that it switched the values of the int and double

16) Create a class called MyResource with one member

int buffer[1000];

Give the class a constructor and destructor that log their activity to screen

Create a shared pointer ptr1 around a new instance of MyResource

Run the program and check that MyResource is created and then destructed

Create a shared\_ptr ptr2, but don't assign it anything

what happens if you try to print ptr2->buffer ?

Now Assign ptr2=ptr1

Check that changes in ptr->buffer are reflected in ptr2->buffer

Check that when ptr2 and ptr1 go out of scope, the MyResource destructor is called

17) Write the function myPrint so that the following code works

int numbers[] = { 1, 1, 2, 3, 5, 8 }; // works with c-style arrays!

vector<int> numbers2 = { 1,2,3,4 }; // will also work with vector

myPrint(numbers);

myPrint(numbers2);

\* hint use templates