SENECA COLLEGE OF APPLIED ARTS AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY – SY

MIDTERM TEST

SEMESTER	SUBJECT NAME	SUBJECT CODE
WINTER 2017	Introduction to Object Oriented Programming	OOP244

NAME: _____

STUDENT NUMBER: _____

SECTION: OOP244SAB

DATE: Wednesday Feb 22nd 2017

TIME ALLOWED: 1.5 Hours

MARK DISTRIBUTION

Walkthrough	20 MARKS
Concept and Definitions	40 MARKS
Coding	77 MARKS
TOTAL MARKS	137
BONUS	12

PROFESSOR: Fardad Soleimanloo

SPECIAL INSTRUCTIONS:

1. Closed Book, One reference sheet printed, double-sided allowed, with student id and name.
2. Write your answers in one or more exam booklets

This test includes a *cover page*, plus 6 pages of *questions*.

SENECA'S ACADEMIC HONESTY POLICY:
As a Seneca student, you must conduct yourself in an honest and trustworthy manner in all aspects of your academic career. A dishonest attempt to obtain an academic advantage is considered an offense, and will not be tolerated by the College.

1- [7 marks total]
- What is a namespace? Explain (no coding); [2 Marks]
- What is the syntax to create and use a namespace? [3 mark]
- If during implementation there are two or more namespaces with the exact same name in different files, how does the compiler handle those [2 marks]?

2- [16 marks total] write which of the aspects of object orientated programming and programming in general relate to each of the statements below:

Aspects in question:
Inheritance, Encapsulation, Polymorphism, Abstraction

Statements:

1- Airplane and Pigeon can both fly.
2- Car is a vehicle
3- A Car has 4 wheels, a steering wheel, a brake, a gas pedal and an Engine. Steering wheel, brake and gas pedal are used by the driver to control the engine and the wheels indirectly.
4- A programming teacher's job is to teach programming; it is not important for me if he can dance!
5- All Mammals reproduce the same way.
6- This printer can print on paper, Envelops and post cards.
7- Motorcycle is essentially a bicycle with an engine.
8- My weekly schedule is a table with days as columns and time as rows.

3- [9 marks total] Explain what modular programming is [3 marks].

Without code, explain how you would define all of the information for a C++ class called "Student" and what would be names of the files that you would create [2 marks].

What would be the common code in those files [2 marks]

How would you ensure that the information is compiled only once? [2 marks]

4- [14 marks total] Consider the following statements:
Foo A;
Faa B;
Fee C;

What are the types and what are the instances?(2 marks)

Write the operator overload signatures for the following, if many implementations are possible, write only one of them. Write the signatures only, do not write the implementations.

Example:
A = B; // void Foo::operator=(const Faa&);

Write the signatures for the following: (2 marks each)

A = B + C;

!C;

A = B -= C;

A = ++C;

B = B++;

C = B = A;

5- [6 marks total] C++ types are divided into 2 categories. Name the categories [2 marks] and explain the difference between them. [4 marks]

6- [18 marks total] Write a program (only main()) to get a user-specified number of doubles and then print them marking the largest one. Use the following algorithm:

- Prompt the user for the number of doubles [1 mark]
- Allocate an array of doubles accordingly [3 marks]
- Ask the user to enter them one by one and store them in the allocated array of doubles. [2 marks]
- Go through the array and save the index of the largest one. [3 marks] (see hint below if you don't know how to do this)
- Print them back. [1 mark] When printing add "<-- Largest" beside the largest number in array. [1 mark]
- Cleanup your code. (2 marks)
- Assume that the user enters input correctly and that your computer has enough memory.

Program essentials [5 marks]

Hint for finding the index of the largest element:
Have an integer variable for the index of the largest value (let's call it "lix") and set it to the first element in the array (i.e zero). Loop through the rest of the array comparing each element with the element at the index "lix", if the current element is larger than the one at the index "lix", update "lix" to the index of the current one. When loop is done "lix" will hold the index of the largest double in the array.

```
Output sample:
Please enter number of doubles values: 4
Please enter the values:
1> 12.34
2> 34.56
3> 34.44
4> 1.3
Values entered:
1: 12.34
2: 34.56 <-- Largest
3: 34.44
4: 1.3
```

7- [17 marks total + 7 bonus] Define a class called "Subject". (Do not implement it)

Subject Specs: (2 marks)
The Subject has:

- Title (length unknown) (1 mark bonus)
- Subject code, (6 characters long) (1 mark)
- Mark (1 mark)

Instantiation

- The Subject can be created with no initial values (will be set to a safe empty state) (1 mark)
- The Subject can be created using 3 arguments to set all the attributes, but if the mark is not provided, it will be set to zero. [4 marks]

Capabilities

- The Subject can print itself. (2 marks)
- Read and fill its attributes from the keyboard (1 mark)
- Set and get the mark attribute safely (3 marks)
- An integer can be added to the subject to increase the mark: [2 marks]
- The Subject can safely be set to another Subject or get Copied. [6 marks bonus]

8- [30 marks total + 5 bonus] The following class encapsulates a Bottle that contains m_quantity of liquid in a bottle with maximum capacity of m_capacity

```
#include <iostream>
class Bottle {
   int m_capacity; // in CC, maximum capacity of the bottle
   int m_quantity;  // in CC the current quantity of liquid in the bottle,
                    // valid values: 0 <= m_quantity <= m_capacity, or else -1
public:

};
```

m_capacity setting rule: Anywhere in the code where m_quantity is to be set to a value , the value must be between 0 and m_capacity inclusive, otherwise it should be set to -1

Write the following:

- Setter and getter for m_quantity; Setter sets the value of m_quantity, following the above rule. And the getter returns the value of m_quantity. [4 marks]
  hint: After this, reuse the setter to set the value of quantity throughout the implementation.
- A Getter for m_capacity returning its value. [2 marks]
- A method called empty that receives and returns nothing. It just sets the quantity to zero. [1 mark]

- A constructor that receives two arguments for m_quantity and m_capacity and sets them. If the quantity argument is not provided it should be set to zero. If the capacity argument is not provided, it should be set to 500 CCs. [4 marks]

  Example:

  Bottle B; sets quantity to 0 and capacity to 500

  Bottle B(100); sets quantity to 100 and capacity to 500

  Bottle B(50,200); sets quantity to 50 and capacity to 200

- A "+" operator to accommodate the following: [4 marks]
  if Bottle A(??,??), B(??,??) , C(??,??):
  A = B + C;

  A will be a Bottle with the sum of the quantities of B and C and a capacity of 500. B and C remain unchanged.

- A "+=" operator to accommodate the following [4 marks]
  if Bottle A(??,??), B(??,??) , C(??,??):
  A = B += C;

B will be a Bottle with the sum of the quantities of B and C and capacity of B is unchanged. A will be a copy of B afterwards. C remains unchanged.

- A prefix "++" operator that accommodates the following: [4 marks]
  if Bottle A(??,??), B(??,??):
  A = ++B;
  B will have 1 CC more than before in it quantity. A will be a copy of B afterwards.

- A postfix "++" operator that accommodates the following: [4 marks]
  if Bottle A(??,??), B(??,??):
  A = B++;
  B will have 1 CC more than before in it quantity. A will be a copy of the original B before the adding one.

- Overload the double cast and return the double division of quantity by capacity. (this value is always between 0 and 1 by design. [3 marks]

- Bonus: Overload the "<<" operator so that the Bottle can be printed using cout. The output format should be as follows: [5 marks bonus]
  if Bottle A(24,300):
  cout << A << endl;
  should print: >24CC/300[NEWLINE]<

9- Determine the exact output of the following program [20 marks]:

```
#include <iostream>
using namespace std;
struct Object {
   int data;
   Object(int val = 2) {
      cout << (data = val);
   }
   ~Object() {
      cout << data;
   }
};


void w2(const char* str) {
   cout << str << endl;
}
void w2(int i = 10, char c= '-') {
   for (int j = 0; j < i; j++) {
      cout << c;
   }
   cout << endl;
}

void w3(char* str) {
   char* chs[6];
   int i, one = 1;
   for (i = 0; i < 5; i++) {
      one = -one;
      chs[i] = &str[(one*i) + 10];
   }
   chs[i] = &str[12];
   for (i = 0; i < 5; i++) {
      cout << *chs[i];
   }
   cout << " " << chs[i] << "!" << endl;
}


int& w1(int& R) {
   R -= 10;
   return R;
}

int main() {
   Object A(1), B, C = 3;
   int i = 10;
   int& R = i;
   w1(R) += 10;
   cout << endl << i << endl;
   cout << ((&i == &R)? "Yes" : "No") << endl;
   w2(2, '$');
   w2(5);
   w2();
   w2("Now what?");
   w3("XBEGEFDXOJ!GMOO");
   return 0;
}
```