University of Swaziland

Department of Computer Science

CSC242 — Object Oriented Programming

Final Examination

May 2017

Instructions

1. The time allowed is **THREE (3) HOURS**.
2. Read all the questions in **Section A** and **Section B** before you start answering any question.
3. Answer **all** questions in Section A. Answer **any two** questions of Section B. Maximum mark is 100.
4. Use correct notation and show all your work on the answer script.

DO NOT OPEN THIS PAPER UNTIL YOU ARE INSTRUCTED TO DO SO BY THE INVIGILATOR

Section A

Question 1 [25]

i What is a *class*? What is an *object*? And what is the relationship between a *class* and an *object*? [3]

ii Name four differences between a *class* and a *structure*? [4]

iii What is the *default access specifier* of a class? [1]

iv What is a function *call by reference*? [2]

v What are the properties of a *constructor*?[4]

vi What is the difference between *overload* and *override*? [2]

vii Name and explain three basic concepts of object oriented programming (OOP) [6].

viii What is a ternary (conditional) operator? With a suitable example statement to demonstrate how it is used in C++. [2]

ix What is a *precondition* of a function? [1]

Question 2 [25]

a Explain the following function prototype. [5]

char *get(***const int** & *x,* **int** *y)* **const**;

b Given the following constructor implemention of the class *Car*

```
Car(int x = 0, char y = 't', bool z = 0)
{
   // class member variables initializing code added here
}
```

Which object decleration statement is legal? [3]

(a) *Car thisCar();*

(b) *Car thisCar;*

(c) *Car thisCar(4);*

c Name 2 built in operations on classes. [2]

d Using a suitable example, draw a UML diagram to demonstrate composition and inheritance. You may add explanations where neccessary.[6]

e Given the following BASE class.

```
class BASE
{
private:
   int x
protected:
   float y
public:
   bool z;
}
```

We are also given the following class called DERIVED defined as follows.

```
class DERIVED: public BASE
{
   ...
}
```

State whether each of the Base class member variables, x, y and z are accessible in following scopes? [9]

(a) in objects of DERIVED type at the DERIVED class scope

(b) in objects of DERIVED type, neither the DERIVED class scope nor the BASE class scope.

(c) in objects of BASE type, that are neither the DERIVED class scope nor the BASE class scope.

Section B

Question 3 [25]

a Given the following declaration:

```
int num;
int *ptr1, *ptr2;
double *ptr3;
```

Mark the following statements as valid or invalid. If a statement is invalid, explain why. [10]

i *ptr1 = ptr2;*

ii *num = ptr1;*

iii **ptr3 = *ptr2;*

iv *ptr1 = &ptr2*

v *num = *ptr2;*

vi *ptr1 = num;*

vii *num = &ptr1;*

b Write down 3 operators that cannot be overloaded in C++.[3]

c Consider the following declaration:

```
class stranger
{
  ...
};
```

i Write a statement that shows the declaration in the **class** *stranger* to overload the **operator** >>. [2]

ii Write a statement that shows the declaration in the **class** *stranger* to overload the binary **operator** + as a member function. [2]

iii Write a statement that shows the declaration in the **class** *stranger* to overload the **operator** <= as a **friend**. [2]

d With suitable code examples, demonstrate the difference between *deep copy* and *shallow copy* of data. [4]

e Name two situations in which a copy constructor executes. [2]

Question 4 [25]

a Explain the following:[6]

i pointer *this*

ii *friend* function

iii operator overloading

b Write down the syntax for function template declaration and for class template decleration. [2]

c Consider the following declaration:

```
template <class type> class stranger
{
  ...
private:
   type a;
   type b
};
```

i Write a statement that declares *sObj* to be an object of type *strange* such that the **private** member variables *a* and *b* are of type **int**. [2]

ii Write a statement that shows the declaration in the **class** *strange* to overload the operator == for the **class** *strange* as a member function. [2]

iii Assume two objects of type *strange* are equal if their corresponding member variables are equal. Write the definition of the function operator == for the **class** *stranger*, which is overloaded as a member function. [4]

d Consider the following statement:
int * *num;*

i Write the C++ statement that dynamically create an array of 10 components of type **int** and *num* contains the base address of the array.[2]

ii Write a C++ code that inputs data into the array *num* from the standard input device.[4]

iii Write the C++ statement that deallocates the memory space of array of which *num* points.[3]

Question 5 [25]

a Give two reasons why do we place the class definition file in the header file, and the definition of member functions in the implementation file? [2]

b For class templates, why does the mechanism of separation of header files and implementation file not working? [2]

c A class definition is sometimes written as follow:

```
#ifndef H_LIST //First Line
#define H_LIST //Second line
class LIST
{
  ...
};
#endif //Last Line
```

Explain the role played by the inclusion of the commented 3 lines. [3]

d In C++, what is the notation for the *scope resolution operator*? [1]

e What is a *mutator function*? And what is an *accessor function*? [2]

f Consider the following base class definition.

```
class Base
{
   double bx;
   char cx;
public:
   Base(char c = 'a', double b = 0.0 );
};
```

The defintion of Base constructor is as follows;

```
Base::Base(char c, double u)
{
   bx = u;
   cx = c;
}
```

Let *Derived* be a class derived from *Base*. Assume *Derived* inherits all members from *Base*, and *Derived* has a single member variable *dx* which is not inherited from Base.

i Write down a class definition of *Derived*. [3]

ii Write a constructor function definition (implementation) for the class *Derived*. [4]

g Mark the following statement as *true* or *false*. [4]

i The order in which **catch** blocks are listed is not important.

ii All exceptions need to be reported to avoid compilation errors.

iii One way to handle an exception is to print an error message and exit the program.

iv An exception can be caught either in the function where it occured or in any of the functions that led to the invocation of this method.

h Explain the following function prototype, for a method in a class called Object. [3]

```
*Object Comp(const Object & obj);
```