Do not award half marks.
In all cases give credit for appropriate alternative answers.

Question 1 (Compulsory)

(a) 'In structured programming, a big problem in the real world is broken into many procedures that each solve a specific portion of the big procedural problem'. How should one use Object-oriented programming to solve a real world problem? [2]

First, one should identify the *objects* that represent the concepts of the real problem [1 mark]. Second, one must identify the *interactions* between objects that enable one to solve or model the real world problem [1 mark].

Marks should be awarded to other equally valid answers that mention the key ideas of objects and messages.

(b) Briefly describe the following terms in the context of the C++ programming language. [4]

Encapsulation

is the process of combining the aspects (i.e. instance variables and methods) of a data type into a larger element by creating a class.

Methods

are associated with a class. They are functions within a class that enable the outside world to gain access to the private member variables and perform some computation.

Destructor

a special function to free the memory allocated to an object when is was created. The object is then no longer in use.

Constant member function

is a function that can be safely invoked through a constant object; otherwise a compiler error occurs.

(c) ‘Data members of a class may be objects of other classes, such modeling is known as “has-a” relationship.’ How is the order of both constructor and destructor execution implemented? [3]

- **the constructor for each inner object is invoked first [1 mark] then the constructor for the outer object [1 mark]**
- **the destructors are called in the exact reverse order of the constructor calls [1 mark]**

(d) When a student compiles the following C++ program, he/she observes an error reported by the compiler. The error is “**Cannot find a default constructor to initialize the array element of type num**”.

```
class  num {
        int *data;
        public:
                num (int d) { data = new int(d); }
}; // end num class

void main( ) { num ArrayOfNum[3]; }
```

(i) What should the student include in the constructor for num class, so that the program can successfully compile? You should not remove any code from the C++ program. [1]

public: num(int d=0 [1 mark]) { data = d }

(ii) Re-write the array declaration, ArrayOf Num, so that it’s elements are initialized with default values. [1]

ArrayOfNum[3] = { num(50), num(60), num(23) };

[1 mark, only given for correct use of nested constructor num]

(iii) The member variable data is a pointer. Thus, the default copy constructor provides a *shallow* copy and should be overridden. Show how to implement a copy constructor that performs a *deep* copy for num class. [2]

num :: num (const num &n) [1 mark]
{ data = new int(*n.data); [1 mark] }
[Note: should not penalize the student for using dereferencing]

(e) Given the declaration of class book as follows.

```
class book {
        char ISBN[14];
        float *cost;
public:
        void PrintBook( ) {
                        cout << “\n\t\ISBN: “ << ISBN << endl;
                        cout <<”\tCost $ “ <<*cost << endl; }

        int FoundBook(char *isbn) {
                if (strcmp(ISBN, isbn) = =0)
                        return 1;
                else
                        return 0; }
        }
}
```

(i) Implement a public *constructor* that takes in two default arguments: **cost** of type float and **ISBN** which is a character pointer, the member variables should be properly initialized with these arguments. The cost stores a value 0 and ISBN contains a string “default”. [4]

```
book::book(float cost =0, char *ISBN=”default”) [1 mark]
{ this→cost = new (cost); [1 mark]
  strcpy(this→ISBN, ISBN); [1 mark] }
```

[1 mark for using this keyword]

(ii) Implement a *destructor* for book class. [2]

```
book:: ~book( ) { delete cost; [1 mark] }
[1 mark for correct destructor syntax]
```

(iii) Implement an *accessor* method **GetCost** that returns the member variable cost. [2]

```
float *book :: GetCost() { return cost; [1 mark] }
[1 mark for correct method declaration]
```

(iv) Fill in the missing C++ code so that the given method performs as a recursive function. The method **SearchBook** takes in an array of books **ListOfBooks**, an integer **size** that represents the size of the array, a character pointer **findISBN**. The method should return a book that has the given ISBN code. Otherwise it should return a book that is initialized with default values. [4]

```
book SearchBook( book ListOfBooks[], int size, char *findISBN)
        {       if ( _____________ )
                        return  ___________;
                else if  (_________________________________)
                        return ListOfBook[size];
                else
                        return ______________________________;
        }
```

book SearchBook(book ListOfBooks[], int size, char *findISBN)

```
{       if( size = = -1)            [1 mark]
                return  book    [1 mark] ;
        else if  (ListOfBook[size].FoundBook(findISBN)) [1 mark]
                return ListOfBook[size];
        else
                return SearchBook(ListOfBook, size-1, findISBN);
                                                        [1 mark]
}
```

(f) Consider the *fragment* of C++ program given below. It creates a class called **car** and a class called **truck**, each containing as a private variable the speed of the vehicle it represents.

```
class truck;

class car {
        int passengers;
        int speed;  };

class truck {
        int weight;
        int speed;  };
```

You should implement a non-member function **sp_greater()** that has access to the private instance variable **speed** in both classes. It should return a value of 1 for when the car object is faster than the truck object, 0 for when their speeds are the same or –1 to represent the case when the truck object is faster. You may have to show how the friend function is written in both classes and the definition/implementation for **sp_greater()**. [5]

```
class truck;
class car {
        int passengers;
        int speed;
        friend int sp_greater( truck t, car c);
};

        class truck {
                int weight;
                int speed;
                friend int sp_greater(truck t, car c);
        };

int sp_greater(truck t, car c)
 {
        if (c.speed = = t.speed)
                return 0;
        elseif ( c.speed > t.speed)
                return 1;
        else    return –1;  }
```

1 mark given for correctly outlining the friend method in both classes.
1 mark for the correct parameters received by the friend function in both classes.
1 mark for using a conditional statement within sp_greater.
1 mark for correct the conditions.
1 mark for correctly returning the described values.

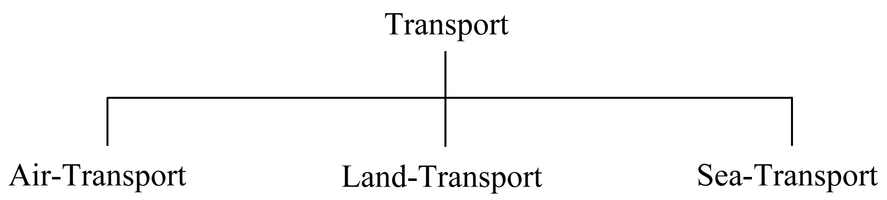
Do not award half marks.
In all cases give credit for appropriate alternative answers.

Question 2

(a) For the following two examples:

(i) Name each class hierarchy.

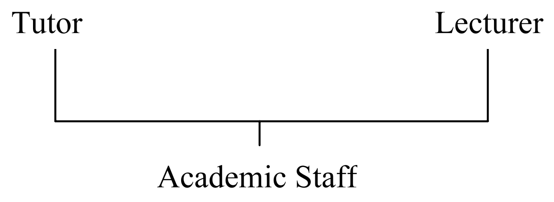
(ii) Implement each class hierarchy using the C++ programming language. Each derived class publicly inherits the base class. [4]



Single inheritance [1 mark]

```
class transport {  };
class Air_Transport: public Transport { };
class Land_Transport: public Transport { };
class Sea_Transport: public Transport { };
```

[1 mark for correctly implementing the class hierarchy]



Multiple Inheritances [1 mark]

```
class Lecturer {  }
class Tutor { }
class Academic Staff: public Lecturer, public Tutor {      }; [1 mark]
```

(b) What happens when a protected member data is inherited as private? [2]

The protected member data becomes private in the derived class [1 mark]
Which means that it is inaccessible to any class [1 mark].

(c) Given the declaration of plant class below.

```
class plant {
        char *PlantName;
protected:
                int age;
public:
                plant (char *p = "default" )
                        {       PlantName = new char[Strlen(p) +1];
                                strcpy(PlantName,p);
                                age =0; }
                char *GetName( ) { return PlantName; }

};
```

(i) Implement a virtual function **SetAge** that takes in an integer **newage** for replacing the member variable age. [2]

void SetAge(int newage) { age=newage; };
[2 marks, -1 mark for each mistake up to a max of 2 marks]

(ii) How should a student define a pure virtual function that overloads the earlier virtual function **SetAge**? [1]

virtual void SetAge( ) = 0; [1 mark]

(iii) Create a derived class called **WeakStem** that privately inherits the base class called plant, and contains a private instance variable **SupportLen** as a float. [2]

```
class WeakStem : private plant [1 mark] {
                private: float SupportLen; [1 mark] }
```

(iv) Implement a constructor for **WeakStem** that takes in appropriate arguments to both pass the parameters to the base constructor and to ensure a default parameter. This default parameter should initialize the **SupportLen** to store the value 0. [3]

```
WeakStem :: WeakStem (char *p, float len =0) [1 mark] : plant(p) [1 mark]
                        { SupportLen = len; [1 mark] }
```

(v) Why can one not create an instance of the WeakStem class? [1]

As a pure virtual function in the base class is not defined in the derived class
The derived class is an abstract class

1 mark for either correct answer given.

Do not award half marks.
In all cases give credit for appropriate alternative answers.

Question 3

(a) 'Data members of a class may be objects of other classes, such modeling is known as "has-a" relationship.'

(i) Given two classes below that depict the "has-a" relationship, name the technique used in the A's classes constructor when the B object's constructor requires parameters

(ii) State one situation where such technique should be used. [2]

```
class A {private:
                int x;
          public:
                A(int s); }

class B {
          private A objectA; }
```

Solution: **Initialization Syntax [1 mark]**
Reference object [1 mark]

(b) Given the declaration of person class below.

```
class person {
        char name[21];
        float testmark;
        public:
                person (char *n="default", float m=0.0) {
                        strcpy(name, n);
                        testmark = m; }
                float GetTestMark() { return testmark; }
                void Print() {
                        cout << "\n\tName " << name << endl;
                        cout << "\tTest Mark " << testmark << endl; }
```

(i) Give an implementation of a method **TestMark ()** for person class, that returns a value1 for testmark is 50 and above; or 0 otherwise. [2]

```
int person :: TestMark()
{           if( testmark >=50) [1 mark]
                    return 1;
            else    return 0;     }
```

[1 mark for correct return values]

(ii) Create a class called **module** that contains the following members

- A pointer, called **stud**, that points to a type person
- Two instance variables of type integer: **currsize** for the number of students in the class; and **studsize** representing the maximum allowed number of students. [2]

The members are inaccessible to any class

```
class module {
        private:
                person *stud; [1 mark]
                int currsize, studsize; [1 mark]
        };
```

(iii) Implement a constructor for the **module** class that takes in an integer **s** denoting the maximum size of the class. It should be used for allocating an array of **s** objects to **stud** using the new operator. It should also assign **s** onto **studsize** and set the **currsize** equal to 0. [3]

```
module::module( int s) {   stud = new person[s]; [1 mark]
                            studsize = s;  [1 mark]
                            currsize = 0;  [1 mark]}
```

(vi) Implement a member function **NoPasses** for the module class that takes in no arguments. It should return an integer representing the number of students who have scored 50 and above in their tests. You should use the **TestMark()** method in your solution. [4]

```
int module::NoPasses() {
        int i, count=0;
        for(i=0; i<currsize;[1 mark] i++)
            count +=stud[i].TestMark();
                        [1 mark for incrementing count]
                        [1 mark for using TestMark()]
        return count; [1 mark] }
```

(v) Write a main function that creates an array of 5 persons such that is pointed by member stud in module class The object is named as **StudCS255,**which invokes member function **NoPass**. [2]

```
void main() { module StudCS255(5); [1 mark]
                cout << " No of Passes " << StudCS255.NoPass() [1 mark] }
```

Do not award half marks.
In all cases give credit for appropriate alternative answers.

Question 4

(a) 'Polymorphism means many forms.'

Briefly explain how the division operator is a good example of ad hoc polymorphism. [2]

If the arguments of the division operator are integral, then integer division is performed. However, if both arguments are in floating point, then floating point division would be performed. So the division operator works on several different types [1 mark] and it converts their values to the correct expected type [1 mark].

(b) Implement a *template* function, called **WhatSmall( )** that takes in two arguments of any type. It should return either of the two arguments if they both denote the same object; otherwise it should return the smallest argument. [4]

```
template <class U> [1 mark]
U What Small( U data1, U data2) [1 mark]
        {       if ( data1 < data2)
                return data1;
                return data2;  }
```

[1 mark for correct condition]
[1 mark for correct return values]

(c) What do parameterized templates provide the programmer with? [1]

Parameterized templates provide you with the ability to create a general class and pass types as parameters to that class in order to build specific instances [1 mark].

(d) Given the following fragment of a C++ program:

```
#define size 10;

class fish {
        private:
        // member variables
        public:

                void PrintFish( );
                friend fish operator >( fish f1, fish f2) ; // method returns a big fish
};

template <class ArrayType>
class Array {
                private:
                        ArrayType container[size]; // holds an array
                int tos; // index of array;
                public:
                        Array( ) { tos =0; }
                };
```

(i) Implement a method that overloads the subscript operator, **operator[]**. This new method takes in an integer index and returns *a reference object* of type **ArrayType** stored at the given index position in the **container**. [3]

Array<ArrayType> &Array<ArrayType> :: operator[](int offset) [2 marks]
{ return container [offset]; } [1 mark]

(ii) Create an instance of Array class called fishtank that stores fishes. [1]

Array<fish> fishtank; [1 mark]

(iii) Complete the implementation of a function **PrintBigElement** that should successfully search for the largest element in the container and display it.

```
        template < class ArrayType, class U>
        void  PrintBigElement(  ArrayType holder)
                {       int i;
                        U temp = holder[0];
```

You should complete the implementation commencing from here.} [4]

```
template < class ArrayType, class U>
void  PrintBigElement(  ArrayType holder)
        {       int i;
                U temp = holder[0];
                for(i=1; i<size; i++) [1 mark] {
                        if (temp<holder[i])
                                temp = holder[i]; } [2 marks]
                temp.display( );  [1 mark]
                }
```

Do not award half marks.
In all cases give credit for appropriate alternative answers.

Question 5

(a) Given the following C++ program:

```
1.   class String {
2.   char *str;
3.   int len;
4.   public:
5.   String(char *s) {   len = strlen(s);
6.                       str = new char[len+1];
7.                       strcpy( str,s); }
8.   void SetString(char *newstr) { strcpy(str, newstr); }
9.   void Display( ) {  cout << str << endl;
10.                     cout << len << endl; }   };
11.  void main( ) {
12.                 String word("Successful");
13.                 String copy_word(word);
14.                 word.SetString("Failure");
15.                 word.Display();
16.                 copy_word.Display(); }
```

(i) What are line 12 and line 13 doing? [2]

Line 12 – creates an instance of String class [1 mark]
Line 13 – makes a copy of the object word onto copy_word [1 mark]

(ii) When the code is run, why does line 13 *not* generate an error message? [1]

The C++ compiler provides a default copy constructor [1 mark].

(iii) What is printed out on line 15 and line 16? [4]

Solution:
Failure [1 mark]
10 [1 mark]
Failure [1 mark]
10 [1 mark]

(b) Given the following C++ program.

```
1.   class ObjCreated {
2.   static int NoObj;
3.   int whichObj;
     public:
4.   ObjCreated( ) { whichObj =NoObj;
5.                     NoObj--; }
6.
7.   static void WhatHave( ) { cout << "No Objects = " << NoObj << endl;
8.                                cout << "Object Status " << whichObj << endl; }
9.     static int GetNoObj( ) { return NoObj; }
10.    };
11.  void main( ){ ObjCreated  ObjArray[10];
12.  cout << ObjArray[0].GetNoObj( ) << "\n "<<ObjCreated::GetNoObj( ) <<
13.  endl; }
```

(i) What is the difference between the two variables **NoObj** and **whichObj** in **ObjCreated** class? [2]

variable whichObj – each instance has a separate copy of the variable [1 mark]
variable NoObj – all instances share one copy of num [1 mark]

(ii) How do static member functions differ from constant member functions? [1]

If a member function only needs to access the static data member of a class, it can be defined as a static member function [1 mark].
[Award marks for more implementation oriented solutions]

(iii) When compiling the above fragment of code, the error "**Member NoObj cannot be used without an object**" is displayed. Explain why this error occurs on line 5. [2]

Static member functions do not have a "this" pointer [1 mark] therefore non-static data members of an object cannot be accessed [1 mark].

(iv) Write a C++ instruction that initializes the class static to 300. [1]

int ObjCreated:: NoObj = 300; [1 mark]

(v) What is/are printed on screen? [2]

290 [1 mark]
290 [1 mark]

- END OF PAPER -