CS6301 Programming & Datastructures II

Question Bank

UNIT – I

PART – A

1. What is an identifier?

Identifiers are names for various programming elements in c++ program. such as variables, arrays,function, structures, union, labels ect., An identifier can be Composed only of uppercase, lower case letter, underscore and digits, but should start only with an alphabet or an underscore.

2. What is a keyword?

Keywords are word whose meanings have been already defined in the c compiler. They are also called as reserved words.

(ex) main(), if, else, else, if, scanf, printf, switch, for, goto, while ect.,

3. Define constant in c++.

Constants in c++ refers to fixed values that do not change during execution of a program.

4. Define a variable.

A quantity ,Which may vary during execution of a program is called as a variable.

5. What are unary operators?

The operators that act upon a single operand are called as unary operators. The unary operators used in c++ are - , ++, -- and sizeof operators.

6. What are binary operators?

The operators that act upon two operands are called binary operators. The binary operators used in c++ are +, -, *, / , %, =. etc.,

7. What are ternary operators?

The operators that act upon three operands are called as ternary operators. The ternary operator available in c++ is (?:).This operator is also referred as conditional operator.

8. What is meant by an expression?

An expression is a combination of constant, variable, operators and function calls written in any form as per the syntax of the c++ language.

9. State the difference between c and c++.

C	C++
(i). Procedural programming language	Object-oriented programming anguage
(ii) Global variable can be declared	It is an error to declare a variable as global
(iii) Function prototypes are optional	All functions must be prototyped.
(iv) Local variables can be declared only the start of a c program in a c++ program.	Local variables can be declared any where
(v) We can call a main() function, within a program.	This is not allowed

10. List the various oops concepts

Four main OOP concepts

Abstraction

creation of well-defined interface for an object, separate from its implementation

e.g., key functionalities (init, add, delete, count, print) which can be called independently of knowing how an object is implemented

Encapsulation

keeping implementation details “private” i.e., inside the implementation hierarchy an object is defined

in terms of other objects Composition => larger objects out of smaller ones

Inheritance => properties of smaller objects are “inherited” by larger

objects

Polymorphism

use code “transparently” for all types of same class of object

i.e., “morph” one object into another object within same hierarchy

11. Define class and object

Class: It is defined as blueprint or it is a collection of objects

Objects: is an instance of a class

Almost like struct, the default privacy specification is private whereas with struct, the default privacy specification is public

Example:

```
class point
{
double x, y; // implicitly private
public:
void print();
void set( double u, double v );
};
```

12. Define inheritance

Inheritance

• Objects are often defined in terms of hierarchical classes with a base class and one or more levels of

classes that inherit from the classes that are above it in the hierarchy.

• For instance, graphics objects might be defined as follows:

Syntax for Inheritance

```
class derivedClass : public baseClass {
private :
// Declarations of additional members, if needed.
public:
// Declarations of additional members, if needed.
protected:
// Declarations of additional members, if needed.
}
```

13. Define encapsulation

Encapsulation is one of thr most important features of a class.It is the process os combining member functions and the data it manipulates by logically binding the data ane keeping them safe from outside interference.

14. Define abstraction.

Creation of well-defined interface for an object, separate from its implementation

e.g., key functionalities (init, add, delete, count, print) which can be called independently of knowing how an object is implemented

15. Define polymorphism

Polymorphism means “having many forms”. It allows different objects to respond to the same message in different ways, the response specific to the type of the object.

E.g. the message displayDetails() of the Person class should give different results when send to a Student

object (e.g. the enrolment number).

16. List out the benefits of oops.

• Can create new programs faster because we can reuse code

• Easier to create new data types

• Easier memory management

• Programs should be less bug-prone, as it uses a stricter syntax and type checking.

• `Data hiding', the usage of data by one program part while other program parts cannot access the data Will whiten your teeth

19. List out the application of oops.

• Client server computing

• Simulation such as flight simulations.

• Object-oriented database applications.

• Artificial intelligence and expert system
• Computer aided design and manufacturing systems.
• Real time systems, such as process control, temperature control.

20. Define data hiding.

The purpose of the exception handling mechanism is to provide a means to detect and report an "exceptional circumstance" so that appropriate action can be taken.

21. What is the use of scope resolution operator?

In C, the global version of the variable cannot be accessed from within the inner block. C++ resolves this problem by introducing a new operator :: called the scope resolution operator. It is used to uncover a hidden variable.

Syntax:

:: variable name

22. When will you make a function inline?

When the function definition is small, we can make that function an inline function and we can mainly go for inline function to eliminate the cost of calls to small functions.

22. What is overloading?

Overloading refers to the use of the same thing for different purposes.

There are 2 types of overloading:

• Function overloading
• Operator overloading

23. What is the difference between normal function and a recursive function?

A recursive function is a function, which call it whereas a normal function does not.

Recursive function can't be directly invoked by main function

24. What are objects? How are they created?

Objects are basic run-time entities in an object-oriented programming system. The class

variables are known as objects. Objects are created by using the syntax:

classname obj1,obj2,…,objn;

(or) during definition of the class:

class classname

{

------- }obj1,obj2,…,objn;

25. List some of the special properties of constructor function.

• They should be declared in the public section.

• They are invoked automatically when the objects are created.

• They do not have return types, not even void and cannot return values.

• Constructors cannot be virtual.

Like other C++ functions, they can have default arguments

26. Describe the importance of destructor.

A destructor destroys the objects that have been created by a constructor upon exit from the program or block to release memory space for future use. It is a member function whose name is the same as the class name but is preceded by a tilde.

Syntax:

~classname(){ }

27. Define modular programming.

It is a process of splitting a large program in to smaller modules to perform the operations fast and, which helps in easy error checking. Modules should be designed, implemented and documented with regard to their possible future use in other projects.

28. What do you mean by friend functions?

C++ allows some common functions to be made friendly with any number of classes, thereby allowing the function to have access to the private data of thse classes. Such a function need not be a member of any of these classes. Such common functions are called friend functions.

29. Mention the types of polymorphism.

The types of polymorphism are

? compile time polymorphism

? runtime polymorphism

30. What are member functions?

Functions that are declared within the class definition are referred as member function.

31. Define dynamic binding.

Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run-time.

32. State the difference between a constructor and destructor.

Constructor Destructor

A constructor is used to initialize the object A destructor is used for releasing dynamically allocated memory No symbol precedes the class name A tilde symbol precedes the class name Constructors can be overloaded Destructors cannot be overloaded

33. What is the use of new operator?

The new operator is used to allocate contiguous unnamed memory during execution time ane returns a pointer to the start of it.

34. What is the use of static data member?

The static data member informs the compiler that only one copy of the data member exit and all objects of the class should share that variable without duplicating it for each instance of the class.

35. What is the different between a pre-increment and post-increment operator.

A pre-increment operation such as ++a, increments the value of a by 1,before a is used for

computation, while a post-increment operation such as a++, uses the current value or present value of a in the calculation and then increments the value of a by 1.

36. Distinguish between break and continue statement.

Break	continue
a) Used to terminate the loops or to Exit loop from a switch.	Used to transfer the control to the Start of loop
b) The break statement when executed causes Immediate termination of loop executed caused immediate containing it.	The continue statement when termination of the current iteration of the loop.

37. Distinguish between while and do-while loop.

While loop	do-while loop
a) The while loop tests the condition before each iteration	The do-while loop tests the condition after the first iteration
b) If the condition fails initially the loop Is skipped entirely even in the first Iteration.	Even if the condition fails initially the loop is executed once

PART B (16 MARK)

1. State the merits and demerits of object oriented methodology.

Merits:

- Can create new programs faster because we can reuse code
- Easier to create new data types
- Easier memory management
- Programs should be less bug-prone, as it uses a stricter syntax and type checking.
- `Data hiding', the usage of data by one program part while other program parts cannot access the data Will whiten your teeth

Demerits:

- disadvantages of C++ over Java:
- Java protects you from making mistakes that C/C++ don’t, as you’ve
- C++ has many concepts and possibilities so it has a steep learning curve

• extensive use of operator overloading, function overloading and virtual
• functions can very quickly make C++ programs very complicated
• shortcuts offered in C++ can often make it completely unreadable, just like in C

2. Explain the basic concepts of object oriented programming in detail with example.

BASIC CONCEPTS OF OBJECT ORIENTED PROGRAMMING

These include:

? Objects

? Classes

? Data abstraction and encapsulation

? Inheritance

? Polymorphism

? Dynamic binding

? Message passing

Objects:

Objects are the basic run-time entities in an object oriented programming. They may represent a person, a place, a bank account or any item that the program has to handle. They may represent user-defined data such as vectors, time and lists. Programming problem is analyzed in terms of objects and the nature of communication b/n them.

When a program is executed, the objects interact by sending messages to another. Each object contains data and code to manipulate the data. Objects can interact without having to know the details of each others data or code. It is sufficient to know the type of message accepted, and the type of message accepted and the type of response returned by the objects.

Classes:

The entire set of data and code of an object can made a user defined data type with the help of a class. In fact the objects are variable of the type class. Once class has been defined we can create any number of objects belonging to that class. In short, a class serves as a blueprint or a plan or a template. It specifies what data and what functions will be included in objects of that class. Defining the class doesn't create any objects, just as the mere existence of a type int doesn't create any variables.

Data Abstraction and Encapsulation:

The wrapping up of data and functions into a single unit is known as encapsulation. It is the most striking feature of the class. The data is not accessible to the outside world and only those functions which are wrapped in the class can access it. These functions provide interface b/n the object's data and the program. This insulation of the data from direct access by the program is called data hiding or **information hiding.**

Abstraction represents the act of representing the essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on these attributes. The attributes are called data members and functions are called member functions or methods.

Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT).

Inheritance:

It is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. For example, the bird 'robin' is a part of the class 'flying bird' which is again a part of the class 'bird'.

This concept provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is possible by a deriving a new class from an existing one. The new class will have the combined features of both the classes.

Polymorphism:

It means the ability to take more than one form. An operation may exhibit different behavior in different instances. The behavior depends upon the types of data used in the operation. For example the operation addition will generate sum if the operands are numbers whereas if the operands are strings then the operation would produce a third string by concatenation. The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.

A single function name can be used to handle different types of tasks based on the number and types of arguments. This is known as function overloading.

It allows objects to have different internal structures to share the same external interface.

Dynamic Binding:

Binding refers to the linking of a procedure call to the code to be executed in response to the

call. Dynamic Binding (late binding) means the code associated with the given procedure call is not known until the time of the call at run-time. It is associated with the polymorphism and inheritance.

Message Passing:

The process of programming in OOP involves the following basic steps:

? Creating classes that define objects and their behavior

? Creating objects from class definitions

? Establishing communication among objects

A message for an object is request for execution of a procedure and therefore will invoke a function (procedure) in the receiving object that generates the desired result.

Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.

E.g.: employee.salary(name);

Object: employee

Message: salary

Information: name

3. State the rules to be followed while overloading an operator.write a program to illustrate overloading.

OPERATOR OVERLOADING:

• Operator overloading means giving additional meaning to existing operators

• By operator overloading an existing operator can be made to perform different operations than the stipulated one.

• It doesn't change the meaning and precedence of the original operator.

• Almost all the operators in c++ can be overloaded except the following

o Sizeof ()

o Conditional operator (?:)

o Scope resolution operator (::)

o Class member access operator (.,.*)

SYNTAX:

Return-type operator op-symbol(argument list)

{

body of the function;

}

Generally there are three general classifications of operator namely

Operator

Unary binary ternary

Among the above unary and binary operators can be overloaded and ternary operator cannot be overloaded.

OVERLOADING UNARY OPERATOR:

Unary operators like unary + , Unary - ….. can be overloaded as follows

EXAMPLE:

```
#include
#include
class unary
{
int a;
public:
void get()
{
a=10;
}
void show()
{
cout <<"\n The value of the object is\n"<
}
void operator - ()
{
a=-a;
}
};
void main()
{
```

```
clrscr();
unary u;
u.get();
u.show();
-u; //unary operator - called
u.show();
getch();
}
```

O/P:

```
The value of the object is
10
The value of the object is
-10
```

• In the above program, the unary operator minus is overloaded.

• In this an object ‘u’ is created and given the value 10

• When the unary operator – is called the operator function is invoked and the value of the member data of the object ‘u’ is negated.

• The negated value is then displayed.

4. (i)Describe the application of oop technology

Applications of OOP:

? Real time systems

? Simulation and modeling

? AI and expert systems

? Neural networks and parallel programming.

Decision support and office automation systems

(ii)What is an inline function?

INLINE FUNCTIONS :

An inline function is a function that is expanded in line when it is invoked. That is , the complier replaces the function call with the corresponding function code.The inline function are defined as follows:

inline function header

```
{
function body
}
```
Example :
```
inline int cube(int a)
{
return (a*a*a);
}
```
Program :
```
#include
inline float mul(float x, float y)
{
return (x*y);
}
inline float div(double p, double q)
{
return( p / q )
}
int main()
{
float a=12.35;
float b=9.32;
cout<< mul(a,b);
cout<
return 0;
}
```
5. (i).Explain copy constructor with suitable c++ coding.

Constructor:

A constructor is a special member function whose task is to initialize the objects of its class. It is special because its name is the same as the class name. The constructor is invoked whenever an object of its associated class it’s created.

A constructor is declared and defined as follows:

```
class sample
{
int m,n;
public:
sample()
{
m = n = 0;
}
// some code
};
void main()
{
sample s; // object is created
// some code
}
```

During the object creation, it also initializes the data member's m and n to 0.

Default Constructor:

A constructor that accepts no parameters is called the default constructor.

Characteristics of a constructor:

? They should be declared in the public section

? They are invoked automatically when the objects are created

? They do not have return types

? They cannot be inherited

? They can have default arguments

? It cannot be virtual

? We cannot refer to their addresses

? An object with a constructor or destructor can not be used as a member of a union

? They make 'implicit calls' to the operators new and delete when memory allocation is required

Copy constructor:

A copy constructor takes a reference to an object of the same class as itself as an argument.

Example:

It sets the value of every data element of s3 to the value of the corresponding data element of s2

Constructors with default arguments:

It is possible to define constructors with default arguments.

Example:

Inside the class definition:

sample(int a, int b = 10)

Inside the main():

sample s2(20);

assigns 20 to x and 10 to y.

where as

sample s5(25,35) ; assigns 25 to x and 35 to y

(ii).List out the rules for overloading operator.

OPERATOR OVERLOADING:

• Operator overloading means giving additional meaning to existing operators

• By operator overloading an existing operator can be made to perform different operations than the stipulated one.

• It doesn't change the meaning and precedence of the original operator.

• Almost all the operators in c++ can be overloaded except the following

o Sizeof ()

o Conditional operator (?:)

o Scope resolution operator (::)

o Class member access operator (.,.*)

SYNTAX:

Return-type operator op-symbol(argument list)

{

body of the function;

}

Generally there are three general classifications of operator namely

Operator

Unary binary ternary

Among the above unary and binary operators can be overloaded and ternary operator cannot be overloaded.

6. Compare and contrast the following control structure with example:

(i). break statement & continue statement.

Break	continue
a) Used to terminate the loops or to Exit loop from a switch.	Used to transfer the control to the Start of loop
b) The break statement when executed causes Immediate termination of loop containing it.	The continue statement when executed caused immediate termination of the current iteration of the loop.

(ii). Do – while and the while statement.

While loop	do-while loop
c) The while loop tests the condition before each iteration	The do-while loop tests the condition after the first iteration
d) If the condition fails initially the loop Is skipped entirely	Even if the condition fails initially the loop is executed once Iteration.

UNIT – II & III

ADVANCED OBJECT ORIENTED PROGRAMMING

Inheritance, Extending classes, Pointers, Virtual functions and polymorphism, File Handling Templates, Exception handling, Manipulating strings.

PART – A

1. What are the c++ operators that cannot be overloaded?

? Size operator (sizeof)

? Scope resolution operator (::)

? Class member access operators(. , .*)

? Conditional operator (?:)

2. What is a virtual base class?

When a class is declared as virtual c++ takes care to see that only copy of that class is inherited, regardless of how many inheritance paths exist between the virtual base class and a derived class.

3. What is the difference between base class and derived class?

The biggest difference between the base class and the derived class is that the derived class contains the data members of both the base and its own data members. The other difference is based on the visibility modes of the data members.

4. What are the rules governing the declaration of a class of multiple inheritance?

• More than one class name should be specified after the : symbol.

• Visibility modes must be taken care of.

If several inheritance paths are employed for a single derived class the base class must be appropriately declared

5. Mention the types of inheritance.

Single inheritance.

2. Multiple inheritance.

3. Hierarchical inheritance.

4. Multilevel inheritance.

5. Hybrid inheritance.

6. Define dynamic binding.

Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run-time.

7. What do u mean by pure virtual functions?

A pure virtual function is a function declared in a base class that has no definition relative to the base class. In such cases, the compiler requires each derived class to either define the function or redeclare it as a pure virtual function. A class containing pure virtual functions cannot be used to declare any objects of its own.

8. What are templates?

Templates enable us to define generic classes. A template can be considered as a kind of macro.

When an object of a specific type is defined for actual use, the template definition for that class is substituted with the required data type. Since a template is defined with a parameter that would be replaced by the specific data type at the time of actual use of the class or function, the templates are sometimes called parameterized classes or functions

9. What is exception handling?

The purpose of the exception handling mechanism is to provide a means to detect and report an "exceptional circumstance" so that appropriate action can be taken.

10. Give the general format of class template.

The general format of a class template is:

```
template
class classname
{
//……
//class member specification
//with anonymous type T
//wherever appropriate
//……
};
```

11. Can we overlaod template function if so explain how.

Yes, a template function can be overloaded either by template functions or ordinary functions of its name. in such cases, the overloading resolution is accomplished as follows:

1) Call an ordinary function that has an exact match.

2) Call a template function that could be created with an exact match.

3) Try normal overloading resolution to ordinary functions and call the one that matches.

12. List the kinds of exception.

Exceptions are of two kinds namely

• Synchronous exceptions

• Asynchronous exceptions

13. What are the errors in synchronous type exceptions?

Errors such as "out-of-range index" and "over-flow" belong to the synchronous type exceptions.

14. What are the errors in asynchronous type exceptions?

The errors that ware caused by errors beyond the control of the program (such as keyboard interrupts) are called asynchronous exceptions.

15. What are the tasks that are performed by the error handling mechanism?

The mechanism suggests a separate error handling code that performs the following tasks:

1) Find the problem (Hit the exception).

2) Inform that an error has occurred (Throw the exception).

3) Receive the error information (Catch the exception).

4) Take corrective actions (Handle the exception).

16. Mention the key words used in exception handling.

The keywords used in exception handling are

• throw

• try

• catch

17. Give the syntax of exception handling mechanism.

The syntax of exception handling mechanism is as follows:

```
try
{
----------------------
throw exception
----------------------
}
catch(type arguments)
{
----------------------
----------------------
}
----------------------
----------------------
```

18. List the ios format function.

The ios format functions are as follows:

• width()

• precision()

• fill()

• setf()

• unsetf()

19. List the manipulators.

The manipulators are:

• setw()

• setprecision()

• setfill()

• setiosflags()

• resetiosflags()

20. Mention the equicalent ios function for manipulators.

Manipulator Equivalent ios function

setw(int w) width()

setprecision(int d) precision()

setfill(int c) fill()

setiosflags(long f) setf()

resetiosflags(long f) unsetf()

endl "\n"

21. Define fill functions.

The fill( ) function can be used to fill the unused positions of the field by any desired character rather than by white spaces (by default). It is used in the following form:

cout.fill(ch);

where ch represents the character which is used for filling the unused positions. For example, the statements

cout.fill('*');

cout.width(10);

cout<<5250<<"\n";

will produce the following output:

PART – B

1. Explain hybrid inheritance with suitable C++ coding.

2. Explain multiple inheritances with suitable c++ coding.

INHERITANCE:

• Inheritance is a mechanism of deriving a new class from a old class.

• It provides the concept of reusability

• By inheritance some or all the properties of a class can be derived in to another class.

• The class which provides the properties is called as base class and the class which derives the properties is called as derived class.

Multiple inheritance:

It is a type of inheritance in which a class can inherit properties from more than one class.

Syntax:

Class derivedclass name : visibility mode baseclass1,visibilitmode baseclass2

{

body of the derived class;

};

visibility mode can be either private or public.

Example :

```
#include
#include
class bc1
{
protected:
int a;
public :
void get()
{
cout <<"\n enter the value for a\n";
cin >>a;
}
};
class bc2
{
```

```
protected:
int b;
public;
void get1()
{
cout <<" \n enter the value for b \n";
cin>>b;
}
};
class dc : public bc1, public bc2
{
public :
void show()
{
cout <<"The values of a and b are" ;
cout <<<"\n"<
}
};
void main()
{
clrscr();
dc d;
d.get();
d.get1();
d.show();
}
```

3. Define polymorphism. Explain the different types of polymorphism.

Polymorphism

Run time compile time

Or or

Dynamic binding static binding

Or or

late binding early binding

Virtual fuctions operator overloading function overloading

• Linking a function call to the function definition is called as binding.

• If the binding is done at compile time then it is called as compile time polymorphism.

• Compile time polymorphism is achieved by using function overloading and operator overloading.

• If the binding is done at run time then it is called as run time polymorphism. It is achieved by using virtual functions.

VIRTUAL FUNCTIONS:

• When the same function is used in both the base class and derived class and if a pointer of base class is used to access the members of the derived class , then the members appropriate to the base class is called.

Example:

```
#include
#include
class base
{
public:
void show()
{
cout <<"\n This is a base class\n";
}
};
class derived : public base
{
public:
void show()
{
cout <<"\n This is a derived class\n";
}
```

```
};
void main()
{
base b,*bptr;
bptr=&b;
bptr->show();
derived d;
bptr=&d;
bptr->show();
getch();
}
```

o/p

This is a base class

This is a base class

• This can be overcome by using virtual functions.

• The base class member functions should be preceeded by a keyword virtual.

Example:

```
#include
#include
class base
{
public:
virtual void show()
{
cout <<"\n This is a base class\n";
}
};
class derived : public base
{
public:
void show()
```

```
{
cout <<"\n This is a derived class\n";
}
};
void main()
{
base b,*bptr;
bptr=&b;
bptr->show();
derived d;
bptr=&d;
bptr->show();
getch();
}
```

o/p

This is a base class

This is a derived class

4. Explain multiple catch statement with help of suitable C++ coding.

EXCEPTION HANDLING

Exceptions are run time anomalies or unusual conditions that a program may encounter while executing. Anomalies might include conditions such as division by zero, access to an array outside of its bounds, or running out of memory or disk space. When a program encounters an exceptional condition, it is important that it is identified and dealt with effectively.

Exceptions are of two kinds, namely, synchronous exceptions and asynchronous exceptions. Errors such as "out-of-range index" and "over flow" belong to the synchronous exceptions. The errors that are caused by events beyond the control of the program are called asynchronous exceptions. The proposed exception handling mechanism is designed to handle only synchronous exceptions. The mechanism performs following tasks:

? Find the problem (Hit the exception).

? Inform that an error has occurred (Throw the exception).

? Receive the error information (Catch the expression).

? Take corrective actions (Handle the exceptions).

The error handling code basically consists of two segments one to detect errors and to throw exceptions, and other to catch the exceptions and to take appropriate actions.

Exception Handling Mechanism:

It is built upon three keywords, namely, try, throw and catch.

The keyword try is used to preface a block of statements which may generate exceptions. This block of statements is known as try block. When an exception is detected, it is thrown using a throw statement in the try block.

A catch block is defined by the keyword catch 'catches' the exception 'thrown' by the throw statement in the try block, and handles it appropriately. If the type of object thrown matches the arg type in the catch statement, then catch block is executed for handling the exception.

If they do not match the program is aborted with the help of the abort() function is invoked by default. When no exception is detected and thrown, the control goes to the statement immediately after the catch block.

Most often exceptions are thrown by the functions that are invoked from within the try blocks.

The point at which the throw is executed is called the throw point.

The general format of code for this kind of relationship is shown below

Multiple catch statements:

It is possible that a program segment has more than one condition to throw an exception. In such cases, we can associate more than one catch statement with a try as shown below:

```
try
{ // try block
}
catch(type1 arg)
{ // catch block1 }
catch(type2 arg)
{ // catch block2 }
…
…
catch(typeN arg)
{ // catch blockN }
```

Catch All Exceptions:

In some situations we may not be able to anticipate all possible types of exceptions we can force a catch statement to catch all exceptions instead of a certain type alone. This is achieved as follows:

```
catch (…)
{
// statement of processing all exceptions
}
```

5. Describe the various file modes and its syntax.

FILE

To handle large volumes of data, we need to use some devices such as floppy disk or hard disk to store the data. The data is stored in these devices using the concept of files. A file is a collection of related data stores in a particular area on the disk. Programs can be designed to perform the read and write operations on these files.

A program involves either or both of the following kinds of data communication:

1. Data transfer between the console unit and the program.
2. Data transfer between the program and a disk file.

The I/O system of C++ uses file stream as an interface b/n the programs and the files.

The stream that supplies data to the program is known as input stream. The stream that receives data from the program is known as output stream. In other words, the input stream reads data from the file and the output stream writes data to the file.

Classes for File Stream Operations:

The I/O system of C++ contains a set of classes that define the file handling methods. These include ifstream, ofstream and fstream. These classes are contained in the header file fstream.

This file should be included for performing file operations.

The filename is a string of characters that make up a valid filename for the operating system. It may contain two parts, a primary name and an optional period with extension.

Examples:

Input.txt

Student

For opening a file, we must create a file stream and then link it to the filename. There are two ways of opening a file:

? Using the constructor function of the class.

? Using the member function open() of the class.

The first method is useful when only one file in the stream is used. The second method is useful when multiple files are managed using one stream.

File Modes:

The two methods that we discussed can also take two arguments instead of one. The second argument will specify the file-mode. The general form of the function open() with two arguments is:

stream-object.open (“filename”, mode);

Parameter Meaning

ios::app Append to end of file

ios::ate Go to end of file on opening

ios::binary Binary file

ios::in Open file for reading only

ios::nocreate Open fails if the file does not exist

ios::noreplace Open fails if the file already exists

ios::out Open file for writing only

ios::trunc Delete the contents of the file if it exists

File pointers and their manipulations:

Each file has two associated pointers known as the file pointers. They are input pointer and output pointer. The input pointer is used for reading the contents of a given file location and the output pointer is used for writing to a given file location.

Read only mode:

Input pointer is automatically set at the beginning so that we can read the file from start.

Write only mode:

Existing contents are deleted and the output pointer is set at the beginning.

Append mode:

The output pointer is set to the end of file.

Functions for manipulations of file pointers:

seekg() – Moves input pointer to a specified location

seekp() – Moves output pointer to a specified location

tellg() – Gives the current position of the input pointer

tellp() – Gives the current position of the output pointer

Example:

infile.seekg(10);

moves file pointer to the byte number 10. The bytes are numbered from zero hence it points to the 11th byte in the file.

ofstream out

out.open("filename",ios::app);

int p = out.tellp();

The above program will give the number of bytes in the file, since the file is opened in the append mode.

6. Discuss the need for exception with try, catch and throw keywords.

Exceptions are run time anomalies or unusual conditions that a program may encounter while executing. Anomalies might include conditions such as division by zero, access to an array outside of its bounds, or running out of memory or disk space. When a program encounters an exceptional condition, it is important that it is identified and dealt with effectively.

Exceptions are of two kinds, namely, synchronous exceptions and asynchronous exceptions. Errors such as "out-of-range index" and "over flow" belong to the synchronous exceptions. The errors that are caused by events beyond the control of the program are called asynchronous exceptions. The proposed exception handling mechanism is designed to handle only synchronous exceptions. The mechanism performs following tasks:

? Find the problem (Hit the exception).

? Inform that an error has occurred (Throw the exception).

? Receive the error information (Catch the expression).

? Take corrective actions (Handle the exceptions).

The error handling code basically consists of two segments one to detect errors and to throw exceptions, and other to catch the exceptions and to take appropriate actions.

Exception Handling Mechanism:

It is built upon three keywords, namely, try, throw and catch.

The keyword try is used to preface a block of statements which may generate exceptions. This block of statements is known as try block.

When an exception is detected, it is thrown using a throw statement in the try block.

A catch block is defined by the keyword catch 'catches' the exception 'thrown' by the throw statement in the try block, and handles it appropriately. If the type of object thrown matches the arg type in the catch statement, then catch block is executed for handling the exception.

If they do not match the program is aborted with the help of the abort() function is invoked by default. When no exception is detected and thrown, the control goes to the statement immediately after the catch block.

Most often exceptions are thrown by the functions that are invoked from within the try blocks.

The point at which the throw is executed is called the throw point.

The general format of code for this kind of relationship is shown below:

Example:

```
#include
void divide(int a, int b)
{
if(b!=0)
cout << "Result = " << a/b;
else
throw(b);
}
void main()
{
cout << "Enter two numbers ";
int x,y;
cin >> x >> y;
try
{
divide(x,y);
}
```

```
catch(int i)
{
cout << “Error! Dividing by Zero “;
}
```

7. Explain the various forms of inheritance in C++ with necessary coding.

INHERITANCE:

• Inheritance is a mechanism of deriving a new class from a old class.

• It provides the concept of reusability

• By inheritance some or all the properties of a class can be derived in to another class.

• The class which provides the properties is called as base class and the class which derives the properties is called as derived class.

Types:

There are five types of inheritance viz

1. Single level inheritance
2. Multiple inheritance
3. Multilevel inheritance
4. Hybrid inheritance and
5. Hierarchical inheritance

Multiple inheritance:

It is a type of inheritance in which a class can inherit properties from more than one class.

Syntax:

```
Class derivedclass name : visibility mode baseclass1,visibilitmode baseclass2
{
body of the derived class;
};
```

visibility mode can be either private or public.

diagram:

refer diagram at pg 193 in E. Balaguruswamy.

Example :

```
#include
#include
```

```
class bc1
{
protected:
int a;
public :
void get()
{
cout <<"\n enter the value for a\n";
cin >>a;
}
};
class bc2
{
protected:
int b;
public;
void get1()
{
cout <<" \n enter the value for b \n";
cin>>b;
}
};
class dc : public bc1, public bc2
{
public :
void show()
{
cout <<"The values of a and b are" ;
cout <<<"\n"<
}
};
```

```
void main()
{
clrscr();
dc d;
d.get();
d.get1();
d.show();
}
```

output:

Enter the value for a

20

Enter the value for b

30

The values of a and b are

20

30

UNIT 4 & 5

21. Define ADT (Abstract Data Type)?

An ADT is a mathematical model with a collection of operations defined on that model.

22. Define linear data structure?

Linear data structures are data structures having a linear relationship between its adjacent elements. Eg: Linked List.

23. Define Non Linear data structure?

Non Linear data structures are data structures don‟t have a linear relationship between its adjacent elements, but had the hierarchical relationship. Eg: Graph, Tree.

24. What are different types of Linked List?

Single linked list Double linked list Circular linked list

25. What are different types of Circular Linked List?

Circular Single linked list Circular double linked list

26. List the basic operations carried out in a linked list?

a. Creation of list b. Insertion of list c. Deletion of list d. Modification of list e. Traversal of List

27. Define a stack?

Stack is an ordered collection of elements in which insertions and deletions are restricted to one end. The end from which elements are added and /or removed is referred to as top of the stack. Stacks are also referred as “piles” and “push-down lists”.

28. Define a queue?

Queue is an ordered collection of elements in which insertions and deletions are restricted to one end. The end from which elements are added and / or removed is referred to as the rear end and the end from which deletions are made is referred to as front end.

29. Difference between Arrays and Linked List?

Arrays	**Linked List**
Size of any array is fixed.	Size of list is variable.
It is necessary to specify the number of elements during declaration	
It is not necessary to specify the elements during declaration	
Insertion and deletions are difficult and costly.	Insertions and deletions are done in less time.
It occupies less memory than a linked List.	It occupies more memory.
Coding is easy.	Careful coding is needed to avoid memory

errors.

30. What is single linked list?

It is a linear data structure which contains a pointer field that points to the address of its next node (successor) and the data item.

31. Define HEAD pointer and NULL pointer?

HEAD - It contains the address of the first node in that list. NULL - It indicates the end of the list structure.

32 What is meant by dummy header?

It is ahead node in the linked list before the actual data nodes.

33. Define Circular linked list?

It is a list structure in which last node points to the first node there is no null value.

34. Write operations that can be done on stack?

PUSH and POP

35. Mention applications of stack?

a. Expression Parsing b. Evaluation of Postfix c. Balancing parenthesis d. Tower of Hanoi e. Reversing a string

36. Define Infix, prefix and postfix notations?

Infix operators are placed in between the operands Prefix operators are placed before the operands Postfix Operators are placed after the operands.

37. What are the conditions that followed in the array implementation of queue?

Condition Situation

REAR<FRONT EMPTY QUEUE

FRONT==REAR ONE ENTRY QUEUE

REAR==ARRAY SIZE FULL QUEUE

38. What are the conditions that could be followed in a linked list implementations of queue?

Condition Situation

REAR==HEAD EMPTY QUEUE

REAR==LINK (HEAD) ONE ENTRY QUEUE

NO FREE SPACE TO INSERT FULL QUEUE

39. What are the conditions that could be followed in a linked list implementations of queue?

Condition Situation

(REAR MOD ARRAY SIZE +1)==FRONT EMPTY QUEUE

(REAR MOD ARRAY SIZE +2)==FRONT FULL QUEUE

FRONT==REAR ONE ENTRY QUEUE

40. Define Circular queue?

A circular queue allows the queue to wrap around upon reaching the end of the array.

41. Define tree.

Trees are non-liner data structure, which is used to store data items in a shorted sequence. It represents any hierarchical relationship between any data Item. It is a collection of nodes, which has a distinguish node called the root and zero or more nonempty sub trees T1, T2,….Tk. each of which are connected by a directed edge from the root.

42. Define Height of tree.

The height of n is the length of the longest path from root to a leaf. Thus all leaves have height zero. The height of a tree is equal to a height of a root.

43. Define Depth of tree.

For any node n, the depth of n is the length of the unique path from the root to node n. Thus for a root the depth is always zero.

44. Define Degree of a node.

It is the number of sub trees of a node in a given tree.

45. Define Degree of a tree.

It is the maximum degree of a node in a given tree.

46. Define Terminal node or leaf?

Nodes with no children are known as leaves. A leaf will always have degree zero and is also called as terminal node.

47. Define Non-terminal node?

Any node except the root node whose degree is a non-zero value is called as a non-terminal node. Non-terminal nodes are the intermediate nodes in traversing the given tree from its root node to the terminal node.

48. Define sibling?

Nodes with the same parent are called siblings.

49. Define binary tree?

A Binary tree is a finite set of data items which is either empty or consists of a single item called root and two disjoin binary trees called left sub tree max degree of any node is two.

50. Define expression tree?

Expression tree is also a binary tree in which the leafs terminal nodes or operands and non-terminal intermediate nodes are operators used for traversal.

51. Define Construction of expression trees

1. Convert the given infix expression into postfix notation

2. Create a stack and read each character of the expression and push into the stack, if operands are encountered.

3. When an operator is encountered pop 2 values from the stack.

52. Define lazy deletion?

When an element is to be deleted it is left in the tree itself and marked as being deleted. This is called as lazy deletion and is an efficient procedure if duplicate keys are present in the binary search tree, because the field that keeps count of the frequency of appearance of the element can be decremented of the element can be decremented.

53. Define AVL tree?

AVL tree also called as height balanced tree .It is a height balanced tree in which every node will have a balancing factor of –1,0,1.

Balancing factor of a node is given by the difference between the height of the left sub tree and the height of the right sub tree.

54. What are the various operation performed in the binary search tree?

1. insertion
2. deletion
3. find
4. find min
5. find max

55. What are the various transformation performed in AVL tree?

1. Single rotation
- Single L rotation
- Single R rotation
2. Double rotation
-LR rotation
-RL rotation

56. General idea of hashing and what is the use of hashing function?

A hash table similar to an array of some fixes size-containing keys. The keys specified here might be either integer or strings, the size of the table is taken as table size or the keys are mapped on to some number on the hash table from a range of 0 to table size

57. What is priority queue?

A priority queue is a data structure that allows at least the following two operations: insert which does the obvious thing; and Deletemin, which finds, returns, and removes the minimum element in the priority queue. The Insert operation is the equivalent to enqueue.

58. Application of priority queues?

1. for scheduling purpose in operating system
2. used for external sorting
3. important for the implementation of greedy algorithm, which operates by repeatedly finding a minimum.

59. What are the main properties of a binary heap?

1. Structure property
2. Heap order property

60. Define tree traversal and mention the type of traversals?

Visiting of each and every node in the tree exactly is called as tree traversal

Three types of tree traversal

1. inorder traversal 2. preoder traversal 3. postorder traversal.

61. What is insertion sort? How many passes are required for the elements to be sorted?

One of the simplest sorting algorithms is the insertion sort. Insertion sort consist of N-1 passes. For pass P=1 through N-1 , insertion sort ensures that the elements in positions 0 through P-1 are in sorted order .It makes use of the fact that elements in position 0

through P-1 are already known to be in sorted order .

62. Write the function in C for insertion sort ?

```
Void insertionsort(elementtype A[ ] , int N)
{
int j, p;
elementtype tmp;
for(p=1 ; p <N ;p++ )
{
tmp = a[ p] ;
for ( j=p ; j>0 && a [ j -1 ] >tmp ;j--)
a [ j ]=a [j-1 ] ;
a [ j ] = tmp ;
}
}
```

63. Who invented shell sort? Define it?

Shell sort was invented by Donald Shell. It works by comparing element that are distant. The distance between the comparisons decreases as the algorithm runs until the last phase in which adjacent elements are compared. Hence it is referred as diminishing increment sort.

64. Write the function in c for shell sort?

```
Void Shellsort(Elementtype A[ ],int N)
{
int i , j , increment ;
elementtype tmp ;
for(elementtype=N / 2;increment > 0;increment / = 2)
For( i= increment ; i <N ; i ++)
{
tmp=A[ ];
for( j=I; j>=increment; j - =increment)
```

```
if(tmp< A[ ]=A[j – increment];
A[ j ]=A[ j – increment];
Else
Break;
A[ j ]=tmp;
}
}
```

65. What is maxheap?

If we want the elements in the more typical increasing sorted order, we can change the ordering property so that the parent has a larger key than the child. it is called max heap.

66. What are the two stages for heap sort?

Stage 1: Construction of heap

Stage 2: Root deletion N-1 times

67. What is divide and conquer strategy?

In divide and conquer strategy the given problem is divided into smaller problems and solved recursively. The conquering phase consists of patching together the answers. Divide and conquer is a very powerful use of recursion that we will see many times.

68. Differentiate between merge sort and quick sort?

Mergesort Quicksort

1. Divide and conquer strategy Divide and conquer strategy
2. Partition by position Partition by value

69. Mention some methods for choosing the pivot element in quicksort?

1. Choosing first element
2. Generate random number
3. Median of three

70. What are the three cases that arise during the left to right scan in quicksort?

1. I and j cross each other
2. I and j do not cross each other
3. I and j points the same position

71. What is the need of external sorting?

External sorting is required where the input is too large to fit into memory. So external sorting is necessary where the program is too large.

72. Define two way merge?

It is a basic external sorting in which there are two inputs and two outputs tapes.

73. Define multi way merge?

If we have extra tapes then we can expect to reduce the number of passes required to sort our input. We do this by extending two way merge to a k-way merge.

74. Define polyphase merge?

The k-way merging strategy requires the use of 2 k tapes. This could be prohibitive for some applications. It is possible to get by with only k+1 tapes.

75. What is replacement selection?

We read as many records as possible and sort them. Writing the result to some tapes. This seems like the best approach possible until one realizes that as soon as the first record is written to a output tape the memory it used becomes available for another record. If the next record on the input tape is larger than the record we have just output then it can be included in the item. Using this we can give algorithm. This is called replacement selection.

76. What is sorting?

Sorting is the process of arranging the given items in a logical order.

Sorting is an example where the analysis can be precisely performed.

77. What is mergesort?

The mergesort algorithm is a classic divide and conquer strategy. The problem is divided into two arrays and merged into single array

78. What are the properties involved in heapsort?

1. Structure property
2. Heap order property

79. Define articulation points.

If a graph is not biconnected, the vertices whose removal would disconnect the graph are known as articulation points.

80. Give some example of NP complete problems.

i. Hamiltonian circuit.
ii. Travelling salesmen problems
iii. Longest path problems
iv. Bin packing
v. Knapsack problem
vi. Graph coloring problem

81. What is a graph?

A graph consists of a set of vertices V and set of edges E which is mathematically represented as G=(V,E).Each edge in a pair(V,W) where V,W, belongs to E ,edges are sometimes referred to as arcs.

82. What are Directed graphs?

If a pair of vertices for any edge is ordered, then that graph is called as Digraph or directed graph.

83. Define Path.

A path in a graph is a sequence of vertices w1,w2w,3,wN such that Wi,Wi+1 belongs to E for a value 1<=I<=N. The length of such a path is the number of edges on the path, which is equal to n-1.

84. Define Cycle.

A cycle is a path in which the first and last vertices are the same.

85. Define Acyclic graph.

A graph with no cycles is called Acyclic graph. A directed graph with no Edges is called as a directed Acyclic graph (or) DAG. DAGS are used for Compiler Optimization process.

86. Define Connected graph.

An undirected graph is connected if there is a path from every vertex to every other vertex. A directed graph with this property is called as strongly connected graph. If a directed graph is not strongly connected but the underline graph. Without direction is connected it is called as a weakly connected graph.

87. What are the conditions for a graph to become a tree?

A graph is a tree if it has two properties.

i. If it is a connected graph.

ii. There should not be any cycles in the graph.

88. Define a Weighted Graph.

A graph is said to be a weighted graph if every edge in the graph is assigned some weight or value. The weight of the edge is a positive value that represents the cost of moving the edge or the distance between two vertices.

89. Give the types of representation of graphs.

1. Adjacency matrix

2. Adjacency linked list

90. What is a minimum spanning tree?

A minimum spanning tree of an undirected graph G is a tree formed from graph edges that connect all the vertices of G at lowest total cost.

91. Explain about Adjacency Matrix

Adjacency matrix consists of a n*n matrix where n is the no. of vertices present. In the graph, which consists of values either 0 or 1.

92. Explain about Adjacency linked list.

It consists of a table with the no. of entries for each vertex for each entry a Linked List is initiated for the vertices adjacent to the corresponding table entry.

93. What is a single source shortest path problem?

Given as an input, a weighted graph, G=<V,E> and a distinguished vertex „S‟ as the source vertex. Single source shortest path problem finds the shortest weighted path from s to every other vertex in G.

94. Explain about Unweighted shortest path.

Single source shortest path finds the shortest path from the source to each and every vertex present in a unweighted graph. Here no cost is associated with the edges connecting the vertices. Always unit cost is associated with each edge.

95. Explain about Weighted shortest path

Single source shortest path finds the shortest path from the source to each and every vertex present in a weighted graph. In a weighted graph some cost is always associated with the edges connecting the vertices.

96. What are the methods to solve minimum spanning tree?

a) Prim‟s algorithm

b) Kruskal‟s algorithm

97. Explain briefly about Prim‟s algorithm

Prim‟s algorithm creates the spanning tree in various stages. At each stage, a node is picked as the root and an edge is added and thus the associated vertex along with it.

98. Define a depth first spanning tree.

The tree that is formulated by depth first search on a graph is called as depth first spanning tree. The depth first spanning tree consists of tree edges and back edges.

99. What is a tree edge?

Traversal from one vertex to the next vertex in a graph is called as a tree edge.

100. What is a back edge?

The possibility of reaching an already marked vertex is indicated by a dashed line, in a graph is called as back edge.

101. Define double linked list?

It is linear data structure which consists of two links or pointer fields Next pointer points to the address of the next (successor) node. Previous pointer points to the address of the previous (predecessor) node.

16 MARK QUESTIONS

UNIT 1 2 & 3

1. Explain in detail about the features of Object Oriented paradigm with diagram.
2. a)What are the elements of Object Oriented Programming?

b) Explain object and classes with examples.

3. Explain in detail about File stream classes with an example.
4. What are the various categories of operators supported by C++? List any two types and give examples.

5. Explain the following with appropriate sample programs.

(i) for loop

(ii) do…while

6. Write programs for the following statements.

(i) Nested if…else

(ii) Switch statement

7. Write an interactive program to perform matrix multiplication.

8. Explain about the three types of Parameter passing with suitable examples.

9. What is Function overloading? Explain it with a suitable program.

10. a) What is a Pointer? Explain its need and advantages.

b) List the various pointer arithmetic operations and explain them with sample expressions.

11. Illustrate with a program class declaration, definition and accessing class members.

12. Explain the client-server model of object communication.

13. a) What is a constructor? Give its syntax.

b) Define Constructor overloading and illustrate its use with a suitable program.

14. a) Write a program to demonstrate the use of Copy constructor.

b) What is a destructor? Explain it with an example.

15. What is operator function? Describe operator function with syntax and examples.

16. Write a program to perform addition of complex numbers using stream loading.

17. a) List the different forms of Inheritance and explain.

b) Explain the Multiple Inheritance model with syntax and example.

18. Write a program to demonstrate the overloading of functions in base and derived classes.

19. Explain the difference between a normal Virtual function and Pure Virtual function With example.

20. What is an Abstract class? Write a program for testing the debuggable class.

UNIT 4&5

21.Explain the types of analysis that can performed on an algorithm?

_ Correctness analysis

_ Rate of growth

_ Time analysis

_ Order analysis

_ Space analysis

_ Example

22.i)Design an algorithm to reverse the linked list. Trace it with an example?

ii) Define an efficient representation of two stacks in a given area of memory with n words and explain.

i) algorithm

_ Define a stack and queue

_ Delete elements from queue and add it to stack

_ Pop elements from stack

_ Example and trace

ii) _ figure ,_ push operation pop operation advantages

23. Explain the process of problem solving?

_ Problem definition phase

_ Getting started with problem

_ The use of examples

_ Simulation among problems

_ Working backwards from solution

_ Problem solving techniques

24. Explain program verification in detail?

_ Input and output assertion

_ Computer model for program execution

_ Implications

_ Verification of program segments

_ Proof of termination

25. Explain top down design?

_ Breaking a problem into sub problem

_ Choice of suitable data structure

_ Construction of loops

_ Initial conditions for loops

_ Iterative construct

_ Termination of loops

26.What is a Stack?Explain with example?

_ Definition of Stack

_ Operations of Stack:PUSH and POP

_ Example

27.Write the algorithm for converting infix expression to postfix expression?

_ Definition of Expression

_ Types of expression

_ Algorithm for infix to postfix expression

_ Example

28.What is a Queue?Explain its operation with example?

_ Definition of Queue

_ Operations of Queue:insert and remove

_ Example

29.Explain the applications of stack?

_ Evaluating arithmetic expression

_ Balancing the symbols

_ Function calls

30. Write an algorithm for inserting and deleting an element from Doubly linked list? Explain linear linked implementation of Stack and Queue?

_ Introduction to Doubly linked list

_ Operations: insertion and deletion with algorithm

_ Linked list implementation of Stack

_ Linked list implementation of Queue

31.What is a Binary tree?Explain Binary tree traversals in C?

Definition of Binary tree

Traversals

Inorder traversal

Preorder traversal

Postorder traversal

32.Explain Representing lists as Binary tree?Write algorithm for finding Kth element and deleting an element?

Representing list as Binary tree

Finding Kth element

Deleting an element

33.Explain Binary search tree representation?

Node representation of Binary search tree

Implicit array representation of Binary tree

Implementation of various operations

34.What is a Priority Queue?What are its types?Explain?

Definition of Priority queue

Types:Ascending and Descending priority queue

Implementation of priority queue

35.Explain AVL tree in detail

Definition

Creation

Types of rotation

Deletion

36.Explain Heap sort?

Heap sort

_ Heap as a priority queue

_ Sorting using a heap

_ Heap sort procedure

37.Explain Insertion sort with example?

_ definition

_ algorithm and example

_ implementation

38.Explain merge sort with example?

_ definition

_ algorithm and example

_ implementation

_ basic merge algorithm with eg

39.Explain shell sort with example?

_ definition

_ algorithm and example

_ implementation

40.Explain quick sort with example?

_ definition

_ algorithm and example

_ implementation

41.Explain Shortest path algorithm with example?

_ Shortest path algorithm

_ Example

42.Explain Depth first and breadth first traversal?

_ Depth first traversal

_ Efficiency of Depth first traversal

_ Breadth first traversal

43.Explain spanning and minimum spanning tree?

_ Spanning tree

_ Minimum spanning tree

44.Explain Kruskal‟s and prim‟s algorithm?

_ Kruskal‟s algorithm

_ Prim‟s algorithm

45.Explain topological sorting?

_ definition _ algorithm and example _ implementation