MULTIMEDIA UNIVERSITY

STUDENT ID NO

VENUE : ____________________

SEAT NO : ____________________

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2018/2019

TCP 1201 – OBJECT-ORIENTED PROGRAMMING AND DATA STRUCTURES

(All sections / Groups)

04 MAR 2019
2.30 p.m. – 4.30 p.m.
(2 Hours)

Question	Mark
1	
2	
3	
4	
Total	

INSTRUCTIONS TO STUDENT

1. This question paper consists of 13 printed pages (including cover page).

2. Attempt all questions. The distribution of the marks for each question is given.

3. Print all your answers **CLEARLY** in the specific answer box provided for each question. Submit this question paper at the end of the examination.

Read the program below and answer questions 1 and 2.

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

class Person
{
   private:
      string name;
   public:
      Person() : name( "NoName" )
      {}
      string getName() const { return name; }
      void setName( string aName )  { name = aName; }
      void print() { cout << "Name : " << name << endl; }
};

class Vehicle
{
   private:
      Person* owner;
      int maxPassenger;
      string vType;
   public:
      Vehicle() : owner( nullptr ), maxPassenger( -1 ), vType( "NoVType" )
      {}

      Person* getOwner() const { return owner; }
      int getMaxPassenger() const { return maxPassenger; }
      string getVType() const { return vType; }
      void setOwner( Person* anOwner ) {  owner = anOwner; }
      void setMaxPassenger( int aMaxPassenger )
      { maxPassenger = aMaxPassenger; }
      void setVType( string aVType ) { vType = aVType; }

      virtual void print()
      {
         if( owner != nullptr ) owner->print();
         else cout << "NoOwner" << endl;

         cout << "Max Passenger : " << maxPassenger << ".   ";
         cout << "Vehicle Type : " << vType << "." << endl;
      }

     virtual ~Vehicle() {}
};

class Bicycle : public Vehicle
{
   public:
     Bicycle() : Vehicle()
     {
        this->setMaxPassenger( 1 );
        this->setVType( "Unregulated Bicycle" );
     }
};

class Bus : public Vehicle
{
   private:
      string plateNumber;
      string companyName;
   public:
      Bus() : Vehicle()
      {
```

Continued...

```
         this->setPlateNumber( "NoPlateNumber" );
         this->setCompanyName( "NoCompany" );
         this->setMaxPassenger( 30 );
         this->setVType( "Regulated Bus" );
      }

      string getPlateNumber() const { return plateNumber; }
      string getCompanyName() const { return companyName; }
      void setPlateNumber( string aPlateNumber)
      { plateNumber = aPlateNumber; }
      void setCompanyName( string aCompanyName)
      { companyName = aCompanyName; }

      void print()
      {
         Vehicle::print();
         cout << "Plate Number : " << plateNumber << ".   ";
         cout << "Company Name : " << companyName << "." << endl;
      }
};

class Car : public Vehicle
{
   private:
     string plateNumber;

   public:
      Car() : Vehicle()
      {
         this->setPlateNumber( "NoPlateNumber" );
         this->setMaxPassenger( 4 );
         this->setVType( "Regulated Car" );
      }

      string getPlateNumber() const { return plateNumber; }
      void setPlateNumber( string aPlateNumber)
      { plateNumber = aPlateNumber; }
      void print()
      {
         Vehicle::print();
         cout << "Plate Number : " << plateNumber << "." << endl;
      }
};

const int NUMBER_OF_VEHICLES = 5;

int main()
{
   Person p1; p1.setName( "Ali" );
   Person p2; p2.setName( "AhKao" );
   Person p3; p3.setName( "Muthu" );

   Vehicle* v[ NUMBER_OF_VEHICLES ];

   Bicycle* a;
   Bus* b;
   Car* c;

   a = new Bicycle;
   v[ 0 ] = a;

   b = new Bus;
   b->setPlateNumber( "WHR 1010" );
   b->setCompanyName( "ABC Travel" );
   b->setOwner( &p2 );
   v[ 1 ]  = b;
```

Continued...

```
    c = new Car;
    c->setPlateNumber( "WWW 2211" );
    c->setOwner( &p3 );
    v[ 2 ] = c;

    c = new Car;
    c->setPlateNumber( "AAY 1234" );
    c->setOwner( &p1 );
    v[ 3 ] = c;

    a = new Bicycle;
    a->setOwner( &p2 );
    v[ 4 ] = a;

    a = nullptr; b = nullptr; c = nullptr;

    //
    // Code insertion point for question 2(d)
    // Assume the code below does not exist for question 2(d)
    //

    for( int i = 0; i < NUMBER_OF_VEHICLES; ++i )
    {
       cout << i + 1 << endl;
       v[ i ]->print();
    }

    for( int i = 0; i < NUMBER_OF_VEHICLES; ++i )
    {
       delete v[ i ];
       v[ i ] = nullptr;
    }

    return 0;
}
```

QUESTION 1 [10 marks]

(a) What is the output of the program?

[3]

Continued...

(b) What kind of relationship exits between the Vehicle class and the Person class? States the reason for your answer?

[2]

(c) In the print() method in the Vehicle class, explain why checking whether owner is a null pointer before invoking the print() method in the Person class is needed. What will happen if the check is not in place?

[2]

(e) Explain the difference between function overriding and overloading? Give an example of each case from the program above if found.

[3]

Continued...

QUESTION 2 [10 marks]

(a) Does the program have memory leak? What will happen at the end of the program to resources held by v?

[2]

(b) If the main() of the program is changed to

```
int main()
{
   Person p1;
   p1.setName( "Ali" );

   Vehicle vehicleArray[ 2 ];
   vehicleArray[ 0 ].setMaxPassenger( 2 );
   vehicleArray[ 0 ].setVType( "Regulated Bus" );
   vehicleArray[ 0 ].setOwner( &p1 );

   cout << vehicleArray[ 0 ] << endl << vehicleArray[ 1 ] << endl;

   return 0;
}
```

It displays the following :

```
Name : Ali
Max Passenger : 2
Vehicle Type : Regulated Bus

Owner is a null pointer
Max Passenger : -1
Vehicle Type : NoVType
```

Write the code to overload operator<< for the code in main() to work.
Hint: Do not use print() in your code as print() is limited to displaying to screen only.

[3]

```
ostream& operator<<(ostream& out, const _________& _____)
{









   out << "Max Passenger : " << v.getMaxPassenger() << endl;
   out << "Vehicle Type : " << v.getVType() << endl;

   return ______;
}
```

Continued...

(c) If the main() of the program is changed to

```
int main()
{
   Car aCar;
   Bus aBus;
   Bicycle aBicycle;

   cout << boolalpha << myCompare( aCar, aBus ) << endl;
   cout << boolalpha << myCompare( aBicycle, aBus ) << endl;
   cout << boolalpha << myCompare( aBus, aBus ) << endl;
   cout << boolalpha << myCompare( aBus, aCar ) << endl;
   cout << boolalpha << myCompare( aCar, aBicycle ) << endl;

   return 0;
}
```

It displays the following :

```
false
false
false
true
true
```

myCompare() is a non-member function which compares two vehicles. If maxPassenger of the first vehicle is higher than that of the second vehicle, it returns true. Otherwise, it returns false.
Write the code to define a function template for MyCompare

[2]

Continued...

(d) If the code at the insertion point in main() of the program is replaced with the code below, the program displays

```
AhKao owns 2 vehicles.
```

Complete the code below so that it counts the number of vehicles owned by ownerToSearch and display the results.

[3]

```
string ownerToSearch = "AhKao";

int vehicleCounter = 0;

for( int i = 0; i < NUMBER_OF_VEHICLES; ++i )
{










}

cout << ownerToSearch << " owns " << vehicleCounter << " vehicles." << endl;

return 0;
```

Continued...

QUESTION 3 [10 marks]

(a) List two benefits of using linked lists over arrays?

[2]

(b) Give the most appropriate data structure for each of the following use cases.
i) Implementation of a depth-first search algorithm
ii) Implementation of a printer buffer supporting multiple users printing
iii) Implementation of a program to store and retrieve personal friend contact info

[3]

i)

ii)

iii)

(c) Determine the output of the following program.

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

int main()
{
    map<string, vector<int>> wordIndexes =
    {
        {"word2", vector<int> {21, 22, 23}},
        {"word1", vector<int> {11, 12, 13}},
        {"word1", vector<int> {21, 22, 23}}
    };

    for (auto word : wordIndexes)
       cout << word.first << endl;

    cout << wordIndexes.count("word1") << endl;
}
```

[2]

Continued...

(d) Given the following MString class declaration, complete the method PrintReverse() so that it causes the printing of the data sting characters in reverse order. PrintReverse() is recursive. The output of the program is

```
ICF UMM
```

[3]

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

class MString {
  string data;

public:

  MString(){
    data="";
  }

  MString(string s){
    data = s;
  }
  void PrintReverse(string t)
  {



  }
  void PrintReverse(){
    PrintReverse(data);
  }
};

int main() {

  MString s("MMU FCI");
  s.PrintReverse();
  return 0;
}
```

Continued...

QUESTION 4 [10 marks]

(a) Given the following two declarations for ListNode and LinkedList,

```
template <class T>
struct ListNode
{
    T value;
    ListNode* next;
    ListNode(T value1, ListNode* next1 = nullptr)
    {
        value = value1;
        next = next1;
    }
};

template <class T>
class LinkedList
{
protected:
    ListNode<T>* head;
public:
    LinkedList();
    ~LinkedList();
    void add(T value);
    void remove(T value);
    void displayList() const;
};
```

Write the code to implement the constructor LinkedList() outside the class declaration.

[2]

(b) Given the following class declaration of the circular queue data structure for the class IntQueue. Write the code in the provided box to complete the member function enqueue.

[3]

```
class IntQueue
{
private:
   unique_ptr<int []> queueArray;
   int queueSize;
   int front;
   int rear;
   int numItems;
public:
   IntQueue(int);

   void enqueue(int);
   void dequeue(int &);
   bool isEmpty() const;
   bool isFull() const;
   void clear();
};
```

Continued...

```
void IntQueue::enqueue(int num)
{
   if (isFull())
   {
      cout << "The queue is full.\n"; exit(1);
   }
   else
   {





   }
}
```

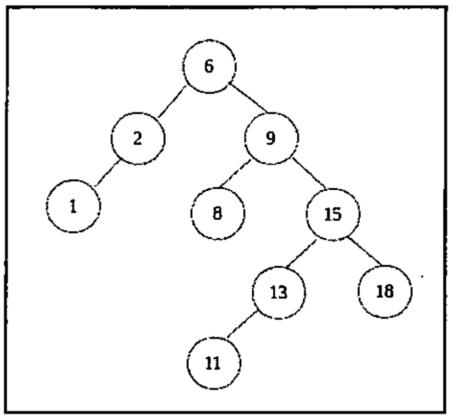
(c) Given the following class declaration for the class DynIntStack. Write your answer in the provided box to complete the code inside the destructor body.

[3]

```
class DynIntStack
{
    struct StackNode
    {
        int value;
        StackNode* next;

        StackNode(int value1, StackNode* next1 = NULL)
        {
            value = value1;
            next = next1;
        }
};
    StackNode* top;
public:
    DynIntStack();
    ~DynIntStack();

    void push(int);
    void pop(int &num);
    bool isEmpty() const;
};

DynIntStack::~DynIntStack()
{
   StackNode* garbage = top;

   while (garbage != nullptr)
   {





   }
}
```

Continued...

(d) Given the following Binary Search Tree, draw the resultant BST after deleting the node with the value 9.



[2]

End of Page