

COMP3506/COMP7505—Algorithms and Data Structures

School of Information Technology and Electrical Engineering

Week 2 Tutorial Sample Solutions

Question 1. *Software engineering has a variety of objectives including reliability, utility, flexibility (portable and adaptable) and efficiency (in programmer effort and/or machine space and time complexity).*

- a. Life-critical software applications: Life-critical software is used in nuclear reactors, aeroplanes, and pacemakers. The Therac-25, a radiation therapy machine, gave overdoses of radiation to six patients because of a software error.
- b. Simple portable devices containing a computer and an application that would be useful to run on it: mobile phone, personal digital assistant (PDA), applications – email, games, stock market reports.
- c. Give an example of an application that processes terabytes of data: google search engine, Telstra phone logs, bioinformatics data bases

Question 2. *Discuss how the design decisions for choosing algorithms for each application in Question 1 might result in different space and time tradeoffs.*

Design decisions have to effectively manage a variety of objectives, including reliability, utility, flexibility (portable and adaptable) and efficiency (in programmer effort and/or machine space and time complexity).

Life critical software has to be reliable. Provably correct code is likely to be favoured over more efficient time or space algorithms that are harder to check.

Portable devices need to be usable in real time. Most have space restrictions and need to have fast response times.

Applications that process terabytes of data need to be effective in space usage and some also require good response time. Google uses simple algorithms that are effective because they have so much data, in favour of more complex algorithms that would take longer to compute.

Question 3. *Compare and contrast the terms Abstract Data Type and Concrete Data Type.*

Both Abstract Data Types and Concrete Data Types express a functional interface that allows user code to perform some meaningful sequence of operations. For example the Stack ADT contains a push(Object) operation, a Concrete Data Type implementing the Stack ADT contains a push operation.

However, while an Abstract Data Type defines the operations available upon it. It does not reveal how those operations are performed (algorithms), nor how data is stored (data-structures). In Java an ADT may be expressed using the "interface" construct.

A Concrete Data Type on the other hand explicitly details how the data is stored - the members of the class - and how the operations are performed - the contents of the method implementations. In Java a Concrete Data Type may be expressed using the "class" construct.

In order to use an ADT, one must always instantiate a Concrete Data Type that provides the operation implementations and the specification of how data is stored. This may be done using a factory method, or by directory instantiating the type and assigning it to an interface variable.

Question 4. *Specify and ADT in Java using the interface construct (you name the interface). The ADT should specify a method (you name the method) that takes a String parameter and returns its length as an int.*

```
public interface MyInterface
{
    public int MyMethod( String aString );
}
```

Question 5. *Now specify the interface described in Question 2 using a generic interface that specifies the parameterised type E. The method should take as a parameter an object of the parameterised type and return an int as the length.*

Note: In Java all objects must implement the "toString" method. Therefore an implementation is able to call the "toString" method on any object type passed in to convert it to a string, then is able to find the length of that string and return it.

```
public interface MyInterface<E>
{
    public int MyMethod( E anObject );
}
```