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 2016
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 are true? (5 marks)

(a) That next C++ standard will be approved in 2017.

(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.

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

(a) Git is a centralised version control system

(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.

1.3 Which of the following statements concerning the code given below are true? (5 marks)

```
1  int main()
2  {
3      auto a{make_shared<int>(2)};
4      shared_ptr<int> b{make_shared<int>(2)};
5      shared_ptr<int> c{new int{2}};
6      auto d{new int{2}};
7
8      return 0;
9  }
```

(a) Lines 3 and 4 are functionally equivalent aside from the name of the variable.

(b) This program will not leak memory if an exception is thrown in the `make_shared` function called on line 4.

(c) This program has a memory leak when it runs without any exceptions being thrown.

(d) More memory is used by the memory allocations performed on line 4 than those on line 5.

(e) Variable `d` is allocated on the heap.

1.4 Which of the following statements are true about exceptions? (5 marks)

(a) Throwing an exception has a greater performance cost than returning an error code.

(b) C++ does not allow you to nest `try-catch` blocks.

(c) The `new` operator can throw an `out_of_memory` exception.

(d) An exception should never occur during the execution of a well designed program.

(e) Exception objects are not required for catching exceptions.

1.5 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.

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

(a) `string::size_type` represents a kind of magic number.

(b) The following line of code will produce an output of "6": `cout « 12.5/2;`

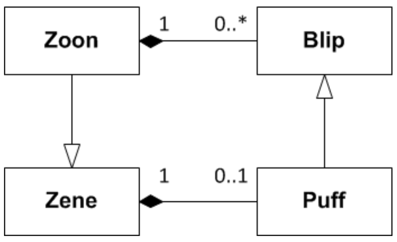
(c) "Magic numbers" are also known as "literals".

(d) The output of the following program is the maximum value that the `int` type can hold.

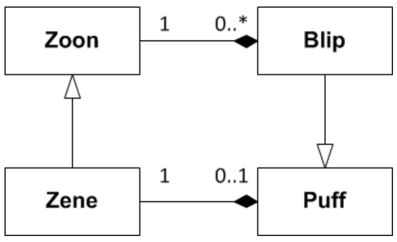
```
#include <iostream>
#include <climits>

int main()
{
   int x = INT_MIN;
   std::cout << --x;
   return 0;
}
```

1.7 All Zoons are Zenes. Zenes may or may not have a Puff. A Puff is a Blip, and a Zoon may have any number of Blips. Review the UML diagrams below and identify which of the following statements are correct.

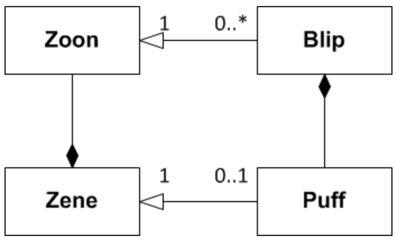
(5 marks)



(a)



(b)



(c)

(a) The UML diagram labelled (a) depicts the given scenario.

(b) The UML diagram labelled (b) correctly depicts the given scenario.

(c) The UML diagram labelled (c) correctly depicts the given scenario.

(d) Aggregation is a form of composition that implies sole ownership.

(e) The relationship between Zoon and Zene in diagram (c) implies navigability from Zoon to Zene.

1.8 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.

[Total Marks Part A: 40]

Part B

Question 2

a) The classes and main program given in Listings 1 and 2 in the appendix compile and run. Give the output of the main program. (14 marks)

b) Why can the use of `protected` data members be problematic? (3 marks)

c) In computer programming the "queue" is a very important data structure. A queue can be thought of as being similar to a line of people waiting to pay in a supermarket. A queue has a front and a back. Data enters the queue at the back and leaves at the front.

Create a class which models a queue of integers and give the complete implementation. Ensure that you provide methods for adding items to the queue and for removing items from the queue. There should also be a method which determines if the queue is empty or not.

You may assume that objects of this class will only store a small number of integers at any one time so performance in terms of speed is not a critical issue.

(8 marks)

[Total Marks 25]

Question 3

Suppose that you are creating an object-oriented design for a 2D arcade game. Everything on the screen is a game object and has a height and width. Some game objects are capable of moving and some are capable of shooting — these behaviours are independent of one another. So there are:

- Stationary, non-shooting game objects
- Moving, non-shooting game objects
- Stationary, shooting game objects
- Moving, shooting game objects

Different game objects can shoot and move in different ways. For example, in the case of moving objects there are vertically moving objects and horizontally moving objects.

a) Give two *different* designs which model this scenario. For each design provide a UML class diagram and a brief description which clearly explains your approach. (12 marks)

b) Compare your two designs by describing their advantages and disadvantages, and provide a justification for the one that you would ultimately choose. (8 marks)

[Total Marks 20]

Question 4

Tic-tac-toe (also known as noughts and crosses) is a game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. The game ends in a draw if neither player is able to achieve a winning row after all spaces in the grid have been filled.

The following example game, as illustrated in Wikipedia, is won by the first player, X:

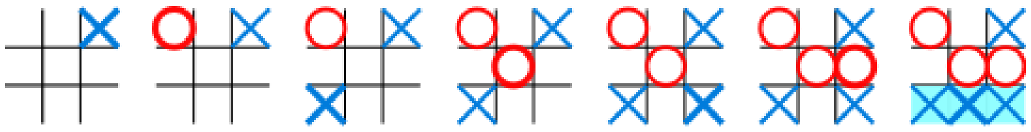


Figure 2: Tic-tac-toe

Provide the complete source code for a Tic-tac-toe game which allows two players to play against each other.

The game must be played from the *command-line*. You need to allow each player to take their turn by having them input the position at which they will make their mark, and you must display the updated game grid after each turn. Output a message when the game is over indicating who won, or if the game was a draw.

Your solution must be object-oriented and you should separate the presentation layer from game logic.

[Total Marks 25]

[Total Marks Part B: 70]

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

Appendix

```
class Sport
{
public:
    Sport(string name): name{name}
    { _number_of_sports++; }

    string official() { return "Referee"; }

    virtual void print() { cout << name << " " << official() << endl; }

    static int _number_of_sports;

protected:
    string name;
};

int Sport::_number_of_sports = 0;

class Rugby : public Sport
{
public:
    Rugby(): Sport{"Rugby"},
             _number_of_players{15}
    {}

    virtual void print() override
    { cout << name << " " << _number_of_players << endl; }

    int _number_of_players;
};

class Sevens : public Rugby
{
public:
    Sevens()
    { _number_of_players = 7; }
};

class Hockey : public Sport
{
public:
    Hockey(): Sport{"Hockey"},
              _number_of_players{11}
    {}

    string official() { return "Umpire"; }

    int _number_of_players;
};
```

Listing 1: Sports class hierarchy

```
int main()
{
    using sport_ptr = shared_ptr<Sport>;
    vector<sport_ptr> sports;
    sports.push_back(make_shared<Hockey>());
    sports.push_back(make_shared<Sport>("Croquet"));
    sports.push_back(make_shared<Sport>("Running"));

    auto a{make_unique<Rugby>()};
    auto b{make_unique<Sevens>()};
    auto c{make_unique<Hockey>()};
    auto d{sports[0]};

    cout << "1. " << sports[0]->official() << endl;
    cout << "2. " << c->official() << endl;
    cout << "3. " << d->official() << endl;
    for(auto i : sports) i->print();
    cout << "a. "; a->print();
    cout << "b. "; b->print();
    cout << Sport::_number_of_sports << endl;

    return 0;
}
```

Listing 2: Main program