The University of Nottingham

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

A LEVEL 2 MODULE, AUTUMN SEMESTER 2012-2013

SOFTWARE ENGINEERING DESIGN

Time allowed TWO Hours

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Complete ALL the examination tasks

You may use either the CODEBLOCKS or MICROSOFT VC compilers

Please note well: for all tasks marks will be given for the correct declaration of classes and functions even if their definition (implementation) is missing.

Only silent, self contained calculators with a Single-Line Display or Dual-Line Display are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so

ADDITIONAL MATERIAL: None

INFORMATION FOR INVIGILATORS:

This is an open book examination and candidates may bring in any material on disks, memory sticks etc that they wish.

The course material must be available to the candidates on the computers.

Internet and email access must be disabled on the computers and not available to the candidates.

Question papers should be collected in at the end of the exam – do not allow candidates to take copies from the exam room.

Turn over

Introduction:

A *power company* operating a number of *power stations,* provides electrical power to a number of *customer_factories* within its region*.* The company desires the development of a C++ software package that will allow it to manage its *power stations* in order to meet the demands of its customers in the most cost effective manner.

Each *power station* is to be characterised by its maximum power generation capability in kW, its cost efficiency measured in pence/kWh and by its status which may be online or offline.

Each *customer_factory* is to be characterised by its maximum power requirement in kW.

Moreover, each *customer_factory* is to be able to communicate with the particular *power company* who supplies it with energy. Note well, many *customer_factories* may be supplied by the same *power company.* The exact details of what form this communication might take is not fully specified at present and task 4 below is an illustration of what might be required in the future.

It is emphasised that the communication referred to here is between objects representing different physical entities within the same software package. This is not to be confused with actual hardware communication between the different physical locations.

Examination Task: Part 1

Your first task is to develop the basic *power station* class. Please provide just the minimum functionality expected of any well designed class. Please make any required design decisions in the context of the project described above and in accordance with good software engineering practice.

[40%]

Examination Task: Part 2

Design and implement a minimum functionality class that embodies a *power company.* It is to hold a list of both its *power stations* and its *customer_factories.* (Hint as this is being implemented before the *customer_factory* class it is necessary to at least declare the existence of *customer_factory* class in order to compile it). For simplicity here, it can assumed that each power company has a maximum capacity of 10 power stations and 100 customers.

[15%]

Examination Task: Part 3

Design and implement a minimum functionality class that embodies a *customer_factory.* Each instance of *customer_factory* must store a data item indicating who its energy supplier, an instance of *power company,* is. This must be done in such a way that the *customer_factory* can send messages to its energy supplier by calling selecting *power company* member functions. Instances of *customer_factory* are not to be constructed without proper initialisation of its energy supplier data.

Please provide the ability to read and write the data held in instances of *customer_factory* in the manner other C++ programmers would expect to find of a well designed class.

[15%]

Examination Task: Part 4

It is desired that whenever a *customer_factory* is created or destroyed it notifies its *power company* which adds it to or removes it from its list of customers. Please add this functionality. You may combine this with task 5 rather than provide two different versions.

[15%]

Examination Task: Part 5

To provide for future expansion, it is appropriate to generalise to the concept of a *customer* which has *customer_factory* as one of many possible sub-categories. Implement a *customer* class and modify *customer_factory* and *power_company* so that the software reflects our understanding that *customer_factory* is a *customer.* In particular please pay careful attention to the feature of task 4, recalling that each sub category of *customer* may have their own additional initialisation and destruction requirements.

[15%]

End