hrs	/ /20		Exams Office Use Only

University of the Witwatersrand, Johannesburg

Course or topic No(s)	ELEN3009
Course or topic name(s) Paper Number & title	Software Development II
Examination/Test* to be held during month(s) of (*delete as applicable)	November 2017
Year of Study (Art & Sciences leave blank)	Third
Degrees/Diplomas for which this course is prescribed (BSc (Eng) should indicate which branch)	BSc(Eng)(Elec)
Faculty/ies presenting candidates	Engineering and the Built Environment
Internal examiners and telephone number(s)	Dr SP Levitt x77209
External examiner(s)	Mr J Lewis
Special materials required (graph/music/drawing paper) maps, diagrams, tables, computer cards, etc)	Computer card for multiple-choice questions

Time allowance

Course Nos	ELEN3009	Hours	3

Instructions to candidates
(Examiners may wish to use this space to indicate, inter alia, the contribution made by this examination or test towards the year mark, if appropriate)

a) Read instructions on page 1 of exam
b) Available marks: 110 - Full marks: 100
c) Closed-book exam
d) Basic scientific calculator allowed

Internal Examiners or Heads of School are requested to sign the declaration overleaf

1. As the Internal Examiner/Head of School, I certify that this question paper is in final form, as approved by the External Examiner, and is ready for reproduction.

2. As the Internal Examiner/Head of School, I certify that this question paper is in final form and is ready for reproduction.

(1. is applicable to formal examinations as approved by an external examiner, while 2. is applicable to formal tests not requiring approval by an external examiner—Delete whichever is not applicable)

Name:_____ Signature:

(THIS PAGE NOT FOR REPRODUCTION)

Instructions

- Answer *all* questions. The questions do not carry equal weight.
- The paper is divided into two parts:
  - **Part A** consists of several multiple choice questions which are to be answered on the computer card provided. You must fill in your student number on the card. There may be more than one correct answer for each of the multiple choice questions, for example (a) and (e). Indicate ALL the correct answers for each question by clearly marking the appropriate blocks of the chosen letters on the card with a dark HB pencil. A negative marking system will be adopted so **DO NOT GUESS**. Marks will be subtracted for incorrect answers. You cannot, however, get less than zero for any single multiple choice question.
  - **Part B** consists of three (3) questions to be answered legibly in the answer book provided.
- For questions which require you to write **source code**, note that:
  - Pencil may be used.
  - You only need to specify `#include`'s if specifically asked.
  - For classes, you can give the implementation entirely in the header file, unless directed otherwise.
  - Marks are not awarded solely for functionality but also for good design; making appropriate use of library functions; following good coding practices; and using a modern, idiomatic C++ style.
  - Your code must be easily understandable or well commented.
- Reference sheets are provided in a separate appendix.

Part A

Question 1

1.1 Which of the following statements concerning Git and GitHub are true? (5 marks)

(a) Git is a centralised version control system. F

(b) Assuming that you have a linear commit history, changing any commit in the history will cause the commit hashes of all of the subsequent commits to change.

(c) The `rebase` command allows you to combine commits together.

(d) The `commit` command pushes code contained in a local repository up to a remote repository.

(e) A pull request on GitHub is primarily used for code review. T

1.2 Which of the following statements are true? (5 marks)

(a) Static member functions cannot be overloaded.

(b) Static member functions can only access static data members and other static member functions.

(c) Static data members can only be accessed by static member functions.

(d) Non-static data members can be accessed by static member functions.

(e) The output of the following program is: `1 2 3`

```
class Player
{
private:
    int id;
    static int next_id;
public:
    int getID() { return id; }
    Player()  {  id = next_id++; }
};

int Player::next_id = 1;

int main()
{
  vector<Player> players(3);
  cout << players[0].getID() << " ";
  cout << players[1].getID() << " ";
  cout << players[2].getID() << " ";
  return 0;
}
```

1.3 Which of the following statements are true about the STL? (5 marks)

(a) All containers provide iterators which support being incremented with ++, dereferenced with $*$, decremented with $--$, and compared against another iterator with `!=`.

(b) The symmetrical range $[m, n)$ has $n - m$ elements.

(c) The STL library includes an `array` container.

(d) Erasing an element from a vector reduces its size.

(e) The memory layout and structure of a container determines the functionality of the iterators that it provides.

1.4 Which of the following statements, related to exceptions, are true? (5 marks)

(a) Custom exception classes can be defined to express the reason for the error.

(b) Exceptions should not be thrown or caught in a normal run of a program.

(c) Exceptions report errors found at compile time and run time.

(d) Exceptions thrown in a constructor will immediately cause the object's destructor to be run.

(e) A post-condition violation means that although a function is given valid inputs it is unable to meet the expectations of the calling code.

1.5 Which of the following statements are true? (5 marks)

(a) The terms *free store* and *heap* are equivalent.

(b) Smart pointers are intended to be created on the stack and not on the heap.

(c) Calling `delete` on a pointer only deletes the pointer, not what is it pointing to.

(d) It is safe to call `delete` multiple times on a pointer without any side effects, if the pointer is set to `nullptr`.

(e) A smart pointer cannot prevent memory leaks if an exception is thrown.

1.6 Which of the following statements, related to *Spartan programming*, are true? (5 marks)

(a) Programming in this style means using fewer variables.

(b) The `const` keyword can be used to support this programming style.

(c) Using range-based for loops instead of ordinary for loops is in keeping with a Spartan programming style.

(d) The use of C-style arrays rather than STL containers is in keeping with a Spartan programming style.

(e) Preferring to use variables allocated on the stack rather than `static` variables conforms to this style.

1.7 SFML is a gaming library which is available for use in C++. Which of the following statements concerning SFML are true? (5 marks)

(a) When setting up a C++ project which makes use of SFML there is no need to add the SFML header files to the project in the IDE, if the include path to the header files has been specified in the project settings.

(b) You need to specify the location of the SFML library files for the pre-processor.

(c) The functionality provided by SFML is part of the `std` namespace.

(d) Linking with the SFML library files is done after the code is compiled.

(e) Dynamically linking to the SFML libraries means that the size of the executable using SFML is smaller than if it were statically linked to the libraries.

1.8 Which of the following statements concerning Figure 1 are true? (5 marks)

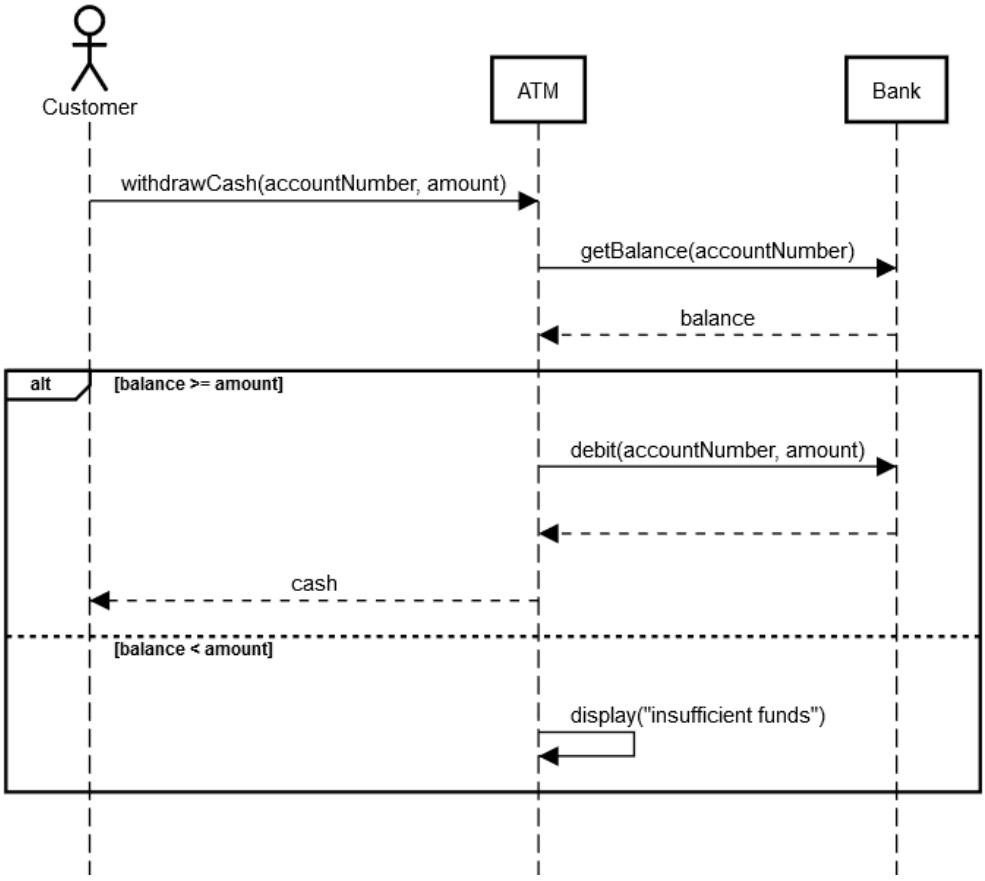

Figure 1

(a) Figure 1 is a type of *interaction diagram*.

(b) The vertical dashed lines are called *timelines*.

(c) Drawing the return arrows is optional.

(d) The `display` method may be either `private` or `public`.

(e) The `getBalance` method may be either `private` or `public`.

[Total Marks Part A: 40]

Part B

Question 2

Examine the code for the `Employee`, `Payroll` and `TelephoneGuide` classes in Listings 1 and 2 and answer the following questions:

a) Name two code smells (indicators of poor design) that apply to these classes. Specifically indicate where the code smells lie, and explain why they are poor design. (6 marks)

b) *Refactor* the code, that is, change the structure but not the external behaviour, to remove the code smells. Refactoring implies that any client code which uses the above classes must continue to work after the changes.

For your answer, you need only write down the code that you modify. (10 marks)

```
using TelephoneNumbers = vector<string>;

class Employee
{
public:
   Employee(const string& name, const string& address):
     _name(name),
     _address(address)
     {}

     void addTelephoneNumber(const string& number)
     { _telephone_numbers.push_back(number); }

     string name() const
     { return _name; }

     string address() const
     { return _address; }

     TelephoneNumbers getTelephoneNumbers() const
     { return _telephone_numbers; }

private:
   string _name;
   string _address;
   TelephoneNumbers _telephone_numbers;
};
```

Listing 1: `Employee` class

```
class Payroll
{
public:
   void createEmployeeLabel(const Employee& employee)
   {
      _label += employee.name() + "\n";
      _label += employee.address() + "\n";
      for (auto e : employee.getTelephoneNumbers())
      {
         _label += e;
         _label += " "; // space used as a separator
      }
      _label.erase(_label.length()-1, 1);
   }

   string getLabel() const { return _label; }

private:
   string _label;
};

class TelephoneGuide
{
public:
   void setCurrentEmployee(Employee* employee)
   { _employee = employee; }

   string formatTelephoneNumbers() const
   {
      string tel_list;
      tel_list += _employee->name() +=  "\n";
      tel_list += _employee->address() +=  "\n";
      auto tel_numbers = _employee->getTelephoneNumbers();
      auto it = begin(tel_numbers);
      for (; it != tel_numbers.end(); it++)
      {
         // dashes used as a separator
         if (it != begin(tel_numbers)) tel_list += "--";
         tel_list += *it;
      }
      return tel_list;
   }

private:
   Employee* _employee;
};
```

Listing 2: `Payroll` and `TelephoneGuide` classes

[Total Marks 16]

Question 3

The School of Electrical and Information Engineering requires a software system for managing final year projects during the laboratory course. This system will consist of a graphical user interface (GUI), C++ core logic and a database.

Part of the system's core code is a `Project` class which represents a final year project and stores all information related to it. It does not interact *directly* with either the GUI or the database; however, other system components must be able to retrieve the information related to a project. Each project has the following information associated with it:

- The project's title.
- A description of the project.
- The name of the project supervisor.
- The names and student numbers of the students doing the project. Students work in pairs on the project.
- A digital photograph of the project which is taken on open day.

A photograph of each project is only taken when the project is complete, whereas the rest of the information is known upfront, before the course begins. Occasionally, the project's title and description changes during the course.

`Project` objects will be contained in other parts of the system in `vectors`. To facilitate this, these objects must be able to be copy constructed and assigned correctly.

A class called `Photo` *already* exists which is responsible for containing the photograph associated with a project. It has the following declaration:

```
class Photo {
public:
    Photo(const string& photo_file_name);   // reads an existing photo from disk
    Photo(const Photo& rhs);
    Photo& operator=(const Photo& rhs);
    ...
};
```

Listing 3: `Photo` class

Design and fully implement the `Project` class (making use of the `Photo` class). Make sure that your class meets the system requirements that have been mentioned. Feel free to create any additional helper classes or functions that will improve your design.

[Total Marks 14]

Question 4

Consider the following scenario. You are shopping in a supermarket and putting items into your shopping cart one-by-one.

The supermarket has various ways of pricing items which are as follows:

Simple Pricing
The total cost is calculated simply by adding up the cost of each individual item purchased.

Three-for-Two Promotions
Buy three of any particular item, and pay for only two. This promotion is specific to the type of item being sold. For example, buy three bottles of All Gold Tomato Sauce, and pay for only two.

Combo Deals
A discount is applied when a specific combination of items is purchased.

It would be handy if you could know the total cost of all of the items in your cart at any point in time. The following classes (Listings 4 and 5) model this scenario. Your tasks are as follows:

a) Given the code in Listings 4 and 5, write a number of unit tests to thoroughly verify that the `ShoppingCart` class is behaving as expected. (20 marks)

b) Provide *all the source code* for your own solution to this problem. You are free to use the public interfaces provided and modify them to suit your needs. You may also discard them entirely. You can create any additional classes that you need. (15 marks)

c) Comment on a noteworthy strength or weakness of the solution that you have given. (5 marks)

```
class Item
{
public:
   Item(string name, double price);
};
```

Listing 4: `Item`'s public interface

```
class ShoppingCart
{
public:
   ShoppingCart();
   void addItem(Item item);
   double total(); // returns the total cost of all items in the cart, with any
       discounts already applied
};

void createThreeForTwoPromotion(Item& item);
void createComboDeal(Item& first_item_in_combo, Item& second_item_in_combo, double
    discounted_total_price);
```

Listing 5: `ShoppingCart`'s public interface and discount setup methods

[Total Marks 40]

[Total Marks Part B: 70]

(Exam Total: Four Questions – 110 marks : Full Marks – 100 marks)
