Midterm Test

Seneca | SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

TERM	COURSE NAME	COURSE CODE	VERSION
Summer 2017	Introduction to Object-Oriented Programming	OOP244	B

Name	
Student Number	
Section	SCC

DATE: Tuesday, April 18, 2017

TIME ALLOWED: 1.5 Hours

QUESTIONS:

PART A	Programming	~~21.5~~ 19.5	MARKS
PART B	Walkthrough	15	MARKS
PART C	Concepts & Short Answers	25	MARKS
	TOTAL	~~61.5~~ 59.5	

PROFESSOR: Fardad Soleimanloo

SPECIAL INSTRUCTIONS:

1. A reference sheet, computer printed, double-sided, size A4 or letter is permitted and must be submitted with the test. **NO** other aids allowed.
2. Write your answers in the spaces provided.

This Test includes a *cover page*, plus 10 pages of *questions* and one *blank page*.

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. I understand this policy, student signature: _____

PART A: PROGRAMMING [~~21.5~~ 19.5 MARKS]

1- [4 marks] Assume there is a library function called sort as follows:
```
void sort(int values[], int size);
```
and it is declared in a headerfile called "sort.h", in a namespace called "tools".
Sort receives the values to be sorted in an array and the size of the array and sorts it.
Complete the following program to get unknown number of integers from user and print them back sorted as follows:
Output:
```
Enter the number of integers: 4
1: 20
2: 34
3: 1
4: 44
1 20 34 44
```
The Program:

```
//includes and namespaces [1 mark]
#include <iostream>
using namespace std;
#include "sort.h"
using namespace tools;
int main() {
   int num, i;
   int* values;
   cout << "Enter the number of integers: ";
    //get the number of integers (num)
   cin >> num;
   // allocate memory (store address in values) [1.5 marks]
   values = new int[num];
   // get the numbers one by one
   for (i =0; i < num; i++) {
     cout << (i + 1) << ": ";
     cin >> values[i];
    }
```

```
    // sort the values using the sort function. [0.5 marks]
    sort(values, num)                              ;

    // now that the arrays is sorted
    // print the elements back space separated
    for (i =0; i < num; i++) {
       cout << values[i] << " ";
    }
    cout << endl;
    // free the allocated memory [1 mark]
    delete[] values                    ;

    return 0;
}
```

2- [7 marks] Design a class definition (no implementation), make sure that the privacy is properly set and member function have meaningful. Attribute names must follow the workshop standards.

NOTE: you are to only write the class definition; that is essentially what you write in a header file. DO NOT IMPLEMENT THE CLASS.

Create a class called a "Student". [1.5 mark]
A Student has following information: [1 mark]

- a name, max 20 chars long
- a student number that is a 6 digit integer
- a Grade Point Average that is a floating point number
- a semester that is a single digit integer.

Constructors and methods:

- A student can get instantiated only in two ways; 1- with no information provided to be set in a safe empty state,2- with a name and a student number. [1.5 marks]
- Grade Point Average and Semester can be set to values using modifier methods. (setters) [1 mark]
- Grade Point average and student number can be accessed through queries. (getters). [1.5 marks]
- A Student can display itself. [0.5 marks]

```
#ifndef SICT_STUDENT_H__  // 0.5
#define SICT_STUDENT_H__
class Student { // 0.5
private:
  char m_name[21];  // 1
  int m_stno;
  double m_gpa;
  int m_semester;
public: // 0.5
  Student();  // 1.5
  Student(const char name[], int stno);
  void gpa(double value); // 1
  void semester(int value);
  double gpa()const; // 1.5
  int stno()const;
  void display()const; // 0.5
};
#endif // !SICT_STUDENT_H__
```

3- [10.5 marks] Complete the following class and fully implement the methods:

```
#include <iostream>
using namespace std;

class Canister { // [3 marks]
   double m_capacity;
public:
   Canister(   double capacity = -1.0    );
   void   display()const        ;
  Canister&    operator +=   (  double capacity  );
   friend Canister    operator  +
         (const Canister& LO, const Canister& RO);
};
// See the end of the question for the tester example
```

A- The Constructor: [1.5 marks]
sets the capacity or sets it to -1 (as safe empty state) if capacity value is not provided.

```
Canister::Canister(double capacity) {
      m_capacity = capacity;
}
```

B- The display method: [2 marks]
if in safe empty state, prints "Empty!" otherwise it prints the m_capacity value in 10 spaces, 2 digits after decimal and padded with zeros at left. Then it goes to new line

```
void Canister::display()const {
  if (m_capacity < 0)
    cout << "Empty!";
  else
    cout << "Cap: " <<
    fixed << setprecision(2) <<
    setw(10) << setfill('0') << m_capacity;
  cout << endl;
}
```

C- Overload the += operator so a double value can be added to a Canister and then return the Canister object. [2 marks]

```
Canister& Canister::operator+=(double capacity) {
  m_capacity += capacity;
  return *this;
}
```

D- Overload the + operator as a helper function that adds the values of two Canisters in a new Canister and then returns it. [2 marks]

```
Canister operator+(const Canister& LO, const Canister& RO) {
  return Canister(LO.m_capacity + RO.m_capacity);
}
```

Your implementation should work with the following main:

```
int main() {
   Canister A = 10.1, B, C = 20.1;
   A.display();
   B.display();
   C.display();
   cout << "----------------" <<endl;
   B = A + C;
   C = B += 5.1;
   A.display();
   B.display();
   C.display();
   return 0;
}
```

Output:

```
Cap: 0000010.10
Empty!
Cap: 0000020.10
----------------
Cap: 0000010.10
Cap: 0000035.30
Cap: 0000035.30
```

PART B: WALKTHROUGH [15 MARKS]

4- What is the output of the following code snippets: [4 marks]

A. `int a = 10; cout << !!a;` _____1_____

B.
```
int foo(int a = 2) { return a + 10; }
cout << foo() << " , " << foo(10);
```
_____12 , 20_____

C.
```
int a = 10; int& r = a;  r += 2;
cout << a << " , " << r ;
```
_____12 , 12_____

D.
```
int a = 10, b = 20, c = 30;
int* p[4] = {&a, &c, &b, &a};
for(int i=0;i<4;i++) cout << *p[i] << "  ";
```
_____10 30 20 10_____

5- Determine the exact output of the following program: [11 marks]

```
#include <iostream>
#include <cstring>
using namespace std;
class Test {
   char data[21];
public:
   Test() {
      cout << "Default" << endl;
      data[0] = 'X'; data[1] = '\0';
   }
   Test(const Test& T) {
      cout << "copy " << T.data << endl;
      strcpy(data, T.data);
   }
   Test(const char* str) {
      cout << "Make " << str << endl;
      strcpy(data, str);
   }
   ~Test() {
      cout << data << " gone!" << endl;
   }
   void prn() {
      cout << data << endl;
   }
   void print() {
      cout << data;
   }
};
```

```
void display(Test T) {
   cout << "Displaying ";
   T.print();
   cout << endl;
}
int main() {
   Test A, B = "BB";
   display(A);
   Test* p = new Test("Dyn");
   p->prn();
   delete p;
   B.prn();
   return 0;
}
```

DEFAULT
MAKE BB
COPY X
DISPLAYING X
X GONE!
MAKE DYN
DYN
DYN GONE!
BB
BB GONE!
X GONE!

PART C: CONCEPTS AND SHORT ANSWER [25 MARKS]

6- Please circle the statements that are CORRECT: [2 marks]
   A. C has no object-oriented support. CORRECT
   B. C++ is C with object-oriented capability, therefore programs written in this language could be object oriented or not. CORRECT
   C. C++ is fully object-oriented. INCORRECT
   D. Java is fully object-oriented. CORRECT

7- In two sentences explain what is a namespace, where do we use them and where do we create them? [3 marks]

Namespace is a scope in which entities are created to prevent name collisions.
It is created and used in cpp and header files but it is only created in a headerfiles and never used in them.

8- Fill in the spaces using some or all of the following words or phrases. (some may be used more than once or not used at all): [14 marks]
an executable, pre-compile, encapsulation, machine code, Modular programming, compiler, polymorphism, compile, linker, Object Oriented Programming, object files, Assembly, Templating, inheritance

   A. __ Modular programming __is to organize all the code for relative tasks and objects into separate files and header files under the name of those tasks and objects.
   B. C++ __Compiler__ works in three stages, first it will _pre-compile_____ the files to interpret all the compiler instructions. These instructions begin with a "#" like "#include". Then it will _compile__ all the CPP files separately into_object files_____. So essentially _the compiler_____ will run to the number of CPP files in your project. Finally, _linker_____ will run to link all the compiled code into_an executable_____.
   C. Object-Oriented Programing is to use __polymorphism___ , __encapsulation_ and __ inheritance _____ to design and code a solution to implement an object to work in the program.
   D. __encapsulation___ is to pack the data and behaviour of an object together in a class.
   E. __polymorphism____ means having many forms(shapes).
   F. Refining, using or expanding an already existing design of a class, to create a new class is called _inheritance____.

9- True or False: [6 marks]

A. Fundamental types are made up of several compound types packed into a package. ___F_____
B. Declaration associates an entity with a type, telling the compiler how to interpret the entity's identifier. _____T_____
C. Including a file is literally copying the contents of the included file where the "#include" statement is written. ____T_____
D. #include "headerfile" is for standard libraries and #include <headerfile> is for custom modules written by programmers. _____F_____
E. The destructor is called right before the object goes out of scope. ___T____
F. To reset an object to its original values, a constructor can be called anywhere like a regular method. ____F_____