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 2019
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 K Ortlepp
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 When writing code, it is considered good practice to: (5 marks)

(a) Rely on comments to explain how a function is implementing its task.

(b) Name types using nouns, and functions using verbs.

(c) Describe the return value type within the function name so that programmers have a better understanding of the function.

(d) Use abbreviations in variable and function names.

(e) Phrase functions that return a Boolean value as questions.

1.2 Given the following program, which of the statements concerning it are <u>true</u>? (5 marks)

```
1 int calculate_length(const string& word) { return word.length(); }
2
3 int main()
4 {
5    auto words = vector<string>{"This", "is", "a", "short", "sentence"};
6    auto word_lengths = vector<int>(words.size()); // for results
7    transform(begin(words), end(words), begin(word_lengths),
         calculate_length);
8    return 0;
9 }
```

(a) An in-place transformation can be used instead of storing the results in `word_lengths`.

(b) The `transform` function does not call the `calculate_length` function when the iterator moving over the range is equal to: `end(words)`.

(c) The following line of code is equivalent to that on line 6:
`auto word_lengths = vector<int>{words.size()};`

(d) `calculate_length` is known as a *function object*.

(e) The implementation of the `transform` function does not use `vector<T>::push_back`.

1.3 Which of the following statements are <u>true</u>? (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.

1.4 SFML is a gaming library which is available for use in C++. Which of the following statements concerning SFML are <u>true</u>? (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) SFML provides inheritance hierarchies which can be extended (derived from) for your own application.

(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 larger than if it were statically linked to the libraries.

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

(a) The next C++ standard will be approved in 2020.

(b) Popular compilers like Microsoft's Visual C++ and GCC fully conform to the current standard.

(c) The standard C++ libraries include a number of standard C libraries.

(d) C++ is known as a *systems* language because it offers direct access to the hardware of a machine.

(e) The use of the `auto` keyword allows C++ type checking to occur at run-time.

1.6 Which of the following statements are true about constructors and destructors? (5 marks)

(a) Constructors can only be `public`.

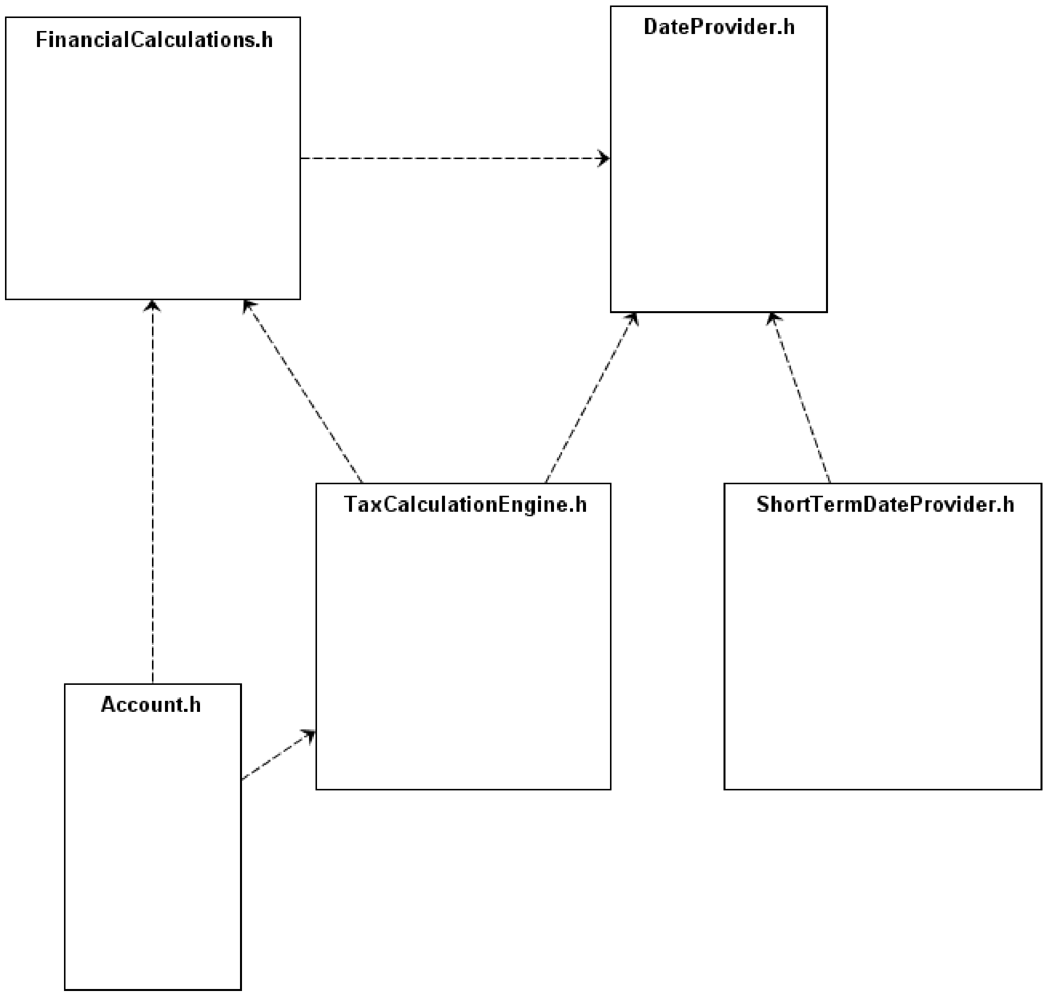
(b) A class may only have one constructor.

(c) A class may only have one destructor.

(d) Default constructors can be invoked without supplying arguments.

(e) Derived classes should always provide their own destructor.

1.7 Examine the following diagram depicting dependencies among C++ *header* files.



Which of the following statements concerning the diagram are <u>true</u>? (5 marks)

(a) Header include guards will prevent the linker from multiply including `FinancialCalculations.h` in `Account.h`

(b) The diagram correctly depicts the dependency between `DateProvider.h` and `ShortTermDateProvider.h`, given that class `ShortTermDateProvider` inherits from class `DateProvider`. Assume that `DateProvider` is declared in `DateProvider.h` and `ShortTermDateProvider` is declared in `ShortTermDateProvider.h`.

(c) Code in `TaxCalculationEngine.h` and `TaxCalculationEngine.cpp` is able to make use of `ShortTermDateProvider` objects.

(d) Code in the `Account` module makes use of financial calculation functions but it is preferable not to include `FinancialCalculations.h` in `Account.h` as it will be included anyway via `TaxCalculationEngine.h`.

1.8 Examine the following C++ code:

```cpp
void aFunction(const vector<int>& v)
{
    auto a = begin(v);
};

class Person {};
class Student: public Person {};

int main()
{
    auto b = 17.0;

    auto c = static_cast<int> (b);

    string default_value{""};
    auto d = default_value;

    auto e = new Student{};

    return 0;
}
```

Which of the following statements concerning the variables a to e are <u>true</u>? (5 marks)

(a) a is of type `vector<int>::iterator`

(b) b is of type `double`

(c) c is of type `int`

(d) d is of type `string`

(e) e is of type `Person*`

[Total Marks Part A: 40]

Part B

Question 2

Examine Listing 1 and answer the following questions. Each question is *independent of the others* and assumes, as a starting point, the code that is given in Listing 1.

```
class LibraryItem {
public:
   LibraryItem(int total_copies): total_copies_{total_copies}, copies_on_loan_{0} {}
   void returnItem() { copies_on_loan_--; }
   void borrowItem() { copies_on_loan_++; }
   int availableCopies() { return total_copies_ - copies_on_loan_; }

private:
    int total_copies_;
    int copies_on_loan_;
};

class Book: public LibraryItem {
public:
   Book(const string& title, const string& author, int copies):
       LibraryItem{copies}, title_{title}, author_{author} {}
   string title() { return title_; }
   string author() { return author_; }

private:
  string title_;
  string author_;
};

class DVD: public LibraryItem {
public:
   DVD(const string& title, int runtime, int copies): LibraryItem{copies},
       title_{title}, runtime_{runtime} {}
   string title() { return title_; }
   int runtime() { return runtime_; }  // runtime in minutes

private:
  string title_;
  int runtime_;
};
```

Listing 1: `LibraryItem` hierarchy

a) Given the code in Listing 1, it is only possible to construct a `Book` and `DVD` by specifying the total number of copies:

```
auto b = Book{"The Great Gatsby", "F.Scott Fitzgerald", 4};
auto d = DVD{"Star Wars: Episode V - The Empire Strikes", 124, 3};
```

Provide all modifications to the above code to also allow for the construction of these library items without specifying the total number of copies, as shown below.

```
auto b2 = Book{"Advanced Control Engineering", "Roland S Burns"};
auto d2 = DVD{"Avengers: Endgame", 182};
```

By default, the total number of copies must be set to one, and the number of copies-on-loan to zero. *You must make use of in-class initializers*. (5 marks)

b) Add functionality to the `LibraryItem` class which allows the total number of *all* library items on loan, at any point in time, to be tracked and queried. (5 marks)

c) i) List the invariants that apply to the `LibraryItem` class. You should make reasonable assumptions in this regard. (3 marks)

ii) Assume the following scenario: There is a single team responsible for both creating and using the `LibraryItem` class hierarchy. The team is absolutely certain that the clients of the hierarchy do not violate `LibraryItem`'s invariants. Nevertheless, you wish to code defensively. Give all the modifications that you would make to `LibraryItem` in order to detect errors related invariant violations. (6 marks)

d) Is the use of inheritance a good design decision in this situation? Explain your answer. (4 marks)

e) Refactor the class hierarchy to use composition instead of inheritance. The client code, which is given in Listing 2 and uses the existing hierarchy, must be able to use the refactored code without having to be modified. (8 marks)

```
auto b = Book{"The Great Gatsby", "F.Scott Fitzgerald", 4};
cout << b.author() << endl;
auto d = DVD{"Star Wars: Episode V - The Empire Strikes", 124, 3};
cout << d.runtime() << endl;

b.borrowItem();
d.borrowItem();
b.returnItem();

cout << d.title() << ": " << "available copies: " << d.availableCopies() <<
    endl;
cout << b.title() << ": " << "available copies: " << b.availableCopies() <<
    endl;
```

Listing 2: Client code

[Total Marks 31]

Question 3

Two team members are using GitHub to work collaboratively on a project. Figure 1 shows a sequence of Git commands made by each of the members over time. Each arrow points to the repo that is affected by the particular command. The sequence of commands is numbered from the first command (1) until the last (11).

Draw a graphical representation of the commit histories for team member 1's repo, team member 2's repo, and the shared GitHub repo. Clearly indicate all commits (using the appropriate sequence number from the diagram), branches, and HEAD for each of the repos. Assume that one or more files are committed with each `commit` command, and that there are no merge conflicts.

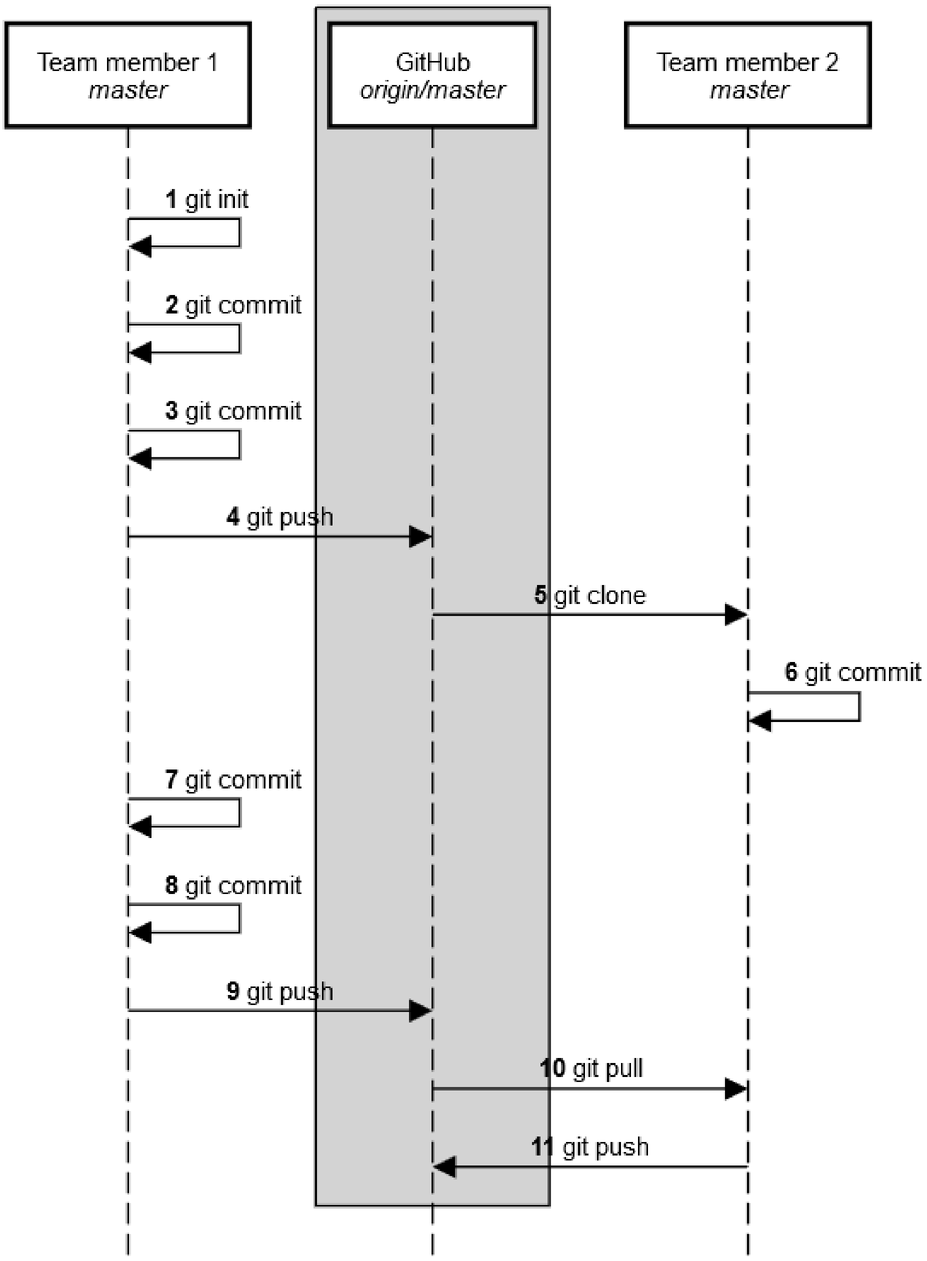


Figure 1: Git command sequence

[Total Marks 9]

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 the same item, and pay for only two. For example, buy three bottles of All Gold Tomato Sauce, and pay for only two. Multiple three-for-two promotions, on different items, may be active concurrently.

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 (Listing 3 and Listing 4) model this scenario. Your tasks are as follows:

a) Given the code in Listing 3 and Listing 4, write a number of unit tests to thoroughly verify that the `ShoppingCart` class is behaving as expected. (15 marks)

b) Provide *all the source code* for your own object-oriented 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. Your solution will be evaluated in terms of how well it conforms with good OO design principles (information hiding, small classes, etc). You may assume that, in future, different types of promotions will come into effect, such as combo specials (reduced prices on certain combinations of items). (15 marks)

```cpp
class Item
{
public:
   Item(string name, double price);
   string name() const  { return _item_name; };
   double price() const  { return _price; };
};
```

Listing 3: `Item`'s public interface

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

// Creates promotion, accepts item on promotion and cart to apply promotion to
void createThreeForTwoPromotion(Item& item, ShoppingCart& cart);
```

Listing 4: `ShoppingCart`'s public interface and discount setup method

[Total Marks 30]

[Total Marks Part B: 70]

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

Please fill in the question numbers on the front page of your script.