Solution for Prelim 1 – Fall 1999

Problem 1. (20 points)

(a) class F extends D implements A { ...}

(b) class E extends D implements A,B { ... }

(c)
(i) Illegal. A is an interface and cannot be instantiated.

(ii) Legal. Class F implements A, so objects of type F can be assigned to references of type A.

(iii) Legal. Class F implements A, so references of type F can be cast to references of type A.

(iv) Legal. Class G extends F which implements A, so G implements A by extension.

(v) Legal. Class G is a sublass of class F.

(vi) Illegal. A reference to an object of type F cannot be downcast into a reference of type G without a cast. This is a compile time error.

(vii) Illegal. A reference to an object of type F cannot be downcast into a reference of type G without a cast. The compiler is happy but at runtime we get a ClassCastException because the object is not of class G.

(viii) Legal. Class I implements interface C, so it implements all superinterfaces of C, one of which is B.

Problem 2. (20 points)

```
  3. private float x;
```

Instance variable should be declared protected, so subclasses can access it.

```
  4. private float y;
```

Declare it to be protected for same reason as above.

```
  6. public Complex complex(float r, float i) {
```

Constructors do not have return type. Replace with

```
  6. public Complex(float r, float i) {
```

```
  8. float x = r;
```

This introduces a local variable named x when the intention is to assign to the instance variable x. Replace with

```
  8. x = r;
```

```
  10. float y = i;
```

Same as previous problem. Replace with

```
  10.       y = i;
```

```
  13.   public int Re() {
```

Return type of method must be float. Replace with

```
  13.   public float Re() {
```

```
    16.   public int Im() {
```

Return type of method must be float. Replace with

```
    16.   public float Im()  { }
```

```
    21. class BComplex implements Complex {
```

Replace ‘‘implements’’ with ‘‘extends’’ since BComplex is a subclass of Complex.

```
25. return null;
```

A void method can have a return without an expression. So either eliminate the line or remove the null.

```
35. public void main(String[] arg){
```

Declaration of main should be

```
35. public static void main(String[] arg){
```

Note that parameter name can be arbitrary.

```
37.     Complex c = new BComplex(1.0f,2.0f);}
```

Constructors are not inherited, so only constructor in BComplex is default constructor. Add a constructor to this class as follows:

```
public BComplex(float r, float i) {
    super(r,i);
}
```

```
38.     c.setRe(3.0);
```

Type of reference c is Complex which does not have a method named setRe. Replace this line with

```
38. ((BComplex)c).setRe(3.0);
```

Another solution: change type of C to BComplex in line 37.

Note the following statements:

```
30. return;
```

This is legal in a method whose return type is void.

```
39. System.out.println(c.x);
```

This is legal if x is declared protected because variables that are declared to be protected are visible to any other class in the same package, even if that class is not a subclass of the class in which x is declared.

```
Problem 3. (25 points)
---------
(a)
  public static float power(float x, int n) {
    if (n == 0) return 1.0f;
    else return x*power(x,n-1);
  }

(b)

(i) There are two base cases: n==0 and n==1.
    The problem can also be solved by one base case: n==0.

(ii) The output is 1.0 and x for the two base cases n==0 and n==1 respectively.
     Or the output is 1.0 if only one base case is used (n==0).

(iii) Here is the code:
  public static float coolPower(float x, int n) {
    if (n == 0) return 1.0f;
    if (n == 1) return x;    // This line can be removed.
    if (n/2 * 2 == n) //n is even
      {float f = coolPower(x, n/2);
       return f*f;}
    else return x*coolPower(x,n-1);
  }
}

Problem 4 (35 points)
---------
/**
The program below computes the frequencies of values (>= 0) in
an array of array of int (AAI) (sometimes called multi-dimensional arrays),
and prints a histogram.

The AAI is initialized to the following values:
1st row: 2
2nd row: 5, 4
3rd row: 4, 5, 4

A frequency table is computed for the AAI, i.e. how many times
a particular value occurs in the AAI.
In our case the frequency table computed would look like this:
Value:      0  1  2  3   4   5
Frequency:  0  0  1  0   3   2

The histogram based on these frequencies would look like this:
    *
    **
  * **
012345
i.e. each "bar" in the histogram has number of "stars" equal
to the frequency of the value indicated below the bar.

You cannot change the code that is provided.
```

```
There are 7 short questions [from (a) to (g)].
*/

public class ArrayOfArrays {

      public static void main(String args[]) {

        // Creates an AAI which has 3 rows.
        int[][] values = new int[3][];

        // (a) (4 points)
        // Write code which creates the following structure for the AAI:
        // 1st row of length 1, 2nd of length 2 and 3rd of length 3.

        for (int i = 0; i < values.length; i++)
            values[i] = new int[i+1];
        // Note that each row is denoted by values[i], for i = 0, 1, 2.

        // (b) (3 points)
        // Write code to initialize the rows of the AAI as follows:
        // 1st row: 2
        // 2nd row: 5, 4
        // 3rd row: 4, 2, 4

        values[0][0] = 2;
        values[1][0] = 5; values[1][1] = 4;
        values[2][0] = 4; values[2][1] = 5; values[2][2] = 4;

        /*
        Block syntax, {..}, can only be used to initialize an array when the
        array reference is declared:

        int[] array1 = {10, 20}; // declared, created and initialized.

        The block syntax CANNOT be used to initialize an array which has been
        declared and created. For example,

        int[] anArray = new int[4]; // array created to hold 4 ints.
        anArray = {20, 40, 60, 70}; // block syntax illegal.

        In the above case each element can be initialized explicitly:

        anArray[0] = 20; anArray[1] = 40; anArray[2] = 60; anArray[3] = 70;

        However you can create an anonymous array, and assign it to an array
        reference any time:

        anArray = new int[] {20, 40, 60, 70}; // anonymous array

        */

        // Computes the frequencies and prints the histogram
        int[] frequencies = computeFrequencies(values);
        printHistogram(frequencies);
```

```java
    // (c) (3 points)
    values[0] = values[2] = values[1];
    // What will the following statement print?
    // Multiple assignment results in aliases with values[0] and
    // values[2] denoting the same row as values[1], i.e. {5,4}.
    // Print out:
    // 5 4
    // 5 4
    // 5 4
    printValues(values);
  }

// Prints an AAI.
public static void printValues(int[][] anAAI) {
    for (int i = 0; i < anAAI.length; i++) {
        for (int j = 0; j < anAAI[i].length; j++)
            System.out.print(anAAI[i][j]);
        System.out.println();
    }
}

// (d) (3 points)
// Write a method to compute the max value in an array of int.
public static int findMax(int[] anIntArray) {
    int max = anIntArray[0];
    for (int i = 1; i < anIntArray.length; i++)
        if(max < anIntArray[i]) max = anIntArray[i];
    return max;
}

// (e) (4 points)
// Write a method to compute the max value in an AAI.
// Uses the method to find the max in an array of int.
public static int findMax(int[][] anAAI) {
    int max = findMax(anAAI[0]);
    for (int i = 1; i < anAAI.length; i++) {
        int maxInRow = findMax(anAAI[i]);
        if(max < maxInRow) max = maxInRow;
    }
    return max;
}

// (f) (8 points)
// Write a method to compute the frequencies of values in an AAI.
// Assume all values in the AAI are greater or equal to 0.
// The method should return an array of int containing the frequencies.
// Note that the length of this array is maximum value in the AAI + 1.
public static int[] computeFrequencies(int[][] anAAI) {
    int[] frequencies = new int[findMax(anAAI)+1];
    for (int i = 0; i < anAAI.length; i++)
        for (int j = 0; j < anAAI[i].length; j++)
            ++frequencies[anAAI[i][j]];
    return frequencies;
}
```

```
    // (g) (10 points)
    // Write a method to print the histogram.
    // Given the present data, it should print the following:
    //     *
    //     **
    //   * **
    // 012345
    // Note that the number of ‘rows’ in the histogram is equal to the
    // maximum frequency in the frequency array.
    public static void printHistogram(int[] anIntArray) {
        for (int maxFreq = findMax(anIntArray); maxFreq > 0; maxFreq--) {
            for (int j = 0; j < anIntArray.length; j++)
                if (anIntArray[j] >= maxFreq )
                    System.out.print("*");
                else
                    System.out.print(" ");
            System.out.println();
        }
        for (int j = 0; j < anIntArray.length; j++)
            System.out.print(j);
        System.out.println();
    }
}
```