Exam Guidelines

COIT 23001

Edition: 1.01 (5 October 2009).

Disclaimer: This exam guideline is provided in good-faith and as a means to assist student in their preparation for examination in this specific course. This guideline is however not provided to students as a guarantee to help pass their exams. This guideline is provided as an external study resource and is not part of the course syllabus. The author and developer of this study guide reserve all rights to remove this guideline without prior notice.
INTRODUCTION

This exam study guideline has been prepared to help students to sit for the examination required for COIT 23001. This guideline also attempts to complement the sample example paper provided on the course website. Some students may find the sample exam paper more challenging than have been expected. With this in mind, I have provided this guideline to, hopefully, help you a little better in understanding the requirements of the exam. Please expect the level of difficulty of the actual exam to be a mixture between the example questions provided here and the ones provided in the sample exam paper.

WHAT YOU MUST KNOW

This is an open book examination, therefore take full advantage of this by being well-prepared. You should not assume that since you can have the textbook and reference material with you during the exam, answers and solutions will be at your fingertips. Ill-preparation and an overload of information (i.e. too many different kind of reference material) can only cause you to fumble and stumble, losing precious time during the exam. This guideline has been provided to help ease you in the preparation of your examination.

(1) The maximum number of marks obtainable from the exam is 70.
(2) Attempt ALL the questions in the exam.
(3) The exam is open book.

WHAT TO STUDY

As this is a postgraduate course, it is assumed that students will have prior and intermediate knowledge of C++ programming concepts and techniques. You will have noticed that the course begins from Chapter 9 onwards in the text. Therefore, it is assumed that you are already familiar with concepts in earlier chapters of the textbook. It is best that you do not attempt to ‘spot’ questions, i.e. do not assume that the exam will only test you on specific areas of the course. The purpose of the exam is to attempt to test you comprehensively in every relevant area in object-oriented principles using C++ as its development platform. In short, there is no short-cut. You will need to put in effort to revise all the specific topics as indicated in the study schedule (available on the course website).

TIPS ON USING REFERENCE MATERIAL

(1) When you revise the material in the textbook, place bookmarks and/or mark the pages that you refer to constantly. This will help you to retrace your steps in the event that you need to refer to certain topic you have studied previously.
(2) Make up a personal index page(s) that point you to topics that are important to you. In this way, you can refer to specific pages that you want. Even though the textbook do provide an index (12 pages long with approximately 1,200 entries), it may be overly generic and too lengthy to allow you to search topics efficiently – especially under time pressure. So use your own indexing system whenever possible.
(3) The powerpoint slides provided for this course is a valuable source of information. It is compact and very readable with the majority of topics and key concepts presented in point/bullet form. Use these for revision whenever possible.
HOW TO REVISE

It is assumed that you have attended classes and tutorials and have submitted both required assignments. In order to attempt to revise in preparation for the exam:

1. Start with the powerpoint slides for the course.
2. Attempt as many questions and exercises as you can beginning from Chapter 9 of the textbook.
3. Revise the tutorial questions.
4. Attempt the sample examination paper provided on the course website.

If you have successfully attempted the revision schedule provided above, you should consider yourself sufficiently prepared for the exam.

TYPES OF EXAM QUESTIONS

Please note that previous exam papers for this paper have never been released due to the pedagogical requirements of the course. However, it is set out consistently from year to year based on a standardised format. There are approximately 15 to 20 questions per exam from year to year, depending on the distribution of marks per questions.

Generally, the questions are presented in, and/or a combination, of four broad categories. Following are the different types of questions with example of possible questions. Following are sample fragments of how the exam questions may be presented and guidelines of what types of answers are expected. Please take note that a separate answer booklet are not provided. Write your answers in the space provided below the question within the question booklet. The marks allocated for the question and the number of lines provided in the answer space is a rough indication of how comprehensive your answers should be. Marks allocated in the examples below are indicative only – actual marks in the actual exam varies.
Example 1: Brief question on a specific concept is given. The expected answer should be written in one or more sentences. Coding or fragment of coding is not expected in the answer.

Question 2: (1 mark)
What is the symbol used in C++ to indicate that a variable is a pointer.

Question 4: (2 marks)
What is the difference between a variable and pointer variable? Provide an example of the two types of variable in your answers.
Example 2: A fragment of C++ code is shown and a question relating to the code is asked. The written answer should be an explanation addressing the question. In some cases involving complex description (e.g. binary trees), providing an illustration may help to clarify your answers. Coding or fragment of coding is not expected in the answer.

Question 1: (2 marks)

The following class contains errors. Identify as many as you can. Write your answers in the space provided.

```cpp
class Triangle
{
private:
    double sideA;
    double sideB;
    double sideC;
public:
    setSides(double, double);
    setArea(double);
};
```
Question 3:  

Consider the following class declaration:

```cpp
class Flower
{
private:
    int bud;
    int stem;
    static int seed;
public:
    Flower()
    { bud = stem = seed; }
    static void putFlower(int bud)
    { seed = bud; }
};
```

(a) How many separate instances of the `bud` member exist?
(b) How many separate instances of the `stem` member exist?
(c) How many separate instances of the `seed` member exist?
(d) What value will be stored in the `bud` and `stem` members of each object?
Question 8: (4 marks)

Write a class declaration named `Box` with private member variable named `Length`, `Width` and `Depth`. Write set and get functions to access these variables. In addition, write a display function to display the contents of `Length`, `Width` and `Depth`. 
Write a function that searches a numeric int array for a specified integer value. The function should return 1 or TRUE if the specified integer value is found in the array. If the specified integer value is not found, the function should throw an exception.
Question 14: (4 marks)

Given the following class declarations:

```cpp
class Plane
{
    protected:
        int numberOfseats;
    public:
        Plane(int s)
        {
            numberOfseats = s;
        }
};

class Helicopter
{
    protected:
        double cargoPayload;
    public:
        Helicopter(float p)
        {
            cargoPayload = p;
        }
};
```

Write the declaration of a class named `SpaceRocket`. The class should be derived from both the `Plane` and `Helicopter` classes above. This should be a case of multiple inheritance, where both `Plane` and `Helicopter` are the base classes.
The class Stock contains three private data members:

- **itemName** a pointer to a dynamic array of characters that contains the name of a specific item in stock
- **supplierName** a pointer to a dynamic array of characters contains the name of the supplier of the specific item stock
- **supplierID** an integer containing the supplier’s ID.

```cpp
#ifndef STOCK_H
#define STOCK_H
#include<iostream>
#include<fstream>
#include<cassert>
#include<string>
using namespace std;

class Stock {
public:
    enum {MAXSTRING=256};
    Stock ( ); // Default constructor
    Stock (const Stock & original); // Copy constructor
    ~ Stock ( ); // Destructor
private:
    char* itemName;
    char* supplierName;
    int supplierID;
};
#endif
```

(a) Write a default constructor for this class. (1 mark)
(b) Convert the given class to a template. (5 marks)
(c) Write a simple driver program to demonstrate the function of the template. (2 marks)

Important note: Example solutions for the questions shown in this document are not available. Please expect the level of difficulty of the actual exam to be a mixture between the example questions provided here and the ones provided in the sample exam paper (available on the course website). Remember, it is always a better idea to overestimate the difficulty of the exam, than to underestimate it.

Edition: 1.01 (5 October 2009).

Disclaimer: This exam guideline is provided in good-faith and as a means to assist student in their preparation for examination in this specific course. This guideline is however not provided to students as a guarantee to help pass their exams. This guideline is provided as an external study resource and is not part of the course syllabus. The author and developer of this study guide reserve all rights to remove this guideline without prior notice.