Phoenix Integration  
Interview Questionnaire

# Part 1: Programming Exercises

Unless otherwise specified, the following may be completed in C++, Java, or C#, as you prefer. Unless otherwise specified, programs may be command-line or GUI-based. Please use appropriate naming and coding conventions for the language you choose.

We have not completely specified the requirements for all of these problems; we expect you to make reasonable assumptions and document them.

Please spend no more than three hours on Part 1.

## Problem 1.1 – Quadratic Formula

Write a program that allows the user to input the coefficients of a quadratic formula; i.e. the values *a*, *b*, and *c* in the formula *ax*2 + *bx* + *c* = 0. The program should then display the roots of the quadratic equation.

Please put the code that computes the roots from the coefficients in a separate function that could be re-used in another program.

## Problem 1.2 – Primes

This problem should be done in either Java or C# and assumes a GUI environment (Swing or WinForms, respectively).

Write a class that calculates prime numbers starting at 2. The class should have a public interface with two methods and an event:

* A “start” method – this will be called from the main GUI thread and cannot block. It will begin the calculation of prime numbers.
* A “stop” method – again, this will be called from the main thread, and *may* block until prime calculation stops (your discretion).
* An event that is fired when a new prime number is calculated. The event must be fired on the main GUI thread. The event data should contain at least the current collection of prime numbers, in a list (i.e. a **List<Integer>** in Java or **IList<int>** in C#). Use language- and framework-appropriate event semantics.

## Problem 1.3 – Class Interface

This problem should be done in C++.

Write a header (.h) file for class representing an aircraft – this will serve as the base class for a polymorphic library of aircraft classes representing specific models of aircraft. You do not need to implement any methods, though you may if they are obviously inline methods.

The class should store the following values:

* The physical geometry of the aircraft specified by a collection of geometry structs. (assume a “Geometry” struct already exists).
* The dry weight of the aircraft.
* The fuel capacity of the aircraft.
* The cargo capacity of the aircraft.
* The max airspeed of the aircraft.
* The make and model names.
* The current pilot and co-pilot’s data (assume a “Pilot” data structure; may be none).

The class should support the following methods:

* Calculate travel time for a particular distance given an initial cargo and fuel load.
* Calculate the minimum airspeed at a given altitude and angle of attack before the plane will stall.

## Problem 1.4 – Unicode

Write a function in C++ program that converts a Unicode string encoded in UTF-8 (a std::string) to UTF-16 (a std::wstring). Assume a Windows environment (std::wstring is UTF-32 on most Unix platforms).

## Problem 1.5 – Code Translation

Consider the following code fragments in C# and Java; both do the same thing. Write equivalent code in C++.

### C#:

// if this is a square, paint it

Square sq = shape as Square;

if (sq != null)

{  
 // get the current graphics caret position

double x = caret.X;

double y = caret.Y;

// always draw with a solid brush; we’ll save the old one just in case

Brush oldBrush = sq.PaintBrush;

if (!(oldBrush is SolidBrush))

{

// create a new solid brush with the default color

sq.PaintBrush = (Brush)graphicsFactory.CreateObject(“SolidBrush”, SQUARE\_COLOR);

}

// draw the sqaure

sq.Draw((int)x, (int)y);

// restore the old paintbrush

sq.PaintBrush = oldBrush;

}

### Java:

// if this is a square, paint it

if (shape instanceof Square)

{

Square sq = (Square)shape;

// get the current graphics caret position

double x = caret.getX();

double y = caret.getY();

// always draw with a solid brush; we’ll save the old one just in case

Brush oldBrush = sq.getPaintBrush();

if (!(oldBrush instanceof SolidBrush))

{

// create a new solid brush with the default color

sq.setPaintBrush((Brush)graphicsFactory.createObject(“SolidBrush”, SQUARE\_COLOR));

}

// draw the sqaure

sq.draw((int)x, (int)y);

// restore the old paintbrush

sq.setPaintBrush(oldBrush);

}

# Part 2: Code Review

In each of the following examples, identify as many flaws, problems, and bugs as possible.

## Example 2.1:

The following code is Java:

/\*\*

\* Read all the data from the connected source into a {@link StringBuilder}.

\* @param sb the {@link StringBuilder} to add data to

\* @return the number of characters read

\*/

public int readSourceData(StringBuilder sb)

{

int numRead = 0;

String line;

try

{

// readLine() returns null when end of source is reached

while ((line = connectedSource.readLine()) != null)

{

numRead = line.length();

sb.append(line);

}

}

catch (IOException ex)

{

throw new IllegalStateException(“read from source failed”);

}

return numRead;

}

## Example 2.2:

The following code is C++:

// write the buffered data out to the file

std::vector<char>& data = bigBuffer.getData();

for (unsigned long idx = 0; idx < data.size(); ++idx)

{

fileStream << data[idx];

}

## Example 2.3:

The following code is C#:

/// <summary>Move a point by an offset.</summary>

/// <param name=“point”>the point to offset</param>

/// <param name=“offset”>the offset to move the point by</param>

public void OffsetPoint(System.Drawing.Point point, System.Drawing.Size offset)

{

point.X += offset.Width;

point.Y += offset.Height;

}

## Example 2.4:

The following code is C++:

// convert a string from mixed to lower case (Unicode-aware)

std::wstring toLowerCase(const std::wstring& toConvert);

// load a previously saved file

bool loadFile(const std::wstring& toLoad);

…

// we store filenames as lower-case because the OS is not case-sensitive

const wchar\_t\* lower = toLowerCase(filename).c\_str();

// look up the save file

SaveFile\* f = recentFiles.lookup(lower);

assert(f != nullptr);

assert(loadFile(f->getFullPath()));

## Example 2.5:

The following code is C++:

// print the string in reverse order

void reversePrint(const wchar\_t\* source)

{

for (auto idx = wcslen(source) ; idx >= 0 ; --idx)

{

print(source[idx]);

}

}

## Example 2.6:

The following code is C#:

/// <summary>Convert an array of double values to our portable XML format so

/// they can be saved in a file for later use.</summary>

/// <param name=“document”>The XML document to generate elements from</param>

/// <param name=“values”>The array containing double values.</param>

public static XmlElement doubleArrayToXml(XmlDocument doc, double[] values)

{

XmlElement arrayEl = doc.CreateElement(“array”);

arrayEl.SetAttribute(“type”, “double”);

foreach (double d in values)

{

XmlElement valueEl = doc.CreateElement(“value”);

valueEl.InnerText = d.ToString();

}

return arrayEl;

}

## Example 2.7:

The following example is C++:

// populate the list with complete structures created from the blueprints,

// up to the cost limit; we’ll clean them up when we’re done with the list

std::vector<Structure\*> structures;

long totalCost = 0;

for (auto it = blueprints.begin(); it != blueprints.end(); ++it)

{

Structure\* s = new Structure(\*it);

if (!s->isComplete())

{

continue;

}

totalCost += s->getCost();

if (totalCost > costLimit)

{

break;

}

structures.push\_back(s);

}

## Example 2.8:

The following example is C++:

// we’ve got a failure - better store the info and alert the user

ErrorInfo\* pInfo = new ErrorInfo(errorType, errorMessage);

while (operationIsRunning())

{

// wait for the operation to abort

}

// pass the error message to the main thread for processing

PostMessage(hMainWnd, WM\_USER, (WPARAM)OperationError, (LPARAM)pInfo);

// make sure we free the memory we allocated

delete pInfo;