Object lifetimes QUIZ

Attempt the questions before you look at the answers at the end.

1.

What do you see wrong in the code below?

```
class MyClassA
{
public:
            MyClassA (MyClassB  b);

private:
            MyClassB &  m_B;
}
MyClassA::MyClassA( MyClassB  b )
: m_B ( b )
{
}
```

2.

What do you see wrong in the code below?

What is the best way to fix it?

```
class MyClassA
{
public:
      MyClassA( int value );

private:
      int m_Value1;
      int m_Value2;

};

MyClass::MyClass ( int value )
: m_Value2 ( value ),
  m_Value1 ( m_Value2 )
{ }
```

3.
What do you see wrong in the code below?

```
int * arr = new int [ 100 ];

// use arr

delete arr;
```

4.

What do you see wrong in the code below?

```
class MyClassA
{
public:

   MyClassA( int i );          // conversion constructor
   MyClassA ( MyClassA  a );   // copy constructor

};
```

5.

Assume we have the following classes:

```
class Parent
{
       Parent(int x);
       ~Parent();

  private:
        int    m_Value;
};

Parent::Parent(int x)
 : m_Value(x)
{}
Parent::~Parent()
{}

class Child : public Parent
{

public:
       Child(int a);
       ~Child();
private:
       int  m_Num;
};

Child::Child (int a)
```

```
 : m_Num(a)
{
}

Child::~Child()
{}
```

What is wrong with the above code ?

How would you fix it?

ANSWERS

1.

What do you see wrong in the code below?

```
class MyClassA
{
public:
            MyClassA (MyClassB  b);

private:
            MyClassB &  m_B;
}
MyClassA::MyClassA( MyClassB  b )
: m_B ( b )
{
}
```

m_B is a reference data member, and the constructor of MyClassA takes MyClassB by value, which means that m_B reference gets initialized to a temporary copy of b passed in to the constructor. This will leave m_B pointing (referring) to an object that gets deleted when the constructor is done executing.

Fix would be to take the MyClassB parameter by reference instead of by value, so , something like:

```
MyClassA (MyClassB &  b);
```

2.

What do you see wrong in the code below?

What is the best way to fix it?

```
class MyClassA
```

```
{
public:
      MyClassA( int value );

private:
      int m_Value1;
      int m_Value2;

};

MyClass::MyClass ( int value )
: m_Value2 ( value ),
  m_Value1 ( m_Value2 )
{ }
```

The initialization list depends on m_Value2 being initialized BEFORE m_Value1. This is in general not a good practice.

One fix is to do something like:

```
MyClass::MyClass ( int value )
: m_Value1 ( value ),
  m_Value2 ( m_Value1 )
{ }
```

But a better fix would be:

```
MyClass::MyClass ( int value )
: m_Value2 ( value ),
  m_Value1 ( value )
{ }
```

3.
What do you see wrong in the code below?

```
int * arr = new int [ 100 ];
// use arr
delete arr;
```

The delete operator called is not right… it should be delete [] arr;

Remember, if you called new, then call delete, if you called new [], you have to pair that up with delete []

4.

What do you see wrong in the code below?

```
class MyClassA
{
public:

   MyClassA( int i );          // conversion constructor
   MyClassA ( MyClassA  other );  // copy constructor

};
```

A copy constructor cannot take the 'other' object by value. It has to be passed by reference, and preferably const reference. You can also define a copy constructor to take its argument by address, but const reference is the convention.

Now, why can we not pass the argument to a copy constructor by value? Its because passing by value creates a copy of the object to be passed in, which then invokes the copy constructor, which then tries to create another copy, and as you can see, this leads to infinite recursion, crashing the program as it will eventually run out of stack space.

5.

Assume we have the following classes:

```
class Parent
{
       Parent(int x);
       ~Parent();

  private:
        int    m_Value;
};

Parent::Parent(int x)
 : m_Value(x)
{}
Parent::~Parent()
{}

class Child : public Parent
{

public:
       Child(int a);
       ~Child();
private:
       int  m_Num;
};

Child::Child (int a)
 : m_Num(a)
```

```
{
}

Child::~Child()
{}
```

What is wrong with the above code ?

How would you fix it?

Child constructor does not call the Parent constructor, which means the compiler will try to generate code to call the default constructor for Parent. However, there is no default Parent constructor defined, and the compiler will not automatically give us a default Parent constructor (Why?), so this will give a compilation error. The fix would be to either call the Parent's conversion constructor from the Child constructor, or add a default Parent constructor.