BIRKBECK (University of London)

BSc EXAMINATION FOR INTERNAL STUDENTS

Department of Computer Science and Information Systems

BUCI056H6 - Software and Programming III

DATE OF EXAMINATION: Tuesday, 9th June 2020

TIME OF EXAMINATION: 09:30

DURATION OF PAPER: Three hours

Rubric

- Open-book and online examination.
- Candidates should attempt **ALL** questions on the paper.
- The number of marks varies from question to question.
- You are advised to look through the entire examination paper before getting started, in order to plan your strategy.
- Simplicity and clarity of expression in your answers is important.
- You should answer the programming questions using the `C#` programming language.
- You should avoid the use of mutable state or mutable collections in your solutions whenever possible.
- You should "`add`/`commit`/`push`" to your repository in the usual way.
- Should you become "detached" from the Internet you can still add/commit locally and "push" at a later point.
- Each question should be submitted as a pdf file and/or a code file, located in the appropriate folder (provided); supplemental files are allowed (e.g., for diagrams, code fragments).

---

1. **[6 marks]**

   > "One should always prefer *inheritance* over *composition* when designing a system."

   Discuss this statement with reference to the object-oriented model of programming, stating whether you agree with the sentiment. You should provide appropriately detailed examples to illustrate your answer.

2. **[8 marks]**
   An Algebraic Data Type (ADT) is a composite type made by composing other types.
   Using appropriate examples of your choice, explain the two main ADT types and how they relate to Object-Oriented programming terminology.
3. **[12 marks]**
   Discuss each of the following statements in turn using appropriately detailed examples to illustrate your answers.

   > "Debugging is easier in programs written in functional style"

   > "We should prefer expressions instead of statements"

   > "Pure functions makes it easier to reason about our code"

   > "If we use pure functions how do we change state?
4. **[10 marks]**

   Certain types of online community networks specialise in holding photos in the cloud.
   The principle is simple: Users upload photos, which all others can view but only friends can download or comment on.

   1. Sketch out the use of a *Proxy* pattern for such a system by providing appropriate classes to illustrate your solution. A diagram of the relationships between the classes may assist your answer.
   2. Provide an explanation of how the download and comment functionality will be implemented for such a system.

   You should clearly state any assumptions that you make about the functionality of the application.
5. **[10 marks]**
   The code shown in `Program.cs` in the `Q1` folder, and also shown below, executes three functions, timing the execution of each.

```
namespace Q5
  {
    public static class Program
    {
      public static void Main(string[] args)
      {
        var watch = new Stopwatch();
        watch.Start();
        DoSomething();
        watch.Stop();
        Console.WriteLine($"DoS() took {watch.ElapsedMilliseconds} ms to
execute.");
        watch.Reset();

        watch.Start();
        DoSomethingElse();
        watch.Stop();
        Console.WriteLine($"DoS2() took {watch.ElapsedMilliseconds} ms to
execute.");
        watch.Reset();

        watch.Start();
        DoSomethingDifferent();
        watch.Stop();
        Console.WriteLine($"DoS3() took {watch.ElapsedMilliseconds} ms to
execute.");
      }

      // These methods perform some type of intensive computation.
      private static void DoS() => Thread.Sleep(1000);
      private static void DoS2() => Thread.Sleep(1200);
      private static void DoS3() => Thread.Sleep(1500);
    }
  }
```

Refactor this code so that the repeated code is reduced, ensuring that the following code fragment:

```
watch.Start();
// do some time expensive operation
watch.Stop();
Console.WriteLine($"DoSomething() took {watch.ElapsedMilliseconds} ms to
execute.");
watch.Reset();
```

is not duplicated in the source code.

Your new function should be called `LogEllapsedMs` and should utilise *higher-order functions* and *generics*. The first parameter of your function should be the message to be displayed, and the second the function to time.

You should list three advantages in using this approach.

6. **[14 marks]**
   This question concerns the items that you usually find in a library, namely, *books*, *magazines*, and *articles*. Articles can be found separated or grouped together in magazines. A magazine will consist of at least one article. Provide `C#` code to support this model.

   Extend this model by writing appropriate `C#` code using `LINQ` so that one would be able to:

   1. Print the total number of pages for all the items that are in the library.
      For example if the library owns a book with 540 pages and a magazine with two articles of 25 pages each then the total number of pages is 590.
   2. Print the number of different authors for the items that are in the possession of the library.
      For example if the library has two books authored by William Shakespeare, one magazine containing two articles, one authored by Stephen Hawking and one authored

by Bart Simpson, then the total number of different authors should be 3.

Which design pattern would you use to implement this functionality?
Provide a short explanation to justify your answer.

7. **[12 marks]**
Consider the classes `LibraryClass` and `MyClass` provided in the folder `Q7`, and shown below:

```
using System;

namespace Q7
{
  public abstract class LibraryClass
  {
    public void CleanUp() => Console.WriteLine("Cleaning up the disc");
  }
}
```

and

```
using System;

namespace Q7
{
  public class MyClass : LibraryClass
  {
    public void Delete() => Console.WriteLine("Erasing the whole disc");
  }
}
```

When a user of the class calls the `CleanUp` method on an instance of `MyClass` it does exactly that; tidies up the library structure. When a user calls `Delete` it erases the hard disc!

Later on in the lifetime of this program the vendor ships a new version of `LibraryClass`.

The user of `MyClass` now calls `CleanUp` again without recompiling `MyClass`.

1. What is the result and why?
2. How would the execution, and resulting output differ if this were a `Java` program rather than `C#`?

8. **[14 marks]**
Consider the `LotsOfFields` class (under the folder `Q6`) which has, "lots of fields".

```
  namespace Q8
  {
      public class LotsOfFields
      {
          private int testInt;
          public double testDouble;
```

```
            protected string testString;
            private long testLong;
            protected double aDouble;
            public string aString;
            private Calendar aCalendar;
            public StringBuilder aBuilder;
            private char testChar;
            public short testShort;
            protected byte testByte;
            public byte aByte;
            protected StringBuilder aBuffer;
            private BigInteger testBigInt;
            protected BigInteger testBigNumber;
            protected float testFloat;
            public float aFloat;
            private Thread aThread;
            public Thread testThread;
            private object aPredicate;
            protected object testPredicate;
            public object anObject;
            private object hiddenObject;
            protected object anotherObject;
            private string anotherString;
            protected string moreStrings;
            public int anotherInt;
            private Exception internalException;
            protected Exception inheritableException;
            public Exception justException;
            public Stream aStream;
            protected Stream moreStreamsz;
            private Stream secretStream;

            public LotsOfFields(int testInt)
            {
                this.testInt = testInt;
            }
        }
    }
```

You are required to write a program which will read from the input, a line at a time.
Once you have read the "end of input" command, you should execute the last command you read.

The list of possible commands is:

- `private` - print all private fields
- `protected` - print all protected fields
- `public` - print all public fields
- `all` - print ALL declared fields
- `END` - end the input

For each executed command you print the fields of `LotsOfFields` that have the given access modifier, as described above; the output format is:

```
<access modifier> <field type> <field name>
```

Sample input and output is shown below:

```
Input:
protected
END

Output:
protected String testString
protected Double aDouble
protected Byte testByte
protected StringBuilder aBuffer
protected BigInteger testBigNumber
protected Single testFloat
protected Object testPredicate
protected Object anotherObject
protected String moreStrings
protected Exception inheritableException
protected Stream moreStreamsz

Input:
private
public
private
END

Output:
private Int32 testInt
private Int64 testLong
private Calendar aCalendar
private Char testChar
private BigInteger testBigInt
private Thread aThread
private Object aPredicate
private Object hiddenObject
private String anotherString
private Exception internalException
private Stream secretStream
private Int32 testInt
private Int64 testLong
private Calendar aCalendar
private Char testChar
private BigInteger testBigInt
private Thread aThread
private Object aPredicate
private Object hiddenObject
private String anotherString
private Exception internalException
private Stream secretStream

Input:
all
END

Output:
private Int32 testInt
public Double testDouble
protected String testString
private Int64 testLong
protected Double aDouble
public String aString
private Calendar aCalendar
public StringBuilder aBuilder
private Char testChar
public Int16 testShort
protected Byte testByte
public Byte aByte
protected StringBuilder aBuffer
private BigInteger testBigInt
protected BigInteger testBigNumber
protected Single testFloat
public Single aFloat
private Thread aThread
public Thread testThread
private Object aPredicate
protected Object testPredicate
public Object anObject
private Object hiddenObject
protected Object anotherObject
private String anotherString
protected String moreStrings
public Int32 anotherInt
private Exception internalException
protected Exception inheritableException
public Exception justException
public Stream aStream
protected Stream moreStreamsz
private Stream secretStream
```

9. **[14 marks]**

Write an implementation of the following `Differences` function, which takes as a paramter, an immutable array of integers, and returns an immutable array of the pairwise differences of the elements of the array:

```
static ImmutableArray<int> Differences(ImmutableArray<int> ls)
```

Specifically, calling `Differences(xs)` must return an array `ys` such that:

- `xs.Length == ys.Length`
- For all `0 < i < ys.Length`, then `ys(i) == xs(i) - xs(i - 1)`

Some examples of the usage of `Differences` would be:

- Applying `Differences` to the empty array would result in the empty array.
- Applying `Differences` to the array consisting of the number one would result in an array containing the number one.
- Applying `Differences` to the array (1, -2, 3, -4, 5, -6) would result in the array (1, -3, 5, -7, 9, -11).

You must use the `ImmutableArray` class and the `Zip` `LINQ` method (no `for` loops).

END OF PAPER