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

Question 1 (Compulsory) (30 marks)

(a) What is the difference between structured programming and object-oriented programming? [2]

Structured programming consists of a series of procedures.[1]
Object-oriented programming consists objects and their interactions.[1]

(b) Identify one drawback of structured programming. Briefly comment on it. [1]

Changing one variable may have adverse effects on the whole program.

(c) What is the function of the keyword **class** in C++? [2]

The keyword class in C++ is to bind data and operations as a whole [1] to represent the implementation of abstract data type[1].

(d) The followings are some of the requirements of an object-oriented language:

- Polymorphism
- Inheritance
- Encapsulation

Briefly explain each term in the context of C++ programming. [3]

Polymorphism is the feature in object-oriented programming that allows the same operation to be carried out differently, depending on the object. [1]

Inheritance – a new class (derived class) contains all the same data members and member functions as the original class (base class)[1]

Encapsulation is a process to bind the data and function members as a whole to represent an object.[1]

(e) When Ali wants to telephone to a friend of his, he takes a long time to locate the friend's telephone number in his telephone directory. Ali plans to design and build an application to manage his telephone directory so he enrolls into Informatics short course which is Object-oriented programming in C++ language.

What are the two main objects of this application? What is their interaction? [3]

Objects - friends and telephone_book [2]
Interaction – has-a relationship. [1]

(f) Given the definition of Test class below.

```
class Test {
        float num;
        char data;
        int value;
        };

void main( ) { Test     obj1,
               Test     obj2(4.5);
               Test     obj3(4.5, 'A');
               Test     obj4(4.5, 'A', 4);
               }
```

(i) Give a constructor that has a suitable declaration for Test class to allow the objects to be created. [2]

Test(float n = 0, char d ='A'; int v=0);
1m for correct argument
1m for correct constructor.

(ii) Briefly explain why the C++ compiler reports an error on obj3.value in the main(). Show how the error can be corrected in C++ code. [4]

- **value is private [1] that is inaccessible to any other class [1]**
- **int Test :: getValue() [1] { return value;} [1]**

(iii) Assume the existence of an interface, **void setValue(int x)** that increases **value** by an integer *x*. Implement a function **increaseBy**–according to the signature given below–that iterates through *n* of the objects in *tarray* and increases their member **value** by *x* if it is less than 40. [4]

void increaseBy(Test tarray[], int n, int x)

```
void increaseBy(Test tarray[], int n, int x)
{
        for(int i=0; i<n ; i++) [1]
                if (tarray[i].getValue()< 40) [2]
                        tarray.[i].setValue(x); [1] }
```

(g) (i) What is composition in the object-oriented programming? [1]

Composition is the creation of one class using another class for its component data.

(ii) Assume the existence of a class **Point**. Create a class called **Line** that contains a class called **coordinates** of type Point as data member. [2]

```
class Line {
        Point [1] Coordinates; [1] };
```

(h) Consider two classes called StudentRequest and ClassRec developed for a college registration system. The StudentRequest class contains a method grantEnrolment that takes in a reference object oneclass of type ClassRec, needs access to a ClassRec private data member. Show the declaration of a class granting friendship to a single function that is a member of another class. You may not give the implementation of grantEnrolment() method. [4]

```
class StudentRequest {
                public:
                        void grantEnrolment(ClassRec &oneclass);

[1m correct declaration,  1m for placing the method at correct position] };

class ClassRec {
                public:
                        void friend [1] StudentRequest ::
                        grantEnrolment(ClassRec &onecclass); [1] }
```

(i) Can a friend function make a request to invoke one of the other methods? Justify your answer. [2]

No[1]
A friend function does not have "this" pointer. [1]

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

Question 2

(a) Briefly explain how the default copy constructor makes a copy of an existing object that contains pointers and structure sharing. [2]

Only the pointers in the objects are copied and not the data being pointed to. [1]
Only one copy exists [1]

(b) Give one suitable example in which copy constructor will take place. [1]

An object is passed as a parameter to a function by value [1]
Or
An object is returned from a function by value [1]

(c) Given the fragment of the Unit class below

```
class Unit {
        char *title;
        int num;
}
```

(i) Implement a constructor that takes in two arguments *t* of type characters pointer and *n* of type integer to initialize the data members. The data member *title* has exact memory size allocated by using the **new** operator. [3]

Unit::unit(char *t, int n)

```
Unit::unit(char *t, int n)
{ title = new char[strlen(t) + 1] [1]
        strcpy(title,t); [1]
        num = n; [1] }
```

(ii) The C++ instruction **Unit obj1(obj2)** in the main() is successfully executed.

Why no error is reported?
What does this C++ instruction perform? [3]

The C++ compiler provides a default copy constructor [1]
obj2 is copied onto obj1 [1]so the data member title in both objects point to the same memory[1]

(d) What data members within a class require initialization? [3]

constant data type[1]
reference data type[1]
another class object that has a constructor [1]

(e) Given the following class definition

```
class Tank {
        const float water_level;
        int &numfishes;
};
```

Implement a suitable constructor that takes in appropriate parameters to initialize the data members in Tank class. [3]

Tank:: tank(float level, int &num)[1correct parameters] :
water_level(level)[1], numfishes(num) [1] {}

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

Question 3

(a) Given the fragment of String class below

```
class String {

        char *buf;
        int len;
}
```

Implement a member function for String class named charAt that takes in an integer *x* to return a character stored at position *x-1*; otherwise null if the parameter *x* is bigger than the length of a string or *x* has a value 0. [3]

```
char String :: charAt(int x) [1]
{       if ( (x > 0) && (x < len+1)) [1] return buf[x-1] ;
                                         return buf[len+1] ;
[1m for correct returns] }
```

(b) What is the difference between single and multiple inheritance? [2]

Single inheritance - the derived class has only one immediate base class. [1]
Multiple inheritances - the derived class has more than one base class. [1]

(c) Rewrite the declaration of Media class given below, including a pure virtual function named getTitle( ) and a virtual function named print( ).

```
class  Media {
        String title;
                }
```
[4]

```
class Media {
        String title;
                public: virtual String getTitle( ) = 0;
                         [deduct 1m for each error; up to a max. 2m]
                        virtual void print( ) { }
                        [deduct 1m for each error; up to a max 2m]
                        }
```

(d) Define a subclass named Newsletter that privately inherits the Media class and has a private member of type float to store its price. [2]

class Newsletter : private Media [1]
{ private: (optional)
float store; [1] }

(e) Why would the C++ compiler fail to compile the code given below?

void main( ) { Newsletter News; } [2]

The Newslatter class inherits the pure virtual function GetTitle( ) [1]
Pure virtual function prevents instantiation [1]

(f) Implement an off line overridden member function in the Newsletter subclass that returns the title. [2]

String title :: getTitle( ) [1] { return title; }[1]

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

Question 4

(a) Briefly explain the term static data member? [2]

static data member: - only one copy per class,[1]
- shared among the objects of same type.[1]

(b) Consider the following declaration of class Test

```
class Test {
        float score;
public:
        test( ) { score =0;
                  count++; }
```

(i) Rewrite the class Test declaration including a static data member count. [1]

```
class test {
                float score;
                static int count; [1]
public:
                test( ) { score =0;
                          count++; }
```

(ii) Show how the static count is initialized to a value 0. [2]

int test [1] :: count = 0; [1]

(iii) Implement a destructor for test class. [2]

test :: ~test( ) [1] { count- -; [1] }

(iv) What is the function of a static member function? [1]

static member function only needs to access the static data member of a class[1]

(v) The compiler reports an error on the static data member definition below. Briefly explain why.

```
static int methodA( ) {  score = 40;
                         return count; }
```
[2]

static member function does not have a this pointer [1]
not allow to access non static data member[1]

(vi) Assume the static member function methodA( ) is error free and returns the static data member count. What is printed on the screen if the following main() is executed?

```
void main( ) { test  test1, test2, test3, test4;
               cout << " count = " << test1.methodA() << endl;
                    }
```
[1]

count = 4

(vii) What are the two ways of invoking a static member function? [2]

through an object **[1]**
explicit full specification, e.g. ClassName :: FunctionName **[1]**

(c) Why would the following program fail to compile? [2]

```
class  Anotherclass {
            int num;
       public:
            void display() { cout << " num " << num << endl; }
                 };
void main() {  const Anotherclass  c1;
               constant c2;
               c1.display();
               c2.display();     }
```

c1 is a constant object [1]
the member function display() should be a constant member function[1]

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

Question 5

(a) Template class definitions and template function definitions promote reuse code. How do they support code reuse? [1]

class and function can be created without explicitly defined the types hence it can be used for many different types.[1]

(b) Create a class called Pair that stores two member variables called *first* and *second* of type **T**. [2]

```
template <class T> [1]
class Pair {
private:
T first, second;  [1]
};
```

(c) Implement an off line copy constructor that performs deep copy. [3]

```
template<class T>
Pair<T> :: Pair [1]( const  Pair obj) [1]
{ first = obj.first;
  second = obj.first;
        [1m for correct initialising the created object] }
```

(d) Create an instance of Pair class called IntPair in which the member variables are of type integer. [1]

Pair<int> IntPair; [1]

(e) Give the declaration of a template function MyTemplate that takes in two arguments, a reference *num* of type T and *data* of type U and returns an element of type P. [3]

```
template<class T, class U, class P> [1]
P Mytemplate< T &num, U data>  [1]
```

[1m for correct reference argument num]

(f) What is the meaning of the term "overloading" in C++? [1]

C++ allows you to use the same name for different functions and they have different parameter type lists.[1]

(g) Implement an overloaded addition operator + for the IntPair class that takes in an object obj of type IntPair and returns an object of type IntPair that is the sum of the first and second members of the two objects. [4]

```
Template<class T>
IntPair<T> operator+(IntPair<T> obj) [1]
{ IntPair<T> tempobj; [1]
        tempobj.first = first + obj.first;
        tempobj.second = second + obj.second;
        [1m for adding the two objects.]
        return tempobj;  [1] }
```

- END OF PAPER -