# COMP2511 Final Exam (20T3)

There are two parts,

- *Part 1: Multiple Choice questions (16 marks)*.
- *Part 2: Programming Questions (24 marks)*.

**Start time** The Final Exam will be available on the class webpage at 09:00am on the exam date (Tuesday 01/December/2020). You will need at most 3 hours to answer all the questions, and you can answer them anytime between 9am to 5pm on the exam day. You can use as many mins/hours as you wish between 9am and 5pm. However, make sure that you submit all your answers by the required due time, that is 5pm on the exam date.The Final Exam will be available on the class webpage at 09:00am on the exam date (Tuesday 01/December/2020). You will need at most 3 hours to answer all the questions, and you can answer them anytime between 9am to 5pm on the exam day. You can use as many mins/hours as you wish between 9am and 5pm. However, make sure that you submit all your answers by the required due time, that is 5pm on the exam date.

**Deadline** You must submit your answers by 5pm (after 8 hours) on the exam date (Tuesday 01/December/2020). If you don't submit your answers for your Final Exam, it will be considered as absent fail for the Final Exam.

**Resources** During the Final Exam, you can (yes you can!) access the course material on the class web page, and also any other pages on the Internet! However, please make sure that you submit your original work and don't copy!

**Assessment** We will award you marks for your final exam based on your submissions. However, please make sure that you submit your original work and don't copy!

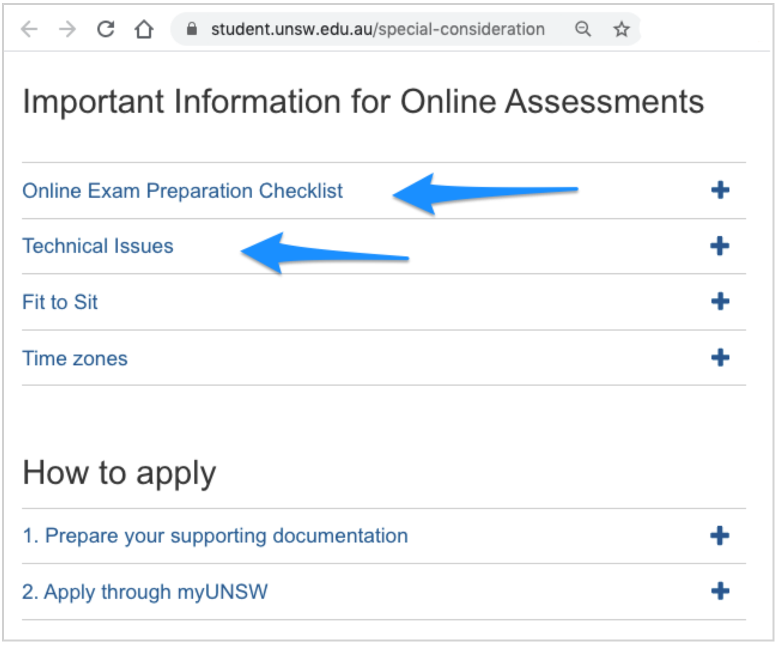
## Notes:

- Answer *all* questions. Questions may *not* be worth equal marks. Questions may be answered in any order.
- All answers must be submitted online using the provided instructions in the respective questions.

- Please note that the topics covered in the final exam may be different to the topics covered in this sample exam. Also, the types/styles of questions and marks distribution across topics may also vary.

## Important Information for Online Assessments

Before your final exam, you must read the section "*Important Information for Online Assessments*" available on the Special Consideration webpage. It offers information on what you should do if you experience a technical issue that is beyond your control and impacts on your ability to complete an assessment, and the other related topics. In particular, how and what to document for a special consideration application.


student.unsw.edu.au/special-consideration

Important Information for Online Assessments

- Online Exam Preparation Checklist
- Technical Issues
- Fit to Sit
- Time zones

How to apply

1. Prepare your supporting documentation
2. Apply through myUNSW

Please also read the check list provided by UNSW Student Services & Systems, **click here** .

## During the Exam

In case you have any queries during the exam, please send an email to `cs2511@cse.unsw.edu.au` .

If there is a correction, we will post a notice on the class webpage. So please check your email and also check the class webpage for possible corrections (if any) during the exam period.

---

## Part 1 (of 2): Multiple Choice (16 marks)

There are ?? multiple choice questions. The marks for each question may vary. The marks for each question may vary. Part 1 uses Moodle Quiz module. You can attempt multiple times. Your final selections will be marked. Make sure to "**Finish attempt**" and also "**Submit all and Finish**".

Go to Part 1: Multiple Choice Questions

## Part 2 (of 2): Programming Questions (24 marks)

There are three questions in this part. You can use Vlab or your own machine. You need to submit your answer files using the provided instructions in the respective questions (using the provided give command or via WebCMS). All answers must be submitted online.

### Part 2: Q1 (8 marks)

Consider a hotel booking system with the following requirements.

- Hotels consist of a set of numbered rooms. Hotels have a name (e.g. "Meriton Suites Sydney"). Hotels are added automatically when the first room for that hotel is added.
- Hotel Managers can add a room to their hotel. Rooms have a room number, and size (small, medium or large).
- Customers can request, change and cancel bookings.
- Booking requests have a request number, starting date, checkout date, and request for 1 room of a single size (small, medium or large).
- Booking changes have the request number of the booking to be changed, and the new starting date, checkout date, and request for 1 room of a single size (small, medium or large).
- A booking/change request is either granted in full or is completely rejected by the system (a request cannot be partially fulfilled).
- Customers can cancel a booking by providing the request number of the booking.
- Hotel Managers can display the occupancy of all rooms at the specified hotel in order of room declarations, then date.

To remove any ambiguity, booking requests and changes are fulfilled as follows: each hotel is checked (in order of definition) to determine whether the room request can be satisfied, and if so, the first available room (again in order of definition in the input) is assigned to the booking request. The system should not try to fulfil requests by allocating medium rooms when a small is requested, or by reassigning rooms to different bookings or hotels to create vacancies, etc.

Booking dates are inclusive - so if you have a booking for a room starting on the 3rd

January with checkout date of 5th January, the earliest date you can start another booking for this room is the 5th January.

Download the following zip file, it contains the Java package for this question.

Click here to download part2Q1.zip

Place all classes and interfaces you create for this question in that package (**don't** create subpackages). You need to submit all your java files (for classes and interfaces) using the WebCMS or `give` command described below.

Given the above problem specification, your task is to write a solution with skeleton classes. They need only contain:

- Fields
- Stub methods
- Brief comments describing what the methods do, and what the fields and classes represent

**Your code must compile, but does not need to execute. Your main class should be HotelBookingSystem (the *main* method may be empty).**

Your methods may be stubs (non-working methods with fake return values so the code compiles), for example:

*public int getNumDays(){*

*return 0;*

*}*

You do not need to supply any tests.

Your design should be for **backend** classes (no frontend). You do not need to take into account authentication/authorization.

Where there are ambiguities, you may decide how to resolve them (for example, you may choose the type for a booking **id** - *String*, *int* are both acceptable).

You need to submit the required files using the provided give command or via WebCMS.

Once you are satisfied with your solution, submit it via WebCMS. (click here) **OR** use the following `give` command (from a vlab terminal),

```
$ give cs2511  part2Q1  *.java
```

## Part 2: Q2 (8 marks)

Download the following zip file, it contains the Java package for this question.

Click here to download part2Q2.zip

The interface `Hamper<E>` is for hampers that can handle elements of a generic type E. A hamper is similar to a set but allows for duplicate elements. A "hamper" in real life is a basket of gifts full of lots of nice things, e.g. fruits, chocolates, toys, etc...

Complete all methods marked TODO in `ArrayListItemHamper`, `CreativeHamper` and `FruitHamper`. The class `ArrayListItemHamper` uses an `ArrayList` of `Count<E>` to track the count of each element. Pay careful attention to the contract and documentation in `Hamper`, as well as the invariants in `ArrayListItemHamper`, `CreativeHamper`, and `FruitHamper` to make sure your implementation is correct. Make sure that your solution successfully passes the tests in `TestHamper`.

`CreativeHamper` is a hamper for which the price of the hamper is $10 more than the sum of the prices of the items inside it. A `CreativeHamper` which contains 5 or more items, must have at least 2 toy cars (at least 1 must be a premium toy car), and at least 2 fruits. When adding items to a `CreativeHamper`, if this condition cannot be adhered to, the items should not be added.

`FruitHamper` is a hamper of fruits for which the price of the hamper is 25% more than the sum of the prices of the fruits inside it (if the result of this is not an integer, the result should be rounded up to the nearest dollar). A `FruitHamper` which contains 6 or more fruits, must have at least 2 avocados, and at least 2 apples. When adding items to a `FruitHamper`, if this condition cannot be adhered to, the items should not be added.

Note that all currency values in this question and the starter code are in Australian dollars.

You need to submit the required files `ArrayListHamper.java`, `CreativeHamper` and `FruitHamper` using the following `give` command or via WebCMS.

The tests used in automarking will be much more extensive than the JUnit tests provided in the dryrun and starter code. Thus, you should test your code thoroughly according to the details in this specification and the requirements/conditions outlined in the starter code.

You should not rely on any modifications to code other than in those files specified to be submitted - only these 3 files can be submitted.

Once you are satisfied with your solution, submit it via WebCMS. (click here) **OR** use the following `give` command (from a vlab terminal),

```
$ give cs2511  part2Q2  ArrayListHamper.java CreativeHamper.ja
va FruitHamper.java
```

## Part 2: Q3 (8 marks)

Download the following zip file, it contains the Java package for this question.

Click here to download part2Q3.zip

For a quick reference, the interfaces/classes from the package are listed below. Note that, most of them are partially implemented.

```
Circle    ShapeColourAreaVisitor ShapeVisitable
Rectangle ShapeColourVisitor     ShapeVisitor
Shape     ShapeGroup             Triangle
```

In this question, you must use the Visitor pattern (as discussed in the lectures) to implement your solution. Make sure that your solution successfully passes the given JUnit tests. Please note that we will use additional tests to extensively test your solution.

**Tasks:**

Using the Visitor pattern (as discussed in the lectures):

1. Complete the interface `ShapeVisitor` for the classes that implement `ShapeVisitable` in the code `part2Q3`.
2. Implement the required method 'accept' in the classes `Circle`, `Rectangle`, `Triangle` and `ShapeGroup`.
3. Implement the visitor class `ShapeColourVisitor` that sets the colour of circles to red, rectangles to green, and triangles to blue.
4. Implement the visitor class `ShapeColourAreaVisitor` that can be used to calculate total area covered by shapes of a given colour.

The tests in `ShapeVisitorTest` will guide you in completing these. You should ensure your solution passes these tests. Note that you **must** use the visitor pattern (as discussed in lectures) for the above tasks. Otherwise, you will not be awarded marks for this question, even if you pass some/all the tests provided.

You need to submit the required seven files (`ShapeVisitor.java`, `Circle.java`, `Rectangle.java`, `Triangle.java`, `ShapeColourVisitor.java`, `ShapeColourAreaVisitor.java` and `ShapeGroup.java`) using the provided give command or via WebCMS.

Once you are satisfied with your solution, submit it via WebCMS. (click here) **OR** use the following `give` command (from a vlab terminal),

```
$ give cs2511  part2Q3
```

End