ANNAI MATHAMMAL SHEELA ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CS6301 PROGRAMMING AND DATA STRUCTURE II
QUESTION BANK Yr/Sem - II/III Sem

UNIT I
PART-A (2 MARKS)

1. **State the characteristics of procedure oriented programming.**
- Emphasis is on algorithm.
- Large programs are divided into smaller programs called functions.
- Functions share global data.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.
- Employs top-down approach in program design.

2**. What are the features of Object Oriented Programming?**
- Emphasis is on data rather than procedure.
- Programs are divided into objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- New data and functions can easily be added whenever necessary.
- Follows bottom-up approach.

3. **Distinguish between Procedure Oriented Programming and Object Oriented Programming.**

Procedure Oriented Programming	Object Oriented Programming
• Emphasis is on algorithm. • Large programs are divided into smaller programs called functions. • Functions share global data. • Data move openly around the system from function to function. • Employs top-down approach in program design.	• Emphasis is on data rather than procedure. • Programs are divided into objects. • Functions that operate on the data of an object are tied together. • Data is hidden and cannot be accessed by external functions. • Follows bottom-up approach

4. Define Object Oriented Programming (OOP).

Object Oriented Programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.

5. List out the basic concepts of Object Oriented Programming.
- Objects
- Classes
- Data Abstraction and Encapsulation

- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

6. Define Objects.

Objects are the basic run time entities in an object oriented system. They are instance of a class. They may represent a person, a place etc that a program has to handle. They may also represent user-defined data. They contain both data and code.

7. Define Class.

Class is a collection of objects of similar data types. Class is a user-defined data type. The entire set of data and code of an object can be made a user defined type through a class.

8. Define Encapsulation and Data Hiding.

The wrapping up of data and functions into a single unit is known as data encapsulation. Here the data is not accessible to the outside world. The insulation of data from direct access by the program is called data hiding or information hiding.

9. Define Data Abstraction.

Abstraction refers to the act of representing the essential features without including the background details or explanations.

10. Define data members and member functions.

The attributes in the objects are known as data members because they hold the information. The functions that operate on these data are known as methods or member functions.

11. State Inheritance.

Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification and provides the idea of reusability. The class which is inherited is known as the base or super class and class which is newly derived is known as the derived or sub class.

12. State Polymorphism.

Polymorphism is an important concept of OOPs. Polymorphism means one name, multiple forms. It is the ability of a function or operator to take more than one form at different instances.

13. List and define the two types of Polymorphism.

- **Operator Overloading** – The process of making an operator to exhibit different behaviors at different instances.
- **Function Overloading** – Using a single function name to perform different types of tasks. The same function name can be used to handle different number and different types of arguments.

14. State Dynamic Binding.

Binding refers to the linking of procedure call to the code to be executed in response to the call. Dynamic Binding or Late Binding means that the code associated with a given procedure call is known only at the run-time.

15. Define Message Passing.

Objects communicate between each other by sending and receiving information known as messages. A message to an object is a request for execution of a procedure. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

16. List out some of the benefits of OOP.

- Eliminate redundant code
- Saves development time and leads to higher productivity
- Helps to build secure programs
- Easy to partition work
- Small programs can be easily upgraded to large programs
- Software complexity can easily be managed

17. Define Object Based Programming language.

Object Based Programming is the style of programming that primarily supports encapsulation and object identity. Languages that support programming with objects are known as Object Based Programming languages. They do not support inheritance and dynamic binding.

18. List out the applications of OOP.

- Real time systems
- Simulation and modeling
- Object oriented databases
- Hypertext, Hypermedia and expertext
- AI and expert systems
- Neural networks and parallel programming
- Decision support and office automation systems
- CIM/CAM/CAD systems

19. Define C++.

C++ is an object oriented programming language developed by Bjarne Stroustrup. It is a super set of C. Initially it was known as "C with Classes". It is a versatile language for handling large programs.

20. What are the input and output operators used in C++?

The identifier cin is used for input operation. The input operator used is >>, which is known as the extraction or get from operator. The syntax is,

cin >> n1;

The identifier cout is used for output operation. The input operator used is <<, which is known as the insertion or put to operator. The syntax is,

cout << "C++ is better than C";

21. What is the return type of main ()?

The return type of main () is integer i.e. main () returns an integer type value to the operating system. Therefore, every main () should end with a return (0) statement. It's general format is,

```
int main ()
{
……
return 0;    }
```

22. List out the four basic sections in a typical C++ program.

- Include files
- Class declaration
- Member functions definition
- Main function program

23. Define token. What are the tokens used in C++?

The smallest individual units in a program are known as tokens. The various tokens in C++ are keywords, identifiers, constants, strings and operators.

24. Define identifier. What are the rules to be followed for identifiers?

Identifiers refer to the names of variables, functions, arrays, classes etc created by the programmer. The rules are as follows:

- Only alphabetic characters, digits and underscores are permitted
- The name cannot start with a digit
- Uppercase and lowercase letters are distinct
- A declared keyword cannot be used as a variable name

25. State the use of void in C++.

The two normal uses of void are

- To specify the return type of the function when it is not returning a value
- To indicate an empty argument list to a function

26. Define an enumeration data type.

An enumerated data type is a user defined data type that provides a way for attaching names to numbers thereby increasing comprehensibility of the code. The **enum** keyword is used which automatically enumerates a list of words by assigning them values 0, 1, 2…

E.g.) enum shape {circle, square, triangle};

27. Define constant pointer and pointer to a constant.

The constant pointer concept does not allow us to modify the value initialized to the pointer.

E.g.) char * const ptr = "GOOD";

The pointer to a constant concept doesn't allow us to modify the address of the pointer.

E.g.) int const * ptr = &n;

28. What are the two ways of creating symbolic constants?

- Using the qualifier const
- Defining a set of integer constants using enum keyword

29. Define reference variable. Give its syntax.

A reference variable provides an alias or alternate name for a previously defined variable. It must be initialized at the time of declaration. Its syntax is given by,

data-type & reference-name = variable-name;

30. List out the new operators introduced in C++.

- :: Scope resolution operator
- ::* Pointer to member declaration
- ->* Pointer to member operator
- .* Pointer to member operator
- delete Memory release operator
- endl Line feed operator
- new Memory allocation operator
- setw Field width operator

31. What is the use of scope resolution operator?

A variable declared in an inner block cannot be accessed outside the block. To resolve this problem the scope resolution operator is used. It can be used to uncover a hidden variable. This operator allows access to the global version of the variable. It takes the form,

:: variable-name

32. List out the memory differencing operator.

- ::* To declare a pointer to the member of the class
- ->* To access a member using object name and a pointer to that member
- .* To access a member using a pointer to the object and a pointer to that member

33. Define the 2 memory management operators.

• new Memory allocation operator

The new operator can be used to create objects of any data-type. It allocates sufficient memory to hold a data object of type data-type and returns the address of the object.
Its general form is,

Pointer variable=new data-type;

• delete Memory release operator

When a data object is no longer needed it is destroyed to release the memory space for reuse. The general form is,

delete pointer variable;

34. List out the advantages of new operator over malloc ().

- It automatically computes the size of the data object.
- It automatically returns the correct pointer type.
- It is possible to initialize the objects while creating the memory space.
- It can be overloaded.

35. Define manipulators. What are the manipulators used in C++?

Manipulators are operators that are used to format the data display. The manipulators used in C++ are

- endl – causes a linefeed to be inserted
- setw – provides a common field width for all the numbers and forces them to be printed

right justified

36. What are the three types of special assignment expressions?

- Chained assignment e.g., x = y = 10;
- Embedded assignment e.g., x = (y = 50) + 10;
- Compound assignment e.g., x + = 10;

37. Define implicit conversion.

Whenever data types are mixed in a expression, C++ performs the conversions automatically. This process is known as implicit or automatic conversion.

e.g., m = 5 + 2.75;

38. Define integral widening conversion.

Whenever a char or short int appears in an expression, it is converted to an int. This is called integral widening conversion.

39. What are the control structures used in C++?

- Sequence structure (straight line)
- Selection structure (branching)
    - if – else (two way branch)
    - switch (multiple branch)
- Loop structure (iteration or repetition)
    - do – while (exit controlled)
    - while (entry controlled)
    - for (entry controlled)

40. Define Function Prototyping.

The function prototype describes the function interface to the compiler by giving details such as the number and type of arguments and type of return values. It is the declaration of a function in a program. It is in the following form,

type function – name (argument – list);

where argument – list -> types and names of arguments to be passed to the function

- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

When we pass arguments by reference, the formal arguments in the called function become the aliases to the actual arguments in the calling function. Here the function works on the original data rather than its copy.

```
e.g., void swap (int &a, int &b)
{
int t = a;
a = b;
b = t;
}
```

42. What are inline functions?

An inline function is a function that is expanded in line when it is invoked. Here, the compiler replaces the function call with the corresponding function code. The inline function is defined as,

```
inline function-header
{
function body
}
```

43. List out the conditions where inline expansion doesn't work.

- For functions returning values, if a loop, a switch, or a goto exists
- For functions not returning values, if a return statement exists
- If functions contain static variables
- If inline functions are recursive

44. Why do we use default arguments?

The function assigns a default value to the parameter which does not have a matching argument in the function call. They are useful in situations where some arguments always have the same value. **e.g., float amt (float P, float n, float r = 0.15);**

45. State the advantages of default arguments.

The advantages of default arguments are,

- We can use default arguments to add new parameters to the existing function.
- Default arguments can be used to combine similar functions into one.

46. Define function overloading.

A single function name can be used to perform different types of tasks. The same function name can be used to handle different number and different types of arguments. This is known as function overloading or function polymorphism.

47. List out the limitations of function overloading.

We should not overload unrelated functions and should reserve function overloading for functions that perform closely related operations.

48. State the difference between structures and class.

By default the members of a structure are public whereas the members of a class are private.

49. Define a class.

A class is a way to bind the data and its function together. It allows the data to be hidden from external use. The general form of a class is,

```
class class_name
{
private:
        variable declarations;
        function declaration;
public:
        variable declarations;
        function declaration;
};
```

50. List the access modes used within a class.

• **Private** – The class members are private by default. The members declared private are completely hidden from the outside world. They can be accessed from only within the class.

• **Public** – The class members declared public can be accessed from anywhere.

• **Protected** – The class members declared protected can be access from within the class and also by the friend classes.

51. How can we access the class members?

The class members can be accessed only when an object is created to that class. They are accessed with the help of the object name and a dot operator. They can be accessed using the general format,

Object_name.function_name (actual_arguments);

52. Where can we define member functions?

Member functions can be defined in two places:

• **Outside the class definition** – The member functions can be defined outside the class definition with the help of the scope resolution operator. The general format is given as,

return_type class_name :: function_name (argument declaration)
{
function body
}

• **Inside the class definition** – The member function is written inside the class in place of the member declaration. They are treated as inline functions.

53. What are the characteristics of member functions?

The various characteristics of member functions are,

- Different classes can use the same function name and their scope can be resolved using the membership label.
- Member functions can access the private data of a class while a nonmember function cannot.
- A member function can call another member function directly without using a dot operator.

54. How can an outside function be made inline?

An outside function can be made inline by just using the qualifier 'inline' in the header line of the function definition. The general format is,

inline return_type class_name :: function_name (argument declaration)
{
function body
}

55. What are the properties of a static data member?

The properties of a static data member are,

- It is initialized to zero when the first object is created and no other initialization is permitted.
- Only one copy of that member is created and shared by all the objects of that class.
- It is visible only within the class, but the life time is the entire program.

56. What are the properties of a static member function?

• A static member function can access only other static members declared in the same class.

• It can be called using the class name instead of objects as follows,

class_name :: function_name;

57. How can objects be used as function arguments?

An object can be used as a function argument in two ways,

• A copy of the entire object is passed to the function. (Pass by value)

• Only the address of the object is transferred to the function. (Pass by reference)

58. Define friend function?

An outside function can be made a friend to a class using the qualifier **'friend'**. The function declaration should be preceded by the keyword friend. A friend function has full access rights to the private members of a class.

59. List out the special characteristics of a friend function.

• It is not in the scope of a class in which it is declared as friend.

• It cannot be called using the object of that class.

• It can be invoked without an object.

• It cannot access the member names directly and uses the dot operator.

• It can be declared as either public or private.

• It has the objects as arguments.

PART-B (16 MARKS)

1. Explain with the Basic Concepts of object oriented programming.
2. (a) Explain the elements of object oriented programming.
   (b) What are the difference between reference variables and normal variables?
3. Explain about call-by-reference and return by reference.
4. (a) Describe the advantages of OOP.
   (b) What are the difference between pointers to constants and constant to pointers?
5. (a) Describe the applications of OOP technology.
   (b) What is function overloading? Explain with an example program.
6. Explain the merits and demerits of object oriented methodology.
7. What is friend function? What is the use of using friend functions in c++? Explain with a program.
8. What is polymorphism? Provide an example to explain it.
9. Describe Abstract base class. Illustrate an example to explain it.
10. What are the advantages of using default arguments? Explain with an example program.
11. Write a program to implement nested classes using c++.
12. Write a program to demonstrate how a static data is accessed by a static member function.
13. Write a program to get the student details and print the same using pointers to objects and pointers to members of a class. Create a class student. And use appropriate functions and data members.
14. Write a program to define a class car. It should contain Make, Color, Size and cost as data members. Write member functions for reading and printing values of car. Define one

more class as carcollection. Carcollection should contain member functions as Add, Delete, and Modify. Carcollection is to be defined as friend of car.

UNIT –II
PART-A (2 MARKS)

1. Define Constructor.

A constructor is a special member function whose task is to initialize the objects of its class. It has the same name as the class. It gets invoked whenever an object is created to that class. It is called so since it constructs the values of data members of the class.

2. List some of the special characteristics of constructor.

- Constructors 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.

3. Give the various types of constructors.

There are four types of constructors. They are

- Default constructors – A constructor that accepts no parameters
- Parameterized constructors – The constructors that can take arguments
- Copy constructor – It takes a reference to an object of the same class as itself as an argument
- Dynamic constructors – Used to allocate memory while creating objects

4. What are the ways in which a constructor can be called?

The constructor can be called by two ways. They are,

- By calling the constructor explicitly
  e.g., integer int1 = integer (0, 100);
- By calling the constructor implicitly
  e.g., integer int1 (0, 100);

5. State dynamic initialization of objects.

Class objects can be initialized dynamically. The initial values of an object may be provided during run time. The advantage of dynamic initialization is that various initialization formats can be used. It provides flexibility of using different data formats.

6. Define Destructor.

A destructor is used to destroy the objects that have been created by a constructor. It is a special member function whose name is same as the class and is preceded by a tilde '~' symbol.

7. Give the general form of an operator function.

The general form of an operator function is given as,

return-type class-name :: operator op (arglist)
{
function body
}

Where,

Return-type -> type of value returned

operator -> keyword
op -> operator being overloaded

8. List some of the rules for operator overloading.

- Only existing operators can be overloaded.
- We cannot change the basic meaning of an operator.
- The overloaded operator must have at least one operand.
- Overloaded operators follow the syntax rules of the original operators.

9. What are the types of type conversions?

There are three types of conversions. They are

- Conversion from basic type to class type – done using constructor
- Conversion from class type to basic type – done using a casting operator
- Conversion from one class type to another – done using constructor or casting operator

10. What are the conditions should a casting operator satisfy?

The conditions that a casting operator should satisfy are,

- It must be a class member.
- It must not specify a return type.
- It must not have any arguments.

11. What is parameterized constructor?

The constructor that takes arguments are called parameterized constructor. When a constructor has been parameterized the object declaration statement such as integer i will not work.

12. What are the kinds of constructor that we call?

- Constructors without arguments
- Constructors with arguments

13. What are copy constructors? Explain with examples?

The constructor that creates a new class from an existing object of the same class. E.g., integer i2 (i1) or integer i2=i1 would define the object i2 at the same time initialize the value of i1.

14. Define dynamic constructor?

Allocation of memory to objects at the time of their construction is known as dynamic constructor.

15. What is the advantage of using dynamic initialization?

The advantage of using dynamic initialization is that various initialization formats can be provided using overloaded constructor.

16. Define operator overloading?

A language feature that allows a function or operator to be given more than one definition. For instance C++ permits to add two variables of user defined types with the same syntax that is applied to the basic types. The mechanism of giving such special meaning to an operator is known as operator overloading.

17. Give the operator in C++ which cannot be overloaded?

- Sizeof → size of operator
- **::** → scope resolution operator
- **?:** → conditional operator
- **.** → membership operator
- ***** →pointer to member operator

18. What are the steps involves in the process of overloading?

- Creates a class that defines the data type that is to be used in the overloading operation.
- Declare the operator function op( ) in the public part of the class.
- Define the operator function to implement the required operation.

19. What are the restriction and limitations of overloading operators?

Operator function must be member functions or friend functions. The overloading operator must have at least one operand that is user defined datatype.

20. Give a function overload a unary minus operator using friend function?

```
friend void operator –(space &s)
{
s.x=-s.x;
s.y=-s.y;
s.z=-s.z;
}
```

21. Define unary and binary operator overloading?

Overloading without explicit argument to an operator function is known as **Unary operator overloading.**

Overloading with single explicit argument is known **Binary operator overloading.**

22. Explain overloading of new and delete operators?

The memory allocation operators **new** and **delete** can be overloaded to handle memory resource in a customized way. The main reason for overloading these functions is to increase the efficiency of memory management.

23. Define type conversion?

A conversion of value from one datatype to another.

24. When and how the conversion function exists?

To convert the data from a basic type to user defined type. The conversion should be defined in user defined object's class in the form of a constructor. The constructor function takes a single argument of basic data type.

25. Give the syntax for overloading with friend function?

```
friend returntype operator op(argument)
{
body of the function
}
```

26. Define inheritance?

The mechanism of deriving a new class from an old one is called **inheritance**. The old class is referred to as the **base class** and the new one is called the **derived class** or the **sub class**.

27. What are the types of inheritance?

The various types of inheritance are,

- Single inheritance
- Multi-level inheritance
- Multiple inheritance
- Hierarchical inheritance
- Hybrid inheritance

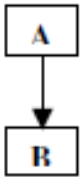
28. Give the syntax for inheritance.

The syntax of deriving a new class from an already existing class is given by,

```
class derived-class : visibility-mode base-class
{
body of derived class
}
```

29. Define single inheritance.

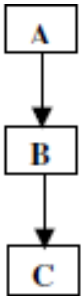
In single inheritance, one class is derived from an already existing base class.



Here A is the base class and B is the derived class.

30. Define multi-level inheritance.

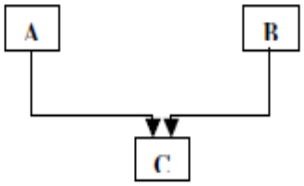
In multi-level inheritance, a new class is derived from a class already derived from the base class.



Here, class B is derived from class A and class C is further derived from the derived class B.

31. Define multiple inheritance.

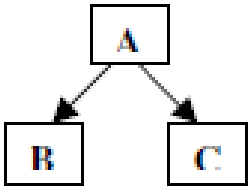
In multiple inheritance, a single class is derived from more than one base class.



Here class C is derived from two base classes A and B.

32. Define Hierarchical inheritance.

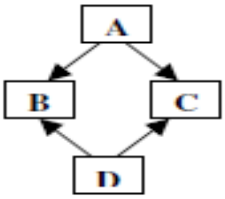
In hierarchical inheritance, more than one class is derived from a single base class.



Here class B and C are derived from class A.

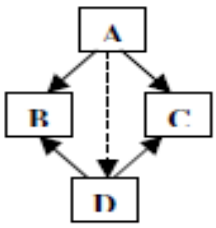
33. Define Hybrid inheritance.

Hybrid inheritance is defined as a combination of more than one inheritance.



Here, Classes A, B and C represent hierarchical inheritance. Classes A, B & D and classes A, C and D represent multilevel inheritance. Classes B, C and D represent multiple inheritance.

34. What is a virtual base class?



Here, class D inherits both classes B and C which are derived from the same base class A. Hence D has two copies of the properties of class A. This can be avoided by declaring classes B and C as virtual base classes.

35. What is an abstract class?

An abstract class is one that is not used to create objects. It is designed only to act as a base class to be inherited by other classes.

36. What are the types of polymorphism?

The two types of polymorphism are,

• **Compile time polymorphism** – The compiler selects the appropriate function for a particular call at the compile time itself. It can be achieved by function overloading and operator overloading.

• **Run time Polymorphism** - The compiler selects the appropriate function for a particular call at the run time only. It can be achieved using virtual functions.

37. Define 'this' pointer.

A 'this' pointer refers to an object that currently invokes a member function. For e.g., the function call a. show () will set the pointer 'this' to the address of the object 'a'.

38. What is a virtual function?

When a function is declared as virtual, C++ determines which function to use at run time based on the type of object pointed to by the base pointer, rather than the type of the pointer.

39. What is a pure virtual function?

A virtual function, equated to zero is called a pure virtual function. It is a function declared in a base class that has no definition relative to the base class.

40. How can a private member be made inheritable?

A private member can be made inheritable by declaring the data members as protected. When declared as protected the data members can be inherited by the friend classes.

41. What are the rules for virtual function?

- They cannot be static members
- They can access by using object pointers
- A virtual function can be a friend of another class.

42. What is RTTI?

Runtime type identification (RTTI) lets you find the dynamic type of an object when you have only a pointer or a reference to the base type. RTTI is the official way in standard C++ to discover the type of an object and to convert the type of a pointer or reference (that is, dynamic typing). The need came from practical experience with C++. RTTI replaces many homegrown versions with a solid, consistent approach.

43. List out RTTI

- Base types
- Compound types
- Container types

PART-B (16 MARKS)

1. Explain copy constructor and destructor with suitable C++ coding.
2. Explain about static member and this pointer with suitable code.
3. What is a Bit field? Explain briefly.
4. Explain about Data Handling and member function.
5. What is a virtual destructor? Explain the use of it.
6. Explain about Unary Operator and Binary Operator Overloading with program.
7. List out the rules for overloading operators with example.
8. Define a supplier class. Assume that the items supplied by any given supplier are different and varying in number. Use dynamic memory allocation in the constructor function to achieve the solution.
9. Define an examiner class. Provide all necessary data and function members to provide the following: The examiner must access answer sheets of at least one subject; He may examine answer sheets of multiple subjects; The examiner represents a college and also a university; Most of the examiners are local and represent local university; and have more than one constructor including one default and one with default argument. Provide a meaningful copy constructor.
10. For a supermarket, define a bill class. All the bill objects will contain bill number, name of clerk preparing the bill, each item with quantity and price and total amount to be paid. Total items in the bill are varying. Define dynamic memory allocation constructor for bill class such that any number of items from 1 to 50 can be accommodated in a single bill. There is an array describing each item with a price. The price is to be picked up from that array. Now overload = operator and provide reason for the need of such operator.
11. Write a program to define a class person with more than three constructors. Define data and function members in the class in such a way that all three constructors are meaningful.
12. What are the virtual functions? Explain their needs using a suitable example. What are the rules associated with virtual functions?
13. What are the different forms of inheritance supported in c++? Discuss on the visibility of base class members in privately and publicly inherited classes.
14. What are abstract classes? Give an example (with the program) to illustrate the use of abstract classes.
15. Explain about Code Reuse with program.
16. Write notes on Typing conversions and derived class with program.
17. Define a student class. Inherit that into MCAStudent class and NonMCAStudent. MCAStudents inherits into GLSSTudents and NonGLSStudents. A function ShowPracticalHours can only be applied to MCAStudents. We have a base class Student pointer to a GLSStudent object. Use dynamic_cast ti check that NonMCAStudents do not ShowPracticalHours

UNIT-III
PART –A (2 MARKS)

1. Define Templates?

In C++ Template is defined as a generalized form of reusable modules such as object, classes, functions etc.

2. What are the advantages of using template?

- It supports all data types
- It reduces number of objects, classes and functions in programs
- This means with the help of single function or class we can do so many operation with different data types.

3. Define Class Template?

A class created from a class template is called a template class.

4. Function Template?

Template is a method for writing a single function or class for a family of similar function or classes in generic manner. When a single function is writing for a family of similar function it called as "**function template**". In this function at least one formal argument is generic.

5. What are the rules for declaring template?

- The keyword template should be placed in front of function name
- The function template call is same as ordinary function call
- The user can use any number for template class data type normally we can use T
- The argument list should contain at least one argument from each tamplate class data type

6. What is the syntax used to declare Function Template with multiple parameters?

Syntax:

```
Template<class T1, class T2…>
 Return type fun _ name(arguments T1,T2,T3,…)
 {
   Body of the function;
   }
```

7. How will you overload Function template?

A template function may be overloaded either by defining template function or ordinary functions of its name. That is to overload template function we need at least one template function.

8. What is Exception Handling?

Exception handling is the programming technique. Using this method, run time errors can be detected and reported to the user for taking necessary action. In another words, it is defined as the run time anomalies that a program may encounter while executing the coding.

9. List out the steps of exception handling?

- Find the problem (hit the exception)
- Inform the error that has occurred (throw the exception)
- Receive the error details (catch the exception)

10. Define Errors and its types?

Errors are produced while we writing a program. Technically, Errors are classified into three major categories. They are

- Syntax Error
- Logical Errors
- Exception (or) runtime Error

11. Distinguish between syntax and logical errors?

Syntax Error

This type of error occurs when the user violates the language's rule and regulations. This happens because of the poor understanding of the language by the user.

Logical Error

This type of error occurs when the programmer violates the logical concepts of the language. This happen because of the poor understanding of the problem solving by the user. These errors can be identified only by the wrong output for the right input.

12. What is mean by Synchronous Exceptions?

It is defined as the errors such as "**out of range**" and "**overflow**" belongs to the synchronous exceptions.

13. What are the three keywords of exception handling Mechanism?

C++ Exception handling mechanism is basically built upon the three keywords those are:

- Try block
- Throw block
- Catch block

14. What is the relationship between the try, catch and throw block?

A try block must enclose the statements that can throw exceptions. A try block begins with the try keyword followed by a sequence of program statements enclosed in braces. Following the try block is a list of handlers called catch clauses. The try block groups a set of statements and associates with these statements a set of handlers to handle the exceptions that the statements can throw. Where should we place a try block or try blocks in the function main () to handle the exceptions.

15. Define Unexpected( ) and Terminate( ) Functions?

Two special library functions are implemented in C++ to process exceptions not properly handled by catch blocks or exceptions thrown outside of a valid try block. These functions are unexpected( ) and terminate( ).

16. How will you specifying list of Exceptions?

This means, defining the possible exceptions expected in a program by the programmer. This can be implemented with the help of functions. This function should be defined before the try block.

General form:

Type functions (arg list) throw (type _ list)
{
……..
……...
}

17. What is mean by Catching mechanism?

The catch statement catches an exception whose type matches with the type of catch argument. When it is caught, the code in the catch block is executed.

18. What is meant by Asynchronous Exceptions?

Errors that are caused by events beyond the control of the programs are called asynchronous exceptions.

- Disk Error
- Keyboard problems etc.

19. Define Member function template?

If a function present within the class template is called member function template.

20. Describe the rules for function template?

Rules:

- The keyword template should be placed in front of function name.
- The function template call is same as ordinary function call.
- The user can use any number for template class data type normally we can use T.
- The argument list should contain at least one argument from each template class data type.

21. What is stream and its types?

Stream is defined as the flow of data between the console files and the data. Streams are classified into two broad categories. They are

- Input stream
- Output stream

22. Define Input stream?

The flow of data from the input device to program is called input streams.

23. What are console stream classes in C++?

- Cin - standard input (keyword)
- Cout - standard output (screen)
- Cerr - standard error output (screen)
- Clog - a fully buffered version of cerr

24. What are the file stream classes in C++?

- filebuf
- fstreambase

- ifstream
- ofstream
- fstream

25. What are the file manipulation function in C++?

- seekg( )
- seekp( )
- tellg( )
- tellp( )

26. What are the file open modes?

Ios::app, ios::binary, ios::oput, ios::ate, isos::nocreate, ios::noreplace, ios::trunk.

27. What are the error handling function in C++?

- eof( )
- fail( )
- bad( )
- good( )

28.Define getline( )?

This function is used to reads a whole line of text that ends with a new line character and it is accessed with the help of the object cin.

29. What are unformatted output operation in C++?

The input operations are used to carry out the following member functions. They are

- overload operator(<<)
- put()
- write()

30. What are the different types of formatted console I/O operations?

- Manipulators
- ios class function and flags
- User defined output functions
- The ios class contains a large number of member functions that would help us to format the output in a number of ways.

31. What do you meant by Manipulators?

Manipulators are special functions that are specifically designed to modify the working of stream.They can be embedded in the I/O statements to

Example:

```
cout<<manip1<<manip2<<item;
cout<<manip1<<item1<<manip<<item2
```

32. List out the parameterized manipulators?

- setw()
- setprecision()
- setfill()
- setbase()

33. What are non-parameterized manipulators?

- flush
- end1

34. Distinguish between width() and precision ()function?

Width():

This function is used to specify the width of the data to be printed. Since it is a member function of the class ios, this is accessed with the help of the object cout.

Precision():

This function is used to specify the number of digits to be displayed after the decimal point. Since it is a member function, this is accessed with the help of object cout.

35. Define file?

A file is a collection to inter related data stored in a particular area on the disk.

36. What is the syntax used for closing a file give example?

file object.close();

where

file object - opened file name

close - member function

Example:

Employee. Close (); // this closes the file std.dat with the file object employee

37. Define Default (or) Random Access?

Read-only mode

Read mode, when a file is opened in opened in read-only mode; the input (get) pointer is initialized to point to the beginning of the file, so that the file can be read from the start.

Read Mode

H	L	L	O		C	S	E

↑Input pointer

38. List out the different types of modes used in fine concepts?

- Read-only mode
- Write –only mode
- Append Mode

39. Define namespace?

It is a feature in C++ to minimize name collisions in the global name space. This namespace keyword assigns a distinct name to a library that allows other libraries to use the same identifier names without creating any name collisions. Furthermore, the compiler uses the namespace signature for differentiating the definitions.

40. How will you declare namespace?

Syntax

```
namespace identifier
{
 Body of the statements;
 }
```

41. Define Namespace Alias?

A namespace alias is an alternative name for a **namespace**. A namespace alias definition declares an alternate name for a namespace.

42. Define Namespace Std?

All the files in the C++ standard library declare all of its entities within the std namespace. That is why we have generally included the using namespace std; statement in all programs that used any entity defined in iostream.

43. What problem does the namespace feature solve?

Multiple providers of libraries might use common global identifiers causing a name collision when an application tries to link with two or more such libraries. The namespace Feature surrounds a library's external declarations with a unique namespace that eliminates the potential for those collisions. This solution assumes that two library vendors don't use the same namespace.

44. Define Standard Template Library (Stl)?

A library of container templates approved by the ANSI committee for inclusion in the standard C++ specification. A programmer who then launches into a discussion of the generic programming model, iterators, allocators, algorithms, and such, has a higher than average understanding of the new technology that STL brings to C++ programming.

45. What is a container class? What are the types of container classes?

A container class is a class that is used to hold objects in memory or external storage. A container class acts as a generic holder. A container class has a predefined behavior and a well-known interface. A container class is a supporting class whose purpose is to hide the topology used for maintaining the list of objects in memory. When a container class contains a group of mixed objects, the container is called a heterogeneous container; when the container is holding a group of objects that are all the same, the container is called a homogeneous container.

46. STL containers-What are the types of STL containers?

There are three types of STL containers;

- Adaptive containers like queue, stack
- Associative containers like set, map
- Sequence containers like vector, deque

47. What are the different types of STL Algorithm?

- Retrieve or non-mutating algorithms
- Mutating algorithms
- Sorting algorithms
- Set algorithms
- Relational algorithms

PART-B (16 MARKS)

1. What is the need of Templates? Explain.
2. Explain about function templates?
3. Implement selection sort as a generic function.

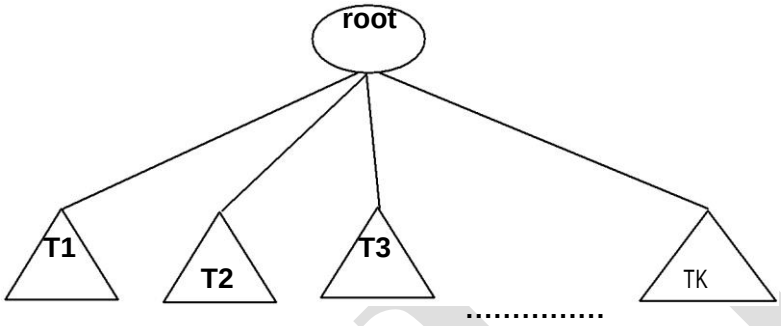
4. Implement quick sort as a generic function.
5. Write a program for generic queue class with two member functions, insert and delete. Use the array to implement the queue.
6. Define a stack. The class should throw an exception when the stack underflow and overflow takes place.
7. Using the Time class, throw an exception when invalid time is input, write set_terminate to provide your own terminate function, which care of this problem.
8. What is uncaught exception function? Give an example.
9. What are the use of terminate() and Unexpected functions? Explain with a program.
10. How to use multiple catch functions inside a program? Explain with a program.
11. Write all blocks of exception handling? Explain with a program.
12. Explain about Template and its types with example.
13. Discuss about Streams and stream classes
14. Write notes on Formatted and Unformatted Console I/O Operations.
15. Explain about File Pointers and Manipulations with example.
16. Discuss about manipulators and file streams with Program.
17. Write on Details about File modes and File I/O.
18. Write notes on Formatted and Unformatted Console I/O Operations.
19. Explain about File Pointers and their manipulations with example.
20. Give the differences between Manipulators and ios Functions.
21. How can we determine errors while dealing with files.
22. Explain in detail about the facilities available for substring operations on the string object?
23. Explain in detail about Sorted Associative Containers.
24. Discuss about different ways of defining namespaces.
25. Explain in detail about Adapted Containers.
26. a) Explain Text stream and Binary stream. (8)
27. b) Write the difference between Text and Binary stream. (8)
28. What is Text stream? Explain the operations of text stream with an example.
29. What is Binary stream? Explain the operations of binary stream with an example.

UNIT-III
PART –A (2 MARKS)

TREE STRUCTURES

1. Define Tree .Give an example.

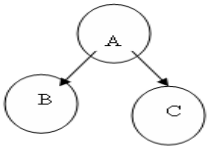
A tree is a collection of nodes .The collection can be empty .Otherwise a tree consists of a distinguished node r called the root and 0 or more non empty sub-trees T_1,T_2,T_3…….T_k each of whose roots are connected by a directed edge from r.



Eg: directory structure hierarchy

2. Define root

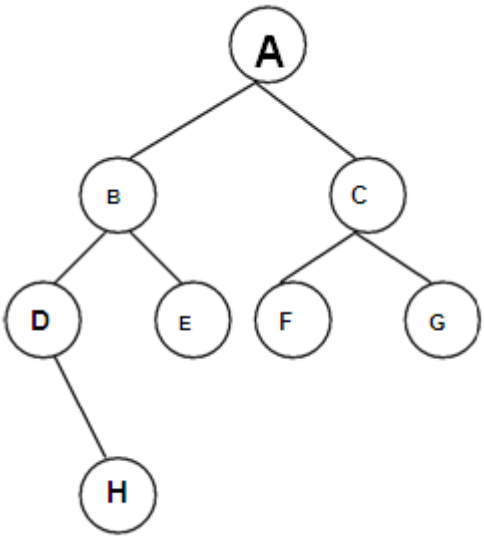
This is the unique node in the tree to which further sub-trees are attached.



Here, A is the root.

3. Define length of the path in a tree with an example.

Length of a path is the number of edges on the path.



The length of the path from A-H is 3.

4.Write the routine for node declaration in trees.

```
typedef struct TreeNode *PtrToNode; struct
TreeNode
    {
        ElementType Element;
        PtrToNode FirstChild;
        PtrToNode NextSibling;
    };
```

5. List the applications of trees.

- Binary search trees
- Expression trees
- Threaded binary trees

6. Define Binary tree.

A Binary tree is a tree in which no node can have more than two children.

7.List the tree traversal applications.

1. Listing a directory in an hierarchal file system (preorder)

2. Calculating the size of a directory (post order)

8. Define binary search tree?

Binary Search tree is a binary tree in which each internal node *x* stores an element such that the element stored in the left sub tree of *x* are less than or equal to *x* and elements stored in the right sub tree of *x* are greater than or equal to *x*. This is called binary-search-tree

9. List the Operations of binary search tree?

- Make Empty
- Find
- Insert
- Delete
- Search
- Display

10.Define Threaded Binary tree.

A Threaded Binary Tree is a binary tree in which every node that does not have a right child has a THREAD (in actual sense, a link) to its INORDER successor. By doing this threading we avoid the recursive method of traversing a Tree, which makes use of stacks and consumes a lot of memory and time.

11.List the uses of binary tree.

1. Searching.
2. Compiler design.

12 .State the properties of a binary tree

- The maximum number of nodes on level n of a binary tree is 2^{n-1}, where n≥1.
- The maximum number of nodes in a binary tree of height n is 2^{n-1}, where n≥1.
- For any non-empty tree, $n_l=n_d+1$ where n_l is the number of leaf nodes and n_d is the number of nodes of degree 2.

13.What is meant by binary tree traversal?

Traversing a binary tree means moving through all the nodes in the binary tree, visiting each node in the tree only once.

14.What are the different binary tree traversal techniques?

- Preorder traversal
- Inorder traversal
- Postorder traversal

15.What are the tasks performed while traversing a binary tree?

- Visiting a node
- Traverse the left sub-tree
- Traverse the right sub-tree

16.What are the tasks performed during preorder traversal?

- Process the root node
- Traverse the left sub-tree
- Traverse the right sub-tree

Routine:

```
void preorder(node *temp)
{
        if(temp!=NULL)
        {
```

```
                printf("%d",temp->data);
                preorder(temp->left);
                preorder(temp->right);
        }
        }
```

17.What are the tasks performed during inorder traversal?

- Traverse the left sub-tree
- Process the root node
- Traverse the right sub-tree

Routine:

```
void inorder(node *temp)
{
        if (temp!=NULL)
        {
                inorder(temp->left);
                printf("%d",temp->data);
                inorder(temp->right);

        }
}
```

18.What are the tasks performed during postorder traversal?

- Traverse the left sub-tree
- Traverse the right sub-tree
- Process the root node

ROUTINE:

```
void postorder(node *temp)
{
        if(temp!=NULL)
        {
        postorder(temp->left);
        postorder(temp->right);
         printf("%d",temp->data);
        }
}
```

19.State the merits of linear representation of binary trees.

- Storage method is easy and can be easily implemented in arrays
- When the location of a parent/child node is known, other one can be determined easily
- It requires static memory allocation so it is easily implemented in all programming language

20.State the demerit of linear representation of binary trees.

Insertions and deletions in a node take an excessive amount of processing time due to data movement up and down the array.

21.State the merit of linked representation of binary trees.

Insertions and deletions in a node involve no data movement except the rearrangement of pointers, hence less processing time.

22.State the demerits of linked representation of binary trees.

- Given a node structure, it is difficult to determine its parent node
- Memory spaces are wasted for storing null pointers for the nodes, which have one or no sub-trees
- It requires dynamic memory allocation, which is not possible in some programming language

23.What do you mean by general trees?

General tree is a tree with nodes having any number of children.

24. Define ancestor and descendant

If there is a path from node n_1 to n_2, then n_1 is the ancestor of n_2 and n_2 is the descendant of n_1.

25. Why it is said that searching a node in a binary search tree is efficient than that of a simple binary tree?

In binary search tree, the nodes are arranged in such a way that the left node is having less data value than root node value and the right nodes are having larger value than that of root. Because of this while searching any node the value of the target node will be compared with the parent node and accordingly either left sub branch or right sub branch will be searched. So, one has to compare only particular branches. Thus searching becomes efficient.

26. What is an expression tree?

An expression tree is a tree which is build from infix or prefix or postfix expression. Generally, in such a tree, the leaves are operands and other nodes are operators.

27. Define right-in threaded tree

Right-in threaded binary tree is defined as one in which threads replace NULL pointers in nodes with empty right sub-trees.

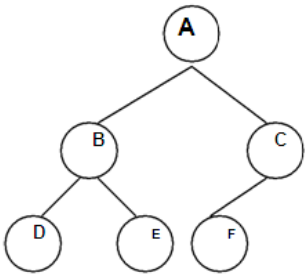
28. Define left-in threaded tree

Left-in threaded binary tree is defined as one in which each NULL pointers is altered to contain a thread to that node's inorder predecessor.

29.Define AVL Tree. Give Example.

An AVL Tree is a binary search tree with a balance condition, which is easy to maintain and ensure that the depth of the tree is O(log N). Balance condition require that the left and the right sub trees have the same height.

Example:



30.Define Balance factor.

The balance factor of a node in binary tree is defined to be h_R- h_L where h_L and h_R are heights of left and right subtrees of T. For any node in AVL tree the balance factor should be 1,0 or -1.

31.When AVL tree property is violated and how to solve it?

After insertion of any node in an AVL tree if the balance factor of any node becomes other than -1,0, or 1 then it is said that AVL property is violated. So the node on the path from the inserted node to the root needs to be readjusted. Check the balance factor for each node in the path from inserted node to the root node and adjust the affected subtree such that the entire subtree should satisfy the AVL property.

32.When AVL tree property is violated and how to solve it?

After insertion of any node in an AVL tree if the balance factor of any node becomes other than -1,0, or 1 then it is said that AVL property is violated. So the node on the path from the inserted node to the root needs to be readjusted. Check the balance factor for

each node in the path from inserted node to the root node and adjust the affected subtree such that the entire subtree should satisfy the AVL property.

33.Mention the four cases to rebalance the AVL tree.

- An insertion of new node into Left subtree of Left child(LL).
- An insertion of new node into Right subtree of Left child(LR).
- An insertion of new node into Left subtree of Right child(RL).
- An insertion of new node into Right subtree of Right child(RR).

34.Define Rotation in AVL tree. Mention the two types of rotations.

Some modifications done on AVL tree in order to rebalance it is called Rotation of AVL tree.

The two types of rotations are

- Single Rotation
  - Left-Left Rotation
  - Right-Right Rotation
- Double Rotation
  - Left-Right Rotation
  - Right-Left Rotation

35.Define Splay Tree.

A splay tree is a self-balancing binary search tree with the additional property that recently accessed elements are quick to access again. It performs basic operations such as insertion, look-up and removal in O (log(n)) amortized time. For many non-uniform sequences of operations, splay trees perform better than other search trees, even when the specific pattern of the sequence is unknown.

36.List the types of rotations available in Splay tree.

Let us assume that the splay is performed at vertex v, whose parent and grandparent are p and g respectively. Then, the three rotations are named as:

Zig: If p is the root and v is the left child of p, then left-left rotation at p would suffice. This case always terminates the splay as v reaches the root after this rotation.

Zig-Zig: If p is not the root, p is the left child and v is also a left child, then a left-left rotation at g followed by a left-left rotation at p, brings v as an ancestor of g as well as p.

Zig-Zag: If p is not the root, p is the left child and v is a right child, perform a left-right rotation at g and bring v as an ancestor of p as well as g.

37.Define B-tree of order M.

A B-tree of order M is a tree that is not binary with the following structural properties:

- The root is either a leaf or has between 2 and M children.
- All non-leaf nodes (except the root) have between ⌈M/2⌉ and M children.
- All leaves are at the same depth.

38.What do you mean by 2-3 tree?

A B-tree of order 3 is called 2-3 tree. A B-tree of order 3 is a tree that is not binary with the following structural properties:

- The root is either a leaf or has between 2 and 3 children.
- All non-leaf nodes (except the root) have between 2 and 3 children.
- All leaves are at the same depth.

39.What do you mean by 2-3-4 tree?

A B-tree of order 4 is called 2-3-4 tree. A B-tree of order 4 is a tree that is not binary with the following structural properties:

- The root is either a leaf or has between 2 and 4 children.
- All non-leaf nodes (except the root) have between 2 and 4 children.
- All leaves are at the same depth.

40.What are the applications of B-tree?

- Database implementation
- Indexing on non primary key fields
-

41.Define binary heaps.

A binary heap is a heap data structure created using a binary tree. It can be seen as a binary tree with two additional constraints:

- the tree is an ***almost complete binary tree***; that is, all levels of the tree, except possibly the last one (deepest) are fully filled, and, if the last level of the tree is not complete, the nodes of that level are filled from left to right.
- The ***heap property***: each node is greater than or equal to each of its children according to some comparison predicate which is fixed for the entire data structure.

42.What are the applications of priority queues?

- The selection problem
- Event simulation

43.What do you mean by the term "Percolate up"?

To insert an element, we have to create a hole in the next available heap location. Inserting an element in the hole would sometimes violate the heap order property, so we

have to slide down the parent into the hole. This strategy is continued until the correct location for the new element is found. This general strategy is known as a percolate up; the new element is percolated up the heap until the correct location is found.

44.What do you mean by the term "Percolate down"?

When the minimum element is removed, a hole is created at the root. Since the heap now becomes one smaller, it follows that the last element X in the heap must move somewhere in the heap. If X can be placed in the hole, then we are done.. This is unlikely, so we slide the smaller of the hole's children into the hole, thus pushing the hole down one level. We repeat this step until X can be placed in the hole. Thus, our action is to place X in its correct spot along a path from the root containing minimum children. This general strategy is known as percolate down

45.List the Applications of Binaryheap

- Heap sort
- Selection Algorithm
- Graph Algorithm

46. What are red black trees?

A **red–black tree** is a binary search tree in which every node is colored with either red or black.It is a type of self-balancing binary search tree

47. Explain the properties of Red-black trees.

1. Every node is either red or black.
2. The root and leaves (NIL 's) are black.
3. If a node is red, then its parent is black.
4. All simple paths from any node x to a descendant leaf have the same number of black nodes = black-height(x)

48. What are binomial heaps.

A binomial heap is implemented as a collection of binomial trees (compare with a binary heap, which has a shape of a single binary tree). A **binomial tree** is defined recursively:

- A binomial tree of order 0 is a single node

49. What are the advantages of Fibonacci heaps?

1. The use of Fibonacci heap improves the asymptotic running time of Dijkstra's algorithm of computing the shortest path.

2. In prime's algorithm Fibonacci heap is helpful in finding the minimum spanning tree.

50. What is amortized analysis?

An amortized analysis means finding average running time per operation over a worst case sequence of operations. An amortized analysis indicates that average cost of a single operation is small if average of sequence of operation is obtained.

51. **Name the techniques used in amortized analysis.**

1. Aggregate analysis
2. Accounting method
3. Potential method

PART-B

1. Define AVL Trees. Explain its rotation operations with example**.(APRIL/MAY 2010)**
2. Explain heap structures. How are binary heaps implemented? Give its algorithm with example.**(APRIL/MAY 2010,NOV/DEC 2007)**
3. Write a function to perform deletion of an element from a binary heap.**(NOV/DEC 2007)**
4. Show the result of inserting 2, l, 4, 5, 9, 3, 6, and 7 into an empty AVL tree. **(NOV/DEC 2007)**
5. Explain about rotations of AVL tree.**(NOV/DEC 2008)**
6. Write function to delete the minimum element from a binary heap.**(MAY/JUNE 2007)**
7. Write ADT operations for heap sort. Using the above algorithm, sort the following 34,45,25,11,6,85,17,35.**(MAY/JUNE 2007)**
8. Explain in detail the B-tree. What are its advantages? **(NOV/DEC 2012)**
9. Explain binary heaps in detail. Give its merits.**(NOV/DEC 2012)**

UNIT-V

1. Define Graph.

A graph G consist of a nonempty set V which is a set of nodes of the graph, a set E which is the set of edges of the graph, and a mapping from the set for edge E to a set of pairs of elements of V. It can also be represented as G=(V, E).

2. Define adjacent nodes.

Any two nodes which are connected by an edge in a graph are called adjacent nodes. For example, if an edge x ε E is associated with a pair of nodes (u,v) where u, v ε V, then we say that the edge x connects the nodes u and v.

3. What is a directed graph?

A graph in which every edge is directed is called a directed graph.

4. What is an undirected graph?

A graph in which every edge is undirected is called a directed graph.

5. What is a loop?

An edge of a graph which connects to itself is called a loop or sling.

6. What is a simple graph?

A simple graph is a graph, which has not more than one edge between a pair of nodes than such a graph is called a simple graph.

7. What is a weighted graph?

A graph in which weights are assigned to every edge is called a weighted graph.

8. Define outdegree of a graph?

In a directed graph, for any node v, the number of edges which have v as their initial node is called the out degree of the node v.

9. Define indegree of a graph?

In a directed graph, for any node v, the number of edges which have v as their terminal node is called the indegree of the node v.

10. Define path in a graph?

The path in a graph is the route taken to reach terminal node from a starting node.

11. What is a simple path?

A path in a diagram in which the edges are distinct is called a simple path. It is also called as edge simple.

12. What is meant by strongly connected in a 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 strongly connected.

13. When is a graph said to be weakly connected?

When a directed graph is not strongly connected but the underlying graph is connected, then the graph is said to be weakly connected.

14. Name the different ways of representing a graph?

a. Adjacency matrix

b. Adjacency list

15. What is an undirected acyclic graph?

When every edge in an acyclic graph is undirected, it is called an undirected acyclic graph. It is also called as undirected forest.

16. What are the two traversal strategies used in traversing a graph?

a. Breadth first search

b. Depth first search

17. What is a minimum spanning tree?

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

18. Name two algorithms two find minimum spanning tree

Kruskal's algorithm

Prim's algorithm

19. Define graph traversals.

Traversing a graph is an efficient way to visit each vertex and edge exactly once.

20. List the two important key points of depth first search.

i) If path exists from one node to another node, walk across the edge – exploring the edge.

ii) If path does not exist from one specific node to any other node, return to the previous node where we have been before – backtracking.

21. What do you mean by breadth first search (BFS)?

BFS performs simultaneous explorations starting from a common point and spreading out independently.

22. Differentiate BFS and DFS.

No.	DFS	BFS
1	Backtracking is possible from a dead end	Backtracking is not possible
2	Vertices from which exploration is incomplete are processed in a LIFO order	The vertices to be explored are organized as a FIFO queue

3	Search is done in one particular direction	The vertices in the same level are maintained parallely

23. What do you mean by tree edge?

If w is undiscovered at the time vw is explored, then vw is called a tree edge and v becomes the parent of w.

24. What do you mean by back edge?

If w is the ancestor of v, then vw is called a back edge.

25. Define biconnectivity.

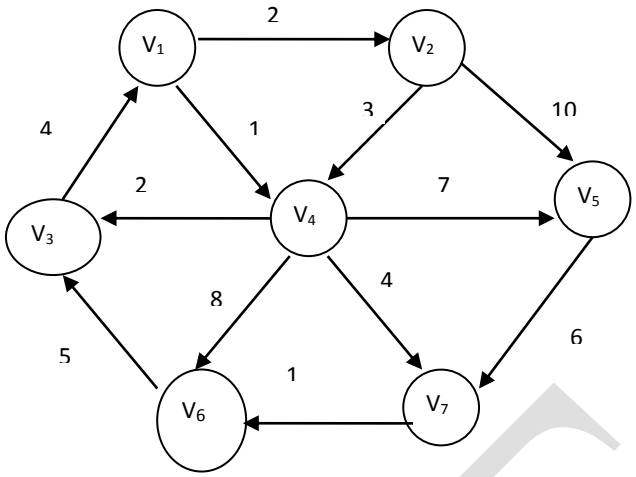
A connected graph G is said to be biconnected, if it remains connected after removal of any one vertex and the edges that are incident upon that vertex. A connected graph is biconnected, if it has no articulation points.

26. What do you mean by articulation point?

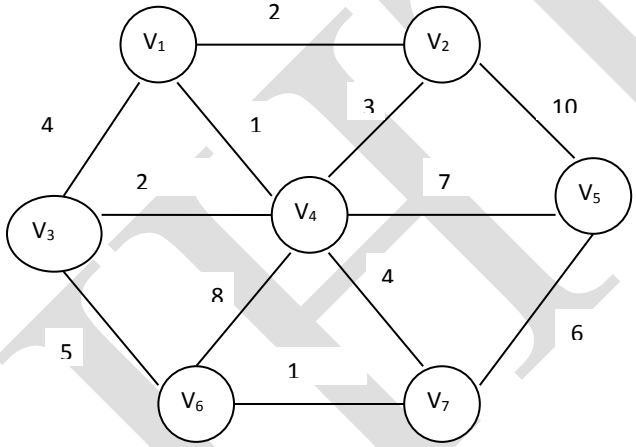
If a graph is not biconnected, the vertices whose removal would disconnect the graph are known as articulation points.

PART-B

1. Explain the Prim's algorithm to find minimum spanning tree for a graph.**(APRIL/MAY 2004,NOV/DEC 2005,MAY/JUNE 2007,APRIL/MAY 2008)**
2. What are strongly connected components? Explain.**(NOV/DEC 2005)**
3. Write short notes on biconnectivity.**(MAY/JUNE 2007)**
4. Explain BFS and DFS in detail. Write the algorithm**.(NOV/DEC 2004)**
5. Explain topological sorting algorithm.**(NOV/Dec 2005,MAY/JUNE 2006)**
6. Write ADT routines for DFS algorithm.**(NOV/Dec 2006)**
7. Formulate an algorithm to find the shortest path using Dijkstra's algorithm.**(MAY/JUNE 2007,APRIL/MAY 2008)**
8. Write an algorithm to find the minimum cost spanning tree of an weighted undirected graph.**(NOV/DEC 2007)**
9. What is single source shortest path problem? Discuss Dijikstra's single source shortest path algorithm with an example.**(NOV/DEC 2012,NOV/DEC 2007)**
10. Explain Dijkstra's algorithm using the following graph' Find the shortest path between v1 to v2, v3, v4, v6, v7**(MAY/JUNE 2007,NOV/DEC 2008)**



11. Construct minimum spanning tree for the graph shown below.**(MAY/JUNE 2007)**



12. (i) What is a strongly connected graph? Give an example. (4)

    (ii) Write the algorithm to compute lengths of shortest path. (4)

    (iii) Explain the depth first search algorithm. (8) **(NOV/DEC 2009)**

13. Explain in detail the Dijkstra's algorithm to solve the shortest path problem. **(NOV/DEC2010)**
14. Discuss in detail the applications of graphs. **(NOV/DEC2010)**
15. (a)(i)Explain Dijkstra's algorithm and solve the single source shortest path problem with an example. (12)

    (ii) Illustrate with an example, the linked list representation of graph.(4) **(APRIL/MAY 2010)**

16. (i) Write the procedures to perform the BFS and DFS search of a graph. (8)

    (ii) Explain Prim's algorithm to construct a minimum spanning tree from an undirected graph. (8) **(APRIL/MAY 2010)**

17. Find MST for the following graph.

(NOV/DEC2012)