UNIVERSITY OF ESWATINI

FACULTY OF SCIENCE & ENGINEERING

DEPARTMENT OF ELECTRICAL & ELECTRONIC ENGINEERING

MAIN EXAMINATION

MAY 2019

TITLE OF PAPER: PROGRAMMING TECHNIQUES II

COURSE CODE: EEE272

DURATION: 3 HOURS

INSTRUCTIONS:

1. There are five (5) questions in this paper. Answer question 1 and any other three (3) questions.
2. Each question carries equal marks.
3. Use correct notation and show all your steps clearly in any program analysis.
4. All programs should be sufficiently commented and indented for clarity.
5. Start each question in a new page.

This paper should not be opened until permission has been given by the invigilator.

This paper contains seven (7) pages including this page.

Question 1

a. What is a class? Using an example describe how a class relates to an object. [3]

b. How are overloaded functions differentiated from each other when a program is executed? [2]

c. What is a class inheritance hierarchy? [2]

d. What problem does the C++ namespace facility solves? [2]

e. Given the declarations shown below, write a program fragment that allocates a nameless variable to which p1 will point [2]

```
int *p1, *p2;
```

f. What is "this" pointer? With the aid of examples describe two instances where you may use a "this" pointer. [6]

g. When would it be necessary and convenient to overload a constructor of a class? [2]

h. What is the purpose of a scope resolution operator in a class implementation file? [2]

i. What is the purpose of a forward declaration of a class when declaring friendship amongst classes? [2]

j. Using an example illustrate copy initialization of an object. [2]

Question 2

a. The member function getValue in the program segment shown below is meant to return an address to some storage that contains an integer 10. Answer the following questions.

```
class DoSomething{


public:
      int *getValue(){
            int x = 10;
            return x;
      }
      /*rest of the class is fitted here*/
};

int main(){
      DoSomething P;
      int *y = P.getValue();
      cout<< *y<<endl;
}
```

i. Identify any incorrectness with the implementation of the function. [3]

ii. What do you think will be printed when the program is executed? [2]

iii. Rewrite the function to correct the program's implementation. [3]

b. Identify and describe all errors in the class definition shown below. Explain how these errors can be rectified. [5]

```
class Jeans
{
private:
      int size;
      string brand;
public:
      Jeans(int mSize = 10,string s = "Levis"):
mSize(size), s(brand)
      {

      }
      int setSize(int size)
      {
            *this ->size = size;
      }

      void setBrand(string b)
      {
            brand = b;
      }
};
int main()
{
      Jeans jeans;
      jeans.size = 8;

}
```

c. Given the class declaration shown in **Figure Q.2c**;

i. Write the C++ implementation of the overloaded operators >> and <<. [8]

ii. Provide set and get functions for the declarations. [4]

```
class Exam{

      friend istream& operator>>(istream &in; Exam E);
      // To provide definition
      friend ostream& operator<<(ostream &out, Exam E);
      // To provide definition
private:
      string courseName;
      string courseCode;
      int   year;
public:
/* To provide set and get functions */
};
```

Figure Q.2c

Question 3

a. Consider static members of a class for the following questions.

i. What is a static member of a class? [2]

ii. Why would a programmer make a member of a class static? [3]

b. Given the program shown in **Figure Q.3b**.

i. Show what will be printed by the program. For each statement printed explain how the output is obtained. [6]

ii. What do you think is the rationale of including line 13 in the program? [3]

```
1  #include  <iostream>
2
3  using namespace std;
4
5  class Illustrate{
6
7  public:
8      static int n;
9      Illustrate(){++n;}
10     ~Illustrate(){--n;}
11 };
12
13 int Illustrate::n = 2;
14
15 int main()
16 {
17     Illustrate a;
18     Illustrate b[4];
19     Illustratee *c = new Illustrate;
20     const<<a.n<<"\n";
21     delete c;
22     cout<<Illustrate::n<<"\n";
23     return 0;
24 }
```

Figure Q.3b

c. Identify all errors in the following class definition, and then explain how to correct them.

[3]

```
class Example
{
      public:
            Example(int y = 10): data(y)
            {
                  // empty
            }

            int getIncrementedData() const
            {
                  return ++data;
            }

            static int getCount()
            {
                  cout<<"Data is "<< data << endl;
            }

      private:
            int data;
            static int count;
};
```

d. When a basic data type variable is declared constant, the contents of the variable may not be changed after initialization. Briefly discuss the meaning of declaring an object constant to the compiler. [3]

e. Define the concept of friendship in object oriented programming. Some people are against this concept: Discuss the possible negative aspects of friendship. [5]

Question 4

Create a class *HugeInteger* that uses a 40-element array of digits to store integers as large as 40 digits each. Provide the following member functions for the class.

a. Input and Output member functions:

   i. *Input:* reads the digits of a HugeInteger object. [3]

   ii. *Output:* writes out the digits of a HugeInteger object. [1]

b. Arithmetic member functions:

   i. *Add:* to calculate the sum of two HugeInteger objects. [5]

   ii. *Subtract:* to calculate the difference between two HugeInteger objects. [5]

c. Member functions for comparing *HugeInteger* objects:

i. *isEqualTo:* returns TRUE if a *HugeInteger* object is greater than or equal to another *HugeInteger* object. Returns FALSE otherwise. [2]

ii. *isNotEqualTo:* returns TRUE if a *HugeInteger* object is NOT equal to another HugeInteger object. Returns FALSE otherwise. [1]

iii. *isGreaterThan:* returns TRUE if a *HugeInteger* object is greater than another *HugeInteger* object. Returns FALSE otherwise. [2]

iv. *isLessThan:* returns TRUE if a *HugeInteger* object is less than another *HugeInteger* object. Returns FALSE otherwise. [2]

v. *isGreaterThanOrEqualTo:* returns TRUE if a *HugeInteger* object is greater than or equal to another *HugeInteger* object. Returns FALSE otherwise. . [1]

vi. *isLessThanOrEqualTo:* returns TRUE if a *HugeInteger* object is less than or equal to another *HugeInteger* object. Returns FALSE otherwise. [1]

vii. *isZero:* returns TRUE if a *HugeInteger* is equal to 0. Returns FALSE otherwise. [2]

Question 5

a. What is inheritance in object oriented programming? Discuss ways by which it contributes to software reuse and short turnaround times in program development. [6]

b. The program segment shown below will produce an error. Identify the cause of the error, and write the corrected version of the program. [4]

```
string &Human::getName() const{
      return name;
}
```

c. Define a class named ***LinearEquation*** for a 2X2 system of linear equations:

$ax + by = e$ $\qquad x = \frac{ed-bf}{ad-bc} \qquad y = \frac{af-ec}{ad-bc}$

$cx + dy = f$

The class should contain the following:

- Private data members a,b, c, d, e and f.
- A constructor with the arguments for a, b, c, d, e and f.
- Six get functions for a,b,c,d,e and f.
- A member function called is ***isSolvable()*** that returns true if $ad - bc$ is not 0.
- Member functions ***getX()*** and ***getY()*** that return the solution for the equation.

Write the implementation of the class and a test program that prompts the user to enter a,b,c,d,e and f, and displays the result. If $ad - bc$ is 0 report that `"the equation has no solution"`.

Note:

- You are not required to separate the interface of the class from implementation.

[15]

End of paper