Solutions to old exam questions

Question 1A [4]

- Environment Canada will report a humidex value as part of a weather forecast if the temperature (T) is greater than or equal to 30 degrees, if the temperature is greater than or equal to 25 degrees and the humidity (H) is greater than 35%, or the temperature is greater than or equal to 20 degrees and the humidity is greater than or equal to 65%.
- Write a Boolean expression that is true if Environment Canada will report a humidex value, and false otherwise.

Question 1A

- Environment Canada will report a humidex value as part of a weather forecast if the temperature (T) is greater than or equal to 30 degrees, if the temperature is greater than or equal to 25 degrees and the humidity (H) is greater than 35%, or the temperature is greater than or equal to 20 degrees and the humidity is greater than or equal to 65%.

- Answer:

```
(T >= 30) || ((T >= 25) && (H > 35)) || ((T >= 20) && (H >= 65))
```

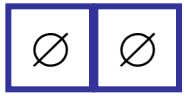
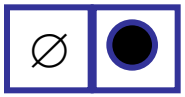
Question 1B [4]

- Consider the following Java program :

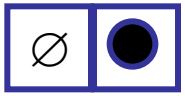
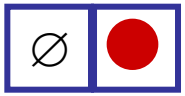
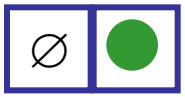
```
MyClass[] obj;
int index;
obj = new MyClass[2];
index = 15;
while( index > 2 )
{
   obj[index % 2] = new MyClass( );
   index = index / 2;
}
// Line X
```

- How many instances of **MyClass** are created during the execution of this program? [2]
- How many instances of **MyClass** are still accessible at **Line X**? [2]

Question 1B

	obj	index	# objects
Initial values	?	?	
`obj = new MyClass[2];`			
`index = 15;`		15	
`while( index > 2 ) : true`			
`obj[index % 2] = new MyClass( );`			1
`index = index / 2;`		7	
`while( index > 2 ) : true`			

Question 1B

	obj	index	# objects
From previous page		7	1
`obj[index % 2] = new MyClass( );`			2
`index = index / 2;`		3	
`while( index > 2 ) : true`			
`obj[index % 2] = new MyClass( );`			3
`index = index / 2;`		1	
`while( index > 2 ) : false`			

Number of objects created: 3 Number still accessible: 1

Question 1C [4]

```
class Foo
{
  private int x1;
  public static int x2;
  public static Bar x3;

  public Foo(int x4)
  {
       ...
  }
}
```

```
class Bar
{
  public int x5;
  public static int x6()
  {
       ...
  }

  public Foo x7()
  {
       ...
  }
}
```

- Suppose that the following instructions are used in the `main()` method in a class `Test`. Each choice should be considered **independently** - as it if were in its own `main()` method. Circle the letter of the statement which causes a compilation error.

```
(a)  Foo[] a = new Foo[5];
     a[4] = new Foo(-1);
(b)  Foo f = Bar.x7();
(c)  Foo.x2 = Bar.x6();
d)  int k = Foo.x3.x5;
e)  Bar b = new Bar();
    Foo f = b.x7();
```

Question 1C

```
class Foo
{
  private int x1;
  public static int x2;
  public static Bar x3;

  public Foo(int x4)
  {
        ...
  }
}
```

```
class Bar
{
  public int x5;
  public static int x6()
  {
        ...
  }

  public Foo x7()
  {
        ...
  }
}
```

a) OK

```
Foo[] a = new Foo[5];
```

Declare and create an array of 5 **`Foo`** object references. The references are all **`null`**.

```
a[4] = new Foo(-1);
```

There is a public **`Foo`** constructor with 1 integer parameter.

Question 1C

```
class Foo
{
  private int x1;
  public static int x2;
  public static Bar x3;

  public Foo(int x4)
  {
        ...
  }
}
```

```
class Bar
{
  public int x5;
  public static int x6()
  {
        ...
  }

  public Foo x7()
  {
        ...
  }
}
```

b) Compile error

```
Foo f = Bar.x7();
```

The method **`x7()`** in the **`Bar`** class does not have the keyword **`static`**, so it is an instance method. Instance methods cannot be called using the class name.

Question 1C

```
class Foo
{
  private int x1;
  public static int x2;
  public static Bar x3;

  public Foo(int x4)
  {
        ...
  }
}
```

```
class Bar
{
  public int x5;
  public static int x6()
  {
        ...
  }

  public Foo x7()
  {
        ...
  }
}
```

c) OK

```
Foo.x2 = Bar.x6();
```

The method **`x6()`** in the **`Bar`** class is **`public`**, **`static`**, and returns a value of type **`int`**. Because of the **`static`** keyword, the method can be called via the class name.

The value **`x2`** in the **`Foo`** class is **`public`**, **`static`**, and of type **`int`**. Since the value is **`static`**, it is a class variable, and since it is **`public`**, it can be accessed outside the class.

Question 1C

```
class Foo
{
  private int x1;
  public static int x2;
  public static Bar x3;

  public Foo(int x4)
  {
        ...
  }
}
```

```
class Bar
{
  public int x5;
  public static int x6()
  {
        ...
  }

  public Foo x7()
  {
        ...
  }
}
```

d) OK

```
int k = Foo.x3.x5
```

The value **x3** in the **Foo** class is **public**, **static**, and of type **Bar**. Because the value is **public**, **x3** can be accessed outside the class, and because it is **static**, it can be accessed via the class name.

In the class **Bar**, the value **x5** is public and of type **int**. Therefore, **x5** can be accessed from outside of the class, and assigned to a variable of type **int**.

Question 1C

```
class Foo
{
  private int x1;
  public static int x2;
  public static Bar x3;

  public Foo(int x4)
  {
        ...
  }
}
```

```
class Bar
{
  public int x5;
  public static int x6()
  {
        ...
  }

  public Foo x7()
  {
        ...
  }
}
```

e) OK

```
Bar b = new Bar();
```

Declare and create a new `Bar()` object. The invisible default constructor is used, since no constructors are defined.

```
Foo f = b.x7();
```

Method `x7()` is a `public` instance method, so it can be called on a `Bar` object. It returns an object of type `Foo`, which can be assigned to `f`.

Question 2 [8]

```
class Football
{
  public static void main(String[ ] args)
  {
    char[] t = {'G', 'e', 'e', '-', 'G', 'e', 'e'};
    Rec(t, t.length - 1);
  }

  public static void Rec(char[] var, int i)
  {
    if (i < 0)
    {   ; // do nothing
    }
    else
    {
      if ( (var[i] > 'A') && (var[i] < 'Z') )
      {
        System.out.print( (char) (var[i] - 'A' + 'a') );
      }
      else
      {
        if ( (var[i] > 'a') && (var[i] < 'z') )
        {
          System.out.print( (char) (var[i] - 'a' + 'A') );
        }
        else
        {     ; // do nothing
        }
      }
      Rec(var, i - 1);
    }
 }}
```

- Here is a program that uses recursion.
- What is printed by this program?

Question 2

- Let's look at various parts of the program:

```
if ( (var[i] > 'A') && (var[i] < 'Z') )
{
   System.out.print( (char) (var[i] - 'A' + 'a') );
}
```

- The above will take any upper case letter (except for `'A'` and `'Z'`) at index `i` in the array `var`, convert it to lower case, and display the character on the console.

Question 2

```
if ( (var[i] > 'a') && (var[i] < 'z') )
{
   System.out.print( (char) (var[i] - 'a' + 'A') );
}
```

- The above will take any lower case letter (except for **'a'** and **'z'**) at index **i** in the array **var**, convert it to upper case, and display the character on the console.

Question 2

```
class Football
{
  public static void main(String[ ] args)
  {
    char[] t = {'G', 'e', 'e', '-', 'G', 'e', 'e'};
    Rec(t, t.length - 1);
  }

  public static void Rec(char[] var, int i)
  {
    if (i < 0)
    {
      ; // do nothing
    }
    else
    {
      // if var[i] is upper case, convert to lower case and print it
      // else if var[i] is lower case, convert to upper case and print it
      // else do nothing

      Rec(var, i - 1);
    }
  }
}
```

Question 2

```
public static void Rec(char[] var, int i)
{
  if (i < 0)                  // Recursion base case
  {
    ; // do nothing
  }
  else
  {
    // do stuff with var[i]

    Rec(var, i - 1);          // Recursive call
  }
}
```

- The above will go through the array **var** in the backwards direction; that is, the index **i** will be decreasing down to **0**.

Question 2

```
public static void main(String[ ] args)
{
  char[] t = {'G', 'e', 'e', '-', 'G', 'e', 'e'};
  Rec(t, t.length - 1);
}
```

- The method **Rec** will start from the end of the array **t**, change the case of each letter (ignoring non-letter characters), and print it.

- The result: **EEgEEg**

- Irrelevant side remark: Did you see why the class was called "Football "? ☺

Question 3 [15]

- The lower right sub-matrix of a matrix is formed by selecting one element position (row and column) and excluding all elements that are to the left or above the selected element. For example, in the matrix M below, if we select $M_{11} = 5$, the matrix S is the lower right sub-matrix.

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad S = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$$

- Write a Java method that will take a matrix of integers M, and a row and column index, and returns a new matrix that is the lower right sub-matrix of M formed from that position. The header of the method is as follows:

```
public static int[][] subMatrix( int[][] m, int theRow, int theCol )
```

Question 4

```
public static int[][] submatrix( int[][] m, int theRow, int theCol  )
{
   int[][] s;   // RESULT:  the submatrix of matrix m
   int sRows;   // INTERMEDIATE :  number of rows in s
   int sCols;   // INTERMEDIATE :  number of columns in s
   int row;     // INTERMEDIATE:  index for row position in s
   int col;     // INTERMEDIATE:  index for column position in s

   sRows = m.length - theRow;
   sCols = m[0].length - theCol;

   s = new int[sRows][sCols];
   for ( row = 0; row < sRows; row = row + 1 )
   {
      for ( col = 0; col < sCols; col = col + 1 )
      {
         s[row][col] = m[row + theRow][col + theCol] ;
      }
   }
   return s;
}
```

Question 4 [25]

- In this question, you will create a class Experiment that represents a record of some sort of scientific experiment.

- In order to verify that the results of an experiment are repeatable, there is a class Trial that contains the results from one run of an experiment. An experiment will then include a number of Trial objects.

Question 4

- The class Trial stores a result that was measured during an experiment, and the duration that the experiment took, measured in milliseconds. The class Trial has already been implemented. A UML class diagram for the class Trial is as follows:

Trial
- result : double - duration: int
+ Trial(theResult : double, theDuration : double) + getResult( ) : double + getDuration( ) : int

Question 4

- In the rest of this question, you will fill in the methods for Experiment. Your Experiment class should provide four public methods and/or constructors that would permit the following class TestExperiment to execute:

```
class TestExperiment
{
   public static void main (String[] args)
   {
      Experiment anExperiment;
      anExperiment = new Experiment( 2 );
      anExperiment.addTrial( new Trial( 99.1, 10000 ) );
      anExperiment.addTrial( new Trial( 97.1, 11000 ) );
      anExperiment.addTrial( new Trial( 94.1, 12000 ) );
      Experiment.setPredictedResult( 98.6 );
      anExperiment.print();
   }
}
```

- Executing `main()` would result in the following being printed on the screen :

```
No more trials can be added to the experiment.
Trial 0: Result 99.1, duration 10000 (within 0.5 of prediction)
Trial 1: Result 97.1, duration 11000 (within 1.5 of prediction)
```

Question 4

```
class Experiment
{
   // FIELD DECLARATION(S): (4 marks)




    // CONSTRUCTOR: (5 marks)
    // Takes one integer parameter representing the maximum
    // number of trials that can be put into the experiment
```

Question 4

```
class Experiment
{
   // FIELD DECLARATION(S): (4 marks)

    private Trial[] trials;
    private int numTrials;
    private static double prediction;

    // CONSTRUCTOR: (5 marks)
    // Takes one integer parameter representing the maximum
    // number of trials that can be put into the experiment

    public Experiment( int maxTrials )
    {
        trials = new Trial[maxTrials];
        numTrials = 0;
    }
```

Question 4

```
// METHOD setPredictedResult: (4 marks)
// Method parameters: a double that is the predicted result
// of the experiment.
// RESULT:  none
```

Question 4

```
 // METHOD setPredictedResult: (4 marks)
 // Method parameters: a double that is the predicted result
 // of the experiment.
 // RESULT:  none

public static void setPredictedResult( doublenewPrediction )
{
   prediction = newPrediction;
}
```

Question 4

```
// MODIFIER METHOD addTrial: (6 marks)
// Method parameters: a Trial object that should be added to the Experiment.
// Results:  will print a message if the experiment has no room to store
// additional trials (see sample output for message format)
// Modified:  the Experiment object
```

Question 4

```
// MODIFIER METHOD addTrial: (6 marks)
// Method parameters: a Trial object that should be added to the Experiment.
// Results:  will print a message if the experiment has no room to store
// additional trials (see sample output for message format)
// Modified:  the Experiment object

public void addTrial( Trial newTrial )
{
   if ( numTrials >= trials.length )
   {
      System.out.println("No more trials can be added to the experiment.");
   }
   else
   {
      trials[numTrials] = newTrial;
      numTrials = numTrials + 1;
   }
}
```

Question 4

```
// METHOD print: (6 marks)
// Method parameters: (none)
// Returns: (none)
// This method prints the result and duration  of each trial, along with
// the absolute value of the difference from the predicted result.
// See the TestExperiment sample output for exact format.
```

Question 4

```
// METHOD print: (6 marks)
// Method parameters: (none)
// Returns: (none)
// This method prints the result and duration  of each trial, along with
// the absolute value of the difference from the predicted result.
// See the TestExperiment sample output for exact format.


public void print( Trial newTrial )
{
   int index;
   double aResult;
   double difference;
   for ( index = 0; index < numTrials; index = index + 1 )
   {
      aResult trials[index].getResult();
      difference = Math.abs( aResult - prediction );
      System.out.print("Trial " + index + ": " );
      System.out.print("Result " + aResult + ", " );
      System.out.print("Duration " + trials[index].getDuration() + ", " );
      System.out.println("(within " + difference + " of prediction)" );
   }
}
```

Question 5 [10]

Binary search

Suppose that you have an array of integers that is already known to be sorted in ascending order, and to contain no duplicate values. A binary search on such an array is done by locating the middle element of an array and comparing the search value with that element. If we have not found the value for which we are searching, we can choose the appropriate sub-interval (either the left or the right side) to restrict the search.

Write a recursive method that will perform a binary search on an array a for the value findMe between positions startIndex and endIndex. You can assume that both positions are less than the length of the array. The method should return true if the value is contained in the array, and false otherwise.

Q5

```
public static boolean searchRec( int[] valueList, int findMe,
                                 int leftIndex, int rightIndex )
{  boolean found; // RESULT: True if search is successful
   int mid; // Index of array closest to the midpoint between
            // leftIndex and rightIndex
 // Check for base case.
 // The base case covers 2 situations: leftIndex and rightIndex are the
 // same, or they are two consecutive array positions. The latter case is
 // needed as there is no useful midpoint between two consecutive array
 // positions, and the possibility of not reducing the size of the interval.
 if ( leftIndex + 1 >= rightIndex )
  {// For the base case, if the value doesn't match one of the two
   // possible endpoints, the value is not in the array.
   found = findMe == valueList[leftIndex] ||
           findMe == valueList[rightIndex];
  }
```

Q5

```
else
 {  // Determine array position closest to the midpoint
   //  between leftIndex and rightIndex.
   mid = ( leftIndex + rightIndex ) / 2;
   // Compare with value at midpoint.
   if ( findMe == valueList[mid] )
    { // We got lucky and found the value.
      found = true;
    }
   else
    { // Decide whether the value, if it were present, would be
      // to left of the midpoint or to the right of the midpoint.
```

Q5

```
    if ( findMe < valueList[mid] )
     { // Value is on left side of midpoint.
       // Search left half of array recursively.
     found = searchRec( valueList, findMe, leftIndex, mid );
     }
    else
     { // Value is on the right side of the midpoint.
       // Search right half of array recursively.
      found = searchRec( valueList, findMe, mid, rightIndex );
     }
   }
 }
 return found; // RETURN RESULT
}
```