

Comp2014 Object Oriented Programming

Lecture 13

Unit Review

and Sample exam questions

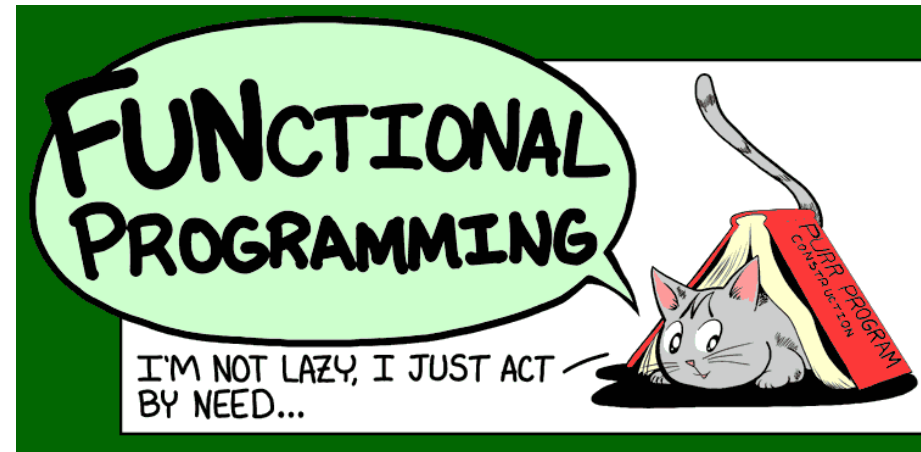
What are covered today

- ◆ Review of topics that were covered in the unit
- ◆ Structure of final exam
- ◆ Sample exam questions
- ◆ Review of Assignment 2
- ◆ Announcements

Topics covered by the unit

◆ Review of fundamental concepts of programming

- Variable declaration and **variable scopes**
- Input and output
- Control logic:
 - » **branching and looping**



◆ Functions

- Function declaration and definition
- Function calls and returns
- Parameter passing: *call-by-value* and *call-by-reference*
e.g., void getMove(Board b, int& x, int& y)
- Function overloading: *f(int)*, *f(double)*, *f(int, int)*
- Default arguments: *f(int i=0, double j=2.3)*

call with a pointer

Topics covered by the unit

◆ Arrays

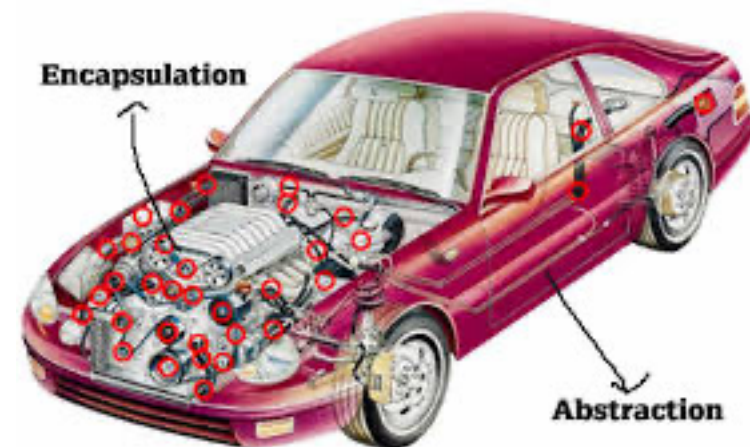
- Declaration: `DataType a[]` in C++ and `DataType[] a` in Java;
- Array name is a **pointer**.
- Data access using index **or** pointer.
- Common process in an array: *find maximum and minimum, average, linear search, binary search, sorting and etc.*
- Pass an array into a function: *its name and size*
- Return an array from a function: *static array can be returned as a pointer*
- Multi-dimensional array: *array of arrays*
- Dynamic array: *memories are from heaps and size is flexible*
 - » *vector in STL is a typical dynamic array*
`vector<ClientRequest> requests;`
`vector<Package> packages;`

Topics covered by the unit

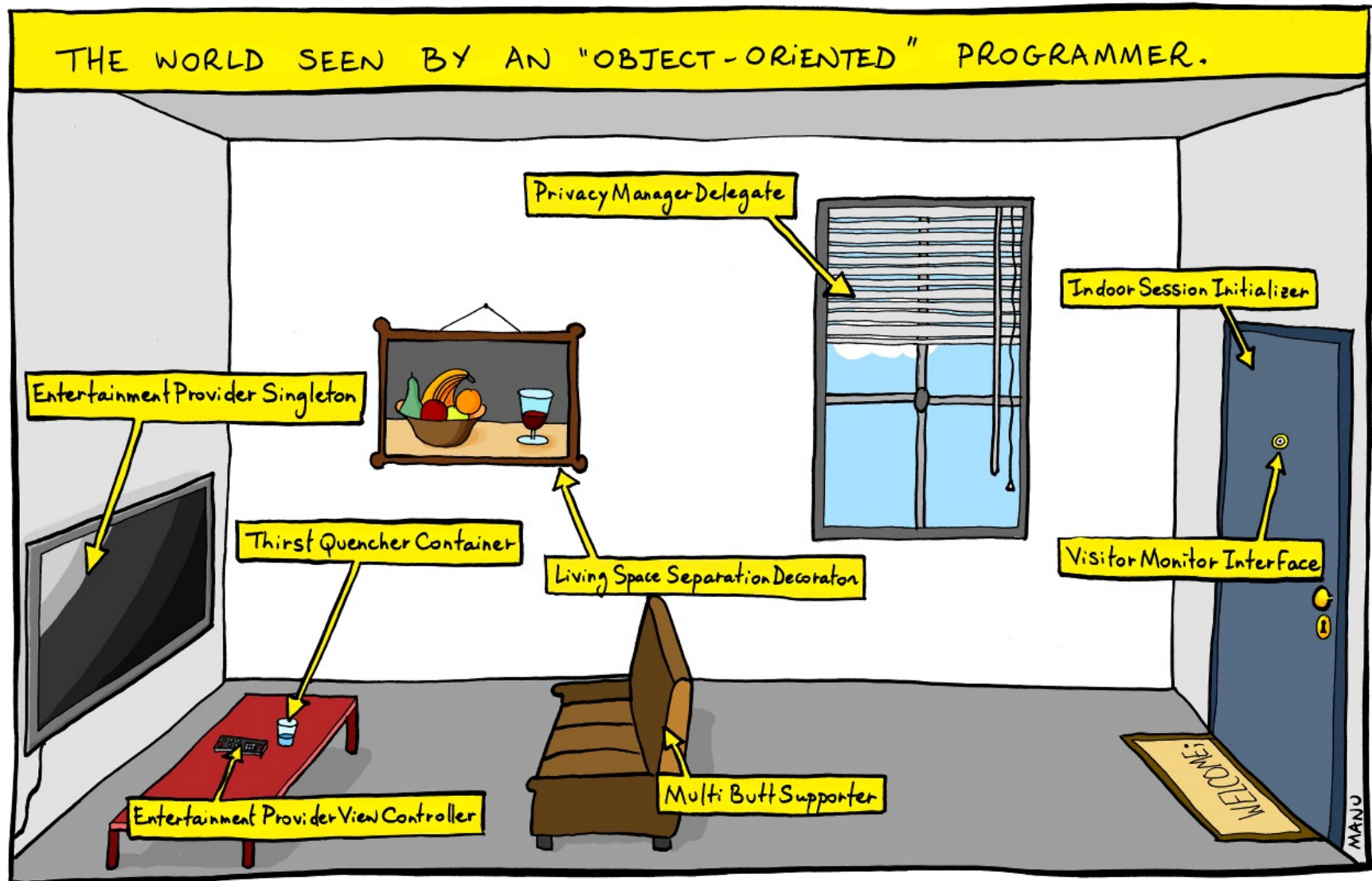
Abstraction
Encapsulation
Data hiding

◆ Objects and classes

- Data abstraction: **abstract data type (ADT)**
- Data encapsulation: **data items and methods**
- Data hiding: **private, protected and public**
- Build a class: declaration and definition
- Class applications: create objects (variables are objects).
- Constructor and destructor



Thinking in OO: abstraction



Picture was acquired from the Internet

Thinking in OO: encapsulation

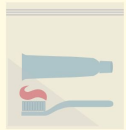
GUIDE TO EFFICIENT PACKING

NICETO HAVES/NEEDS

Small accessories,
sunglasses, phone
charger and mp3player



7.
Pack toiletries in
separate bag



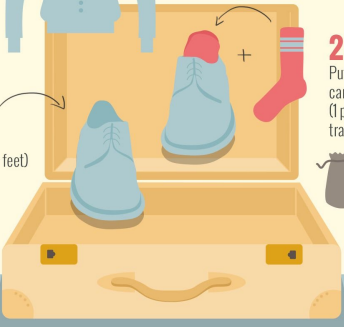
5.
Roll up clothes (max. 1
upper element for a
day and 1 bottom for
each second day)



3.
If necessary, put some
warmer item as next
(jacket, sweater)



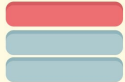
1.
First tlayer consists
of one pair of shoes
(Second pair is on your feet)



8.
Passport, tickets store
in separate, easy
accessible pocket



6.
On top of them, place
your folded items.



4.
Place underwear in
separate bag (1 pair of
underwear for 1
traveling day)



2.
Put as many socks as you
can inside those shoes
(1 pair of socks for 1
traveling day)

NICETO DO:
Put Shoes in
Cloth Bag



CREATED BY:
'SERIOUSLY WE THINK FREE' MAGAZINE



Methods
(less memory!!!)

Data members
(occupy
memory!!!)

Picture was acquired from the Internet

Thinking in OO: data hiding



Topics covered by the unit

- ◆ **Static variables:** in a function or in a class
 - Unique to a function (independent to function calls, initialise once)
 - Unique to a class (share by all the objects of the class)
- ◆ **Reference:** an alias of a variable, not a copy of object but referred to the same object it is assigned to.
- ◆ **File I/O:** input/output to/from iostream
 - Make use of redirection operators << and >>, which automatically recognize built-in data types.

```
ClientRequest request; char c;  
Fin >> request.budget >> c >> request.hotelType >> c;  
while (c != ']') {  
    int eid;  
    fin >> eid;  
    request.events[eid] = true;  
    fin >> c;  
}
```

9020, 4 [14, 2, 7]

Topics covered by the unit

◆ Strings

- C-style strings
- String class
- Conversion

◆ Operator overloading

- Make the operators that have been used for built-in data type also available for the data types you define.
- Typical ones that may need to be overloaded: `<`, `==`, `<<`, `++`, `+=`, `[]`, ...
- Overload an operator only if you feel it is helpful.

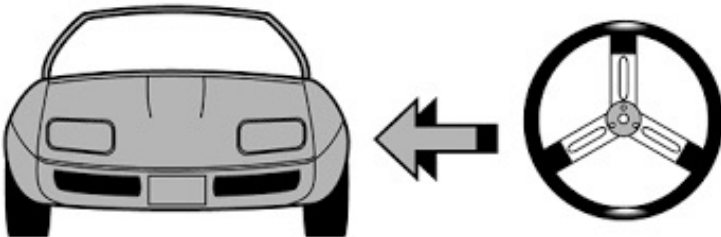
```
bool operator<(const ClientRequest& cr) const {  
    if (budget > cr.budget) return true;  
    return false;  
}
```

Topics covered by the unit

inheritance

- ◆ **Composition:** *part-of* or *has-a* relation
- ◆ **Inheritance:** *is-a* relation
 - Class declaration: use **:** for *extends*
 - Inheritance type: *private*, *protected* and *public*
 - Access control: *how to use base class's members*
 - Constructors and destructors
 - Type conversion: *upcasting* (always possible) and *downcasting*
 - Class hierarchy: a tree of “is-a” relations (base classes on top)

A Car has a Steering Wheel



Thinking in OOP

- Derived classes of Ticket:

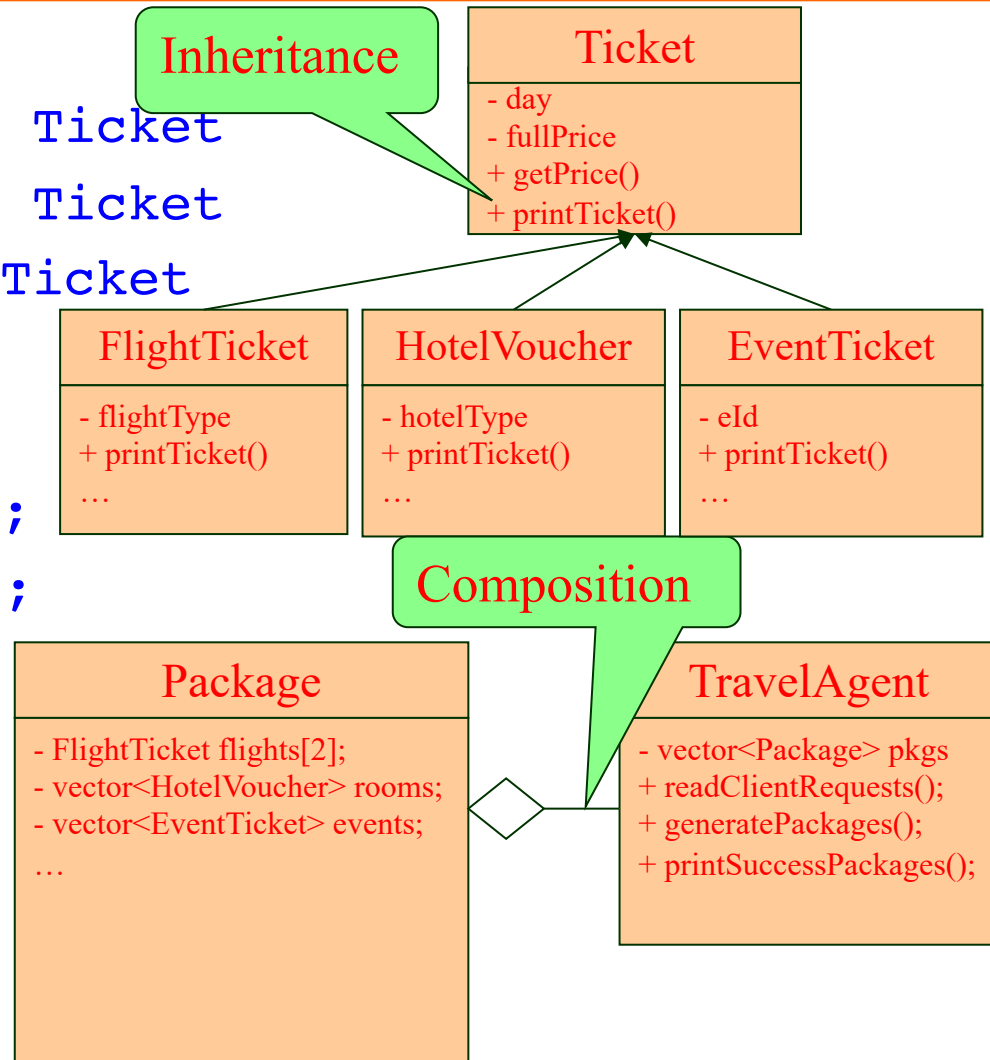
```
class FlightTicket: public Ticket
class HotelVoucher: public Ticket
class EventTicket: public Ticket
```

- A travel package contains:

```
FlightTicket flights[2];
vector<HotelVoucher> rooms;
vector<EventTicket> events;
```

- A travel agent does:

```
readClientRequests();
generatePackages();
printSuccessfulPackages();
```



Unified Modeling Language: UML

Topics covered by the unit

◆ Polymorphism

- Static binding
- Dynamic binding
- Virtual functions
- Purely virtual functions and abstract classes

polymorphism

virtual



Player

```
- playerType  
- Board b  
+ getMove(Board, int&, int&)
```

SmartPlayer

```
+ getMove(Board, int&, int&)  
...
```

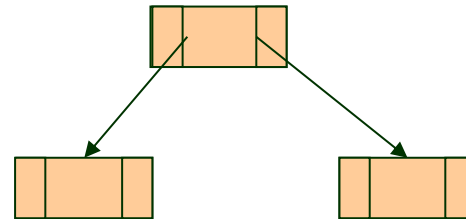
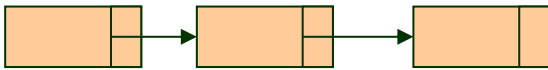
Game

```
- Board b  
- Player players*[2];  
+ Game(Board b, Player* p1,  
        Player* p2)  
...
```


Topics covered by the unit

◆ Linked list

- The simplest data structure other than array
- More efficient for insertion and deletion
- Fundamental for understanding tree structures
- Typical operations in a linear structure
- Efficiency of operations with different implementations



Topics covered by the unit

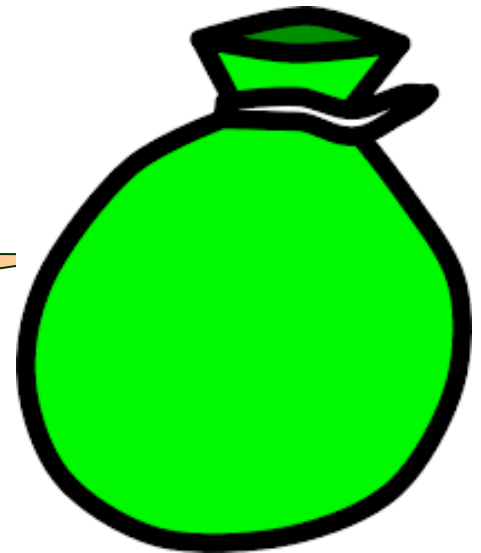
◆ Templates

- Function template
- Class template

◆ Standard Template Library

- Containers: *sequential containers, associative containers and container adaptors*
- Iterators
- Algorithms

Templates are mostly used
for generic data structures.



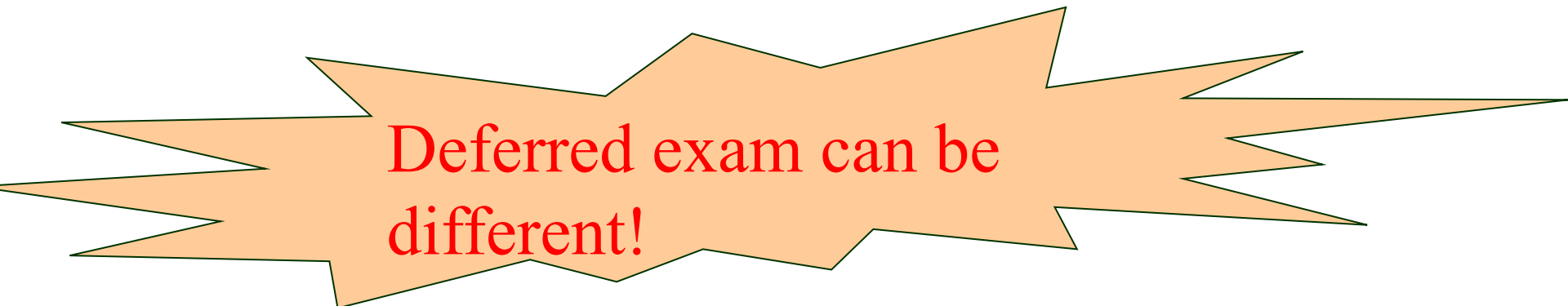
Format of final exam

- ◆ Due to the COVID-19 Pandemic, the final examination in this semester will be run online again via zoom and vUWS.
- ◆ All students who have enrolled this unit are required to take the final examination during the university scheduled time (unless a deferred examination is approved).
- ◆ The students are requested to connect via zoom to the unit coordinator's zoom id 30 minutes before the examination starts. The connection should remain with video showing students face until the end of examination.
- ◆ The students must check out with the teaching team to make sure all submissions have been recorded by the system.
- ◆ The whole examination will be recorded for future reference.

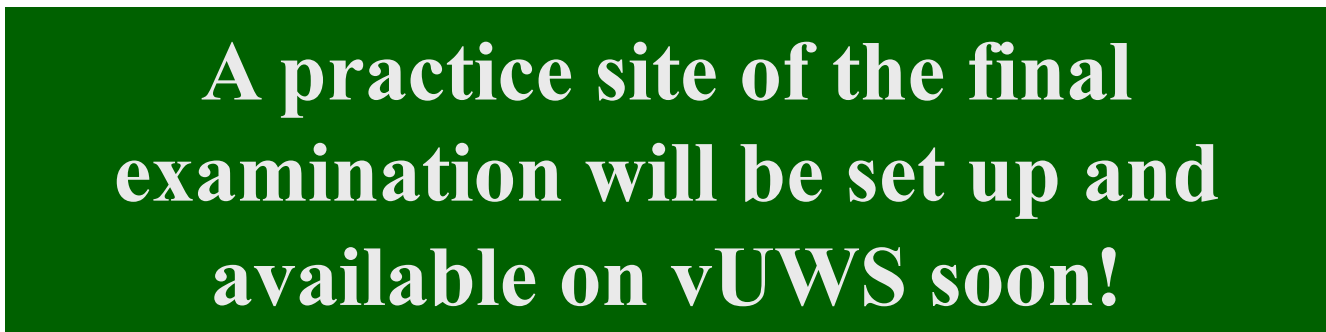
9:30-11:30am 7 Nov 2022

Format of Final Exam

- ◆ **Part A:** multiple choice questions (30 questions 1 mark each)
- ◆ **Part B:** Programming (5 tasks for 20 marks)

An orange starburst graphic with multiple points, containing the text "Deferred exam can be different!".

Deferred exam can be different!

A solid green rectangular box containing the text "A practice site of the final examination will be set up and available on vUWS soon!".

A practice site of the final examination will be set up and available on vUWS soon!

Format of final exam

◆ Multiple choice questions (30 marks):

- 30 questions, one mark each.
- Designed for being completed within 30-45 minutes.
- Completed on vUWS at the first hour, automatically closed at the end of the first hour. No access after that.
- Questions are taken from a big pool (120 questions) and presented one question each time in random order. You would expect some typos with the multiple choice questions. Screenshot any possible errors and email me **after the examination**. I will investigate it and adjust marks accordingly.

Format of final exam

◆ Programming tasks (20 marks):

- The programming part consists of five **tasks** for 20 marks in total.
- Designed to be completed within 75 minutes.
- You can only use C++ to complete the tasks.
- The code for the programming tasks must be submitted on vUWS by the end of the examination (will be given an additional 10 minutes for submission).
- Only source code is needed for submission. It can be based on any IDE. If we can't run your code, we will contact you but we will mainly focus on your implementation logic.

Format of final exam

- You will be required to submit two identical versions: one **for Turnitin checking** and the other one **for documentation and marking**.
- The submission to the Turnitin site is for similarity verification. If the similarity of two submissions is higher than a threshold, the related students will be requested to demonstrate their work to a tutor during a scheduled time after the examination.

Format of deferred exam

- ◆ The deferred examination will be in the same format as the final examination except that every student is required to demonstrate their code for the programming tasks.
- ◆ As long as the campuses are accessible to students, the deferred exam will be held at the campus.
- ◆ Students with AIP will take examination at the same time but will be possibly given longer time for completion based on AIP specification.

Source of the questions

- ◆ **Lecture notes:** demonstrated code and basic concepts.
- ◆ **Practical questions:** practice for both multiple choice questions and programming
- ◆ **Online tutorials:** practice for multiple choice questions
- ◆ **Assignment tasks:** practice for programming
- ◆ A few multi-choice questions might be out of teaching scope.

Sample questions: Part A

Part A Multiple-choice questions

Which of the following statements is INCORRECT?

- A. A constructor is a special kind of member function. It is automatically called when an object of that class is declared.
- B. A constructor has the name of its class name.
- C. A constructor can return any type of values.
- D. A constructor can have multiple parameters.

Sample questions: Part A

Part A Multiple-choice questions

Which of the following statements is INCORRECT?

- A. A constructor is a special kind of member function. It is automatically called when an object of that class is declared.
- B. A constructor has the name of its class name.
- C. A constructor can return any type of values.
- D. A constructor can have multiple parameters.

Sample question: Part A

If a variable is declared as _____ data member in a class, then it is common to all the objects of the class.

- A. static
- B. const
- C. public
- D. private

Sample question: Part A

If a variable is declared as _____ data member in a class, then it is common to all the objects of the class.

- A. static
- B. const
- C. public
- D. private

Sample questions: Part A

Given the following program, which of the class member accesses in the `main ()` function are **LEGAL**?

```
class DayOfYear {  
public:  
    void input();  
    string output();  
    // other public members  
private:  
    int month;  
    int day;  
    // other private members  
};
```

```
int main() {  
    DayOfYear birthDay;  
    birthDay.input();           // a)  
    birthDay.day = 25;         // b)  
    cout << birthDay.output(); // c)  
    if(birthDay.month == 1)    // d)  
        cout << "January\n";  
}
```

- A. b) and d)
- B. a) and c)
- C. a), b) and c)
- D. None of the above

Sample questions: Part A

Given the following program, which of the class member accesses in the `main ()` function are **LEGAL**?

```
class DayOfYear {  
public:  
    void input();  
    string output();  
    // other public members  
private:  
    int month;  
    int day;  
    // other private members  
};
```

```
int main() {  
    DayOfYear birthDay;  
    birthDay.input();           // a)  
    birthDay.day = 25;         // b)  
    cout << birthDay.output(); // c)  
    if(birthDay.month == 1)    // d)  
        cout << "January\n";  
}
```

- A. b) and d)
- B. a) and c)**
- C. a), b) and c)
- D. None of the above

Part B

Download the program `BaseProgram.cpp` from. Convert it into OO-Style by making the following changes on the original code:

Task 1 (5%): Change the definition of `CDAccount` from `struct` to `class`. Make all the data members of the class to be private. Move the free function `inputDate(Date& theDate)` and `getCDDData(CDAccount& the Account)` in the class as public member functions of the class (remove the parameters of function `getCDDData`)

Task 2 (5%): Change the data member `double balanceAtMaturity` of class `CDAccount` into a public member function `double balanceAtMaturity()` to calculate the balance at maturity (simply move the related code from the main function of the original program to implement the function).

Part B: Programming tasks

- ◆ **Task 3 (5%):** Add a new public member function, *printAccount()*, to class `CDAccount` to print the information of a CD account (you may also simply move the related code from the main function of the original code).
- ◆ **Task 4 (5%):** Make appropriate changes on the whole program to guarantee the new program does the same as the original code.

Look ahead

- OOP is just the beginning towards professional programmer
- Other units for further programming training:
 - 300103 Data Structures and Algorithms
 - 300167 Systems Programming 1
 - 300130 Internet Programming
 - 300698 Operating Systems Programming
 - 300115 Distributed Systems and Programming
 - 300960 Mobile Applications Development
 - 300165 Systems Administration Programming
 - 300582 Technologies for Web Applications
- AI Major: M3110 Major in Artificial Intelligence
- Units related to Artificial Intelligence:
 - 301174 Artificial Intelligence
 - 300093 Computer Graphics
 - 300569 Computer Security
 - 301205 Robotic Programming