FNU FIJI NATIONAL UNIVERSITY

Final Examination

College	Engineering, Science & Technollogy
School	Electrical & Electronics Engineering
Programme	Bachelor of Engineering [Year 2]
Semester	II
Year	2016
Unit Code	EEE661
Unit Title	Introduction to C++ Programming
Date of Examination	TBA
Time	TBA
Venue	TBA
Duration	3 Hours *(extra 10 mins allowed to read the paper)*
Maximum Marks	100

Instructions

1. There are four (5) sections (A - E). Attempt all sections in the answer booklet.
2. Write your answers legibly in the answer booklet.
3. Write your student identification number on each page used.

Section A: Matching *(20 Marks)*

For each term in the column on the left, write the corresponding letter for the description that best matches it from the column on the right.

_____	1. Inheritance	(a)	Class from which others are derived.
_____	2. Derived class	(b)	Deriving from more than one base class.
_____	3. "Has a" relationship	(c)	Class that is created by inheriting from an existing class.
_____	4. "Is a" relationship	(d)	Inheritance.
_____	5. Single inheritance	(e)	Passes arguments to the base-class constructor.
_____	6. Base class	(f)	Composition.
_____	7. Base-class initializer	(g)	Deriving from only one base class.
_____	8. Multiple inheritance	(h)	New classes are created from existing classes.
_____	9. `new` and `delete`	(i)	Process of replacing an inherited base-class member function with a derived-class one.
_____	10. Memory leak	(j)	Function prototypes that end with "= 0".
_____	11. `friend` function	(k)	Class from which objects can be instantiated.
_____	12. `static` class variable	(l)	Class that is defined, but never intended to be used by the programmer to create objects.
_____	13. default constructor	(m)	Enables C++/s operators to have class objects as operands.
_____	14. Dangling pointer	(n)	A constructor that takes as its argument a reference to an object of the same class as the one in which the constructor is defined.

_____	15.	Copy constructor	(o)	Problem that may occur when default memberwise copy is used on objects with dynamically allocated memory.
_____	16.	Operator overloading	(p)	The compiler provides one of these for a class that does not declare any.
_____	17.	Abstract base class	(q)	Used when only one copy of a variable should be shared by all instances of a class.
_____	18.	Concrete class	(r)	Defined outside the class's scope, yet has access to private members of the class.
_____	19.	Pure `virtual` function	(s)	Occurs when objects are allocated but never deallocated.
_____	20.	Override a `virtual` function	(t)	Operators used for performing dynamic memory allocation and deallocation.

Section B: Short Answers *(20 Marks)*

1. How are constructors and functions similar? How are they different? (2)
2. Explain when to use the dot operator (.) and when to use the arrow operator (->). (2)
3. What is the purpose of declaring certain objects with keyword `const`? (1)
4. What are `friend` functions? Where in a class definition is a `friend` function specified? (2)
5. What are `static` data members? Why might they be used? What is their scope? (3)
6. Why are some operators overloaded as member functions while others are not? (2)
7. What is inheritance? (1)
8. What is protected access? (1)
9. What is the difference between single and multiple inheritance? (2)
10. What is meant by redefining a base-class member? How does this process differ from function overloading? (2)
11. What are some of the program-design advantages of using polymorphism? (2)

Section C: Programming Output *(20 Marks)*

For each of the given complete programs or program segments, read the code and write the output.

1. What is the output of the following program? (5)

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

// class Test definition
class Test
{
public:
     Test( int = 0 );
     void print() const;
private:
     int x;
}; // end class Test

// default constructor
Test::Test( int a )
{
     x = a;
} // end class Test constructor

// function print definition
void Test::print() const
{
     cout << x
     << this->x
     << ( *this ).x << endl;
} // end function print

int main()
{
     Test testObject( 4 );
     testObject.print();
} // end main
```

2. What is the output of the following program? (5)

```
#include <iostream>
using namespace std;
class M
{
public:
     M( int );
     int mystery( int );
private:
     int data;
```

```
10      int number;
11 }; // end class M
12
13 // constructor
14 M::M( int q = 0 )
15 {
16      data = q;
17      number = 2;
18 } // end class M constructor
19
20 // function mystery definition
21 int M::mystery( int q )
22 {
23      data += q;
24      return data;
25 } // end function mystery
26
27 int main()
28 {
29      M mObject( 2 );
30      M *mPtr = &mObject;
31
32      cout << mObject.mystery( 20 ) << endl;
33      cout << mPtr->mystery( 30 );
34 } // end main
```

3. What is the output of the following program? (5)

```
1 #include <iostream>
2 using namespace std;
3
4 class M
5 {
6 public:
7      M( int );
8      int mystery( int );
9 private:
10     int data;
11     double number;
12 }; // end class M
13
14 // constructor
15 M::M( int q )
16 {
17     data = q;
18     number = .5;
19 } // end class M constructor
20
21 // function mystery definition
22 int M::mystery( int q )
23 {
24     data += q;
```

```
25      return data * number;
26  } // end function mystery
27
28  int main()
29  {
30      M stuff( 44 );
31      cout << stuff.mystery( 78 );
32  } // end main
```

4. What is the output of the following program? (5)

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  // class Oyster definition
6  class Oyster
7  {
8  public:
9      // constructor
10     Oyster( string genusString )
11     {
12         genus = genusString;
13     } // end class Oyster constructor
14
15     // function getPhylum definition
16     string getPhylum() const
17     {
18         return "Mollusca";
19     } // end function getPhylum
20
21     // function getName definition
22     virtual string getName() const
23     {
24         return "Oyster class";
25     } // end function getName
26
27     // function getGenus definition
28     string getGenus() const
29     {
30         return genus;
31     } // end function getGenus
32
33     // print function
34     virtual void print() const = 0;
35 private:
36     string genus;
37 }; // end class Oyster
38
39 // class VirginiaOyster definition
40 class VirginiaOyster : public Oyster
41 {
```

```
42      public:
43      // constructor calls Base-class constructor
44      VirginiaOyster()
45      : Oyster( "Crassostrea" )
46      {
47          // empty
48      } // end class VirginiaOyster constructor
49
50      // function getName definition
51      virtual string getName() const
52      {
53          return "VirginiaOyster class";
54      } // end function getName
55
56      // print function
57      virtual void print() const
58      {
59          cout << "Phylum: " << getPhylum()
60          << "\nGenus: " << getGenus();
61      } // end print function
62  }; // end class VirginiaOyster
63
64  int main()
65  {
66      VirginiaOyster oyster;
67      Oyster *baseClassPtr;
68
69      baseClassPtr = &oyster;
70      baseClassPtr->print();
71
72      cout << endl;
73  } // end main
```

Section D: Correct the Code *(20 Marks)*

For each of the given complete programs or program segments, determine if there is one or more errors in the code. Write down the line number and describe the error or write the corrected form.

1. The following defines class `Increment`. (5)

```
1  #include <iostream>
2  using namespace std;
3
4  // class Increment definition
5  class Increment
6  {
7  public:
8      Increment( int c = 0, int i = 1 );
```

```
 9     void addIncrement() { count += increment; }
10     void print() const;
11 private:
12     int count;
13     const int increment;
14 }; // end class Increment
15
16 // constructor
17 Increment::Increment( int c, int i )
18 {
19     count = c;
20     increment = i;
21 } // end class Increment constructor
22
23 // function print definition
24 void Increment::print() const
25 {
26     cout << "count = " << count
27     << ", increment = " << increment << endl;
28 } // end function print
```

2. Class X inherits from class Y. (5)

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 // class Y definition
 5 class Y
 6 {
 7 public:
 8     Y(); // default constructor
 9     ~Y(); // destructor
10 private:
11     int data;
12 }; // end class Y
13
14 // class X definition
15 class X ; public Y
16 {
17 public:
18     // function print
19     void print() const
20     {
21         cout << data;
22     } // end function print
23 }; // end class X
```

3. The following code should construct a Derived object. (5)

```
1 #include <iostream>
2 using namespace std;
3
```

```
// class Base definition
class Base
{
private:
    // constructor
    Base( int b )
    {
        cout << b;
    } // end class Base constructor
}; // end class Base

//  base Derived definition
class Derived : public Base
{
    // constructor calls base-class constructor
    Derived( int a )
    : Base( a )
    {
        // empty
    } // end class Derived constructor
}; // end class Derived

int main()
{
    Derived d( 5 );
} // end main
```

4. The following code creates an object of type B. Class B inherits from class A. (5)

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

// class A definition
class A
{
public:
    // constructor
    A( int a )
    {
        value = a;
    } // end class A constructor

    // return value
    int getValue() const
    {
        return value;
    } // end function getValue
private:
    int value;
}; // end class A

```

```
23 // class B definition
24 class B
25 {
26 public:
27      // constructor
28      B( int b )
29      : A( b )
30      {
31           // empty
32      } // end class B constructor
33 }; // end class B
34
35 int main()
36 {
37      B object( 50 );
38      cout << object.getValue();
39 } // end main
```

Section E: Programming *(20 Marks)*

Create a class called `Date` that includes three pieces of information as data members - a month (type `int`), a day (type `int`) and a year (type `int`). Your class should have a constructor with three parameters that uses the parameters to initialize the three data members. For the purpose of this programming question, assume that the values provided for the year and day are correct, but ensure that the month value is in the range 1-12; if it is not, set the month to 1. Provide a *set* and a *get* function for each data member. Provide a member function `displayDate` that displays the month, day and year separated by forward slashes (/). Write a test program that demonstrates class `Date`'s capabilities. Your output should appear as follows:

```
Original date:
5/6/1981

New date:
1/1/2005
```

The End

Designed using LaTeX 2$_\varepsilon$