VALLIAMMAI ENGINEERING COLLEGE

SRM Nagar, Kattankulathur-603 203

DEPARTMENT OF INFORMATION TECHNOLOGY

CS6301 PROGRAMMING AND DATASTRUCTURES II

YEAR / SEMESTER: II / III

ACADEMIC YEAR: 2014-15 (ODD SEMESTER)

QUESTION BANK

<u>UNIT I</u>

OBJECT ORIENTED PROGRAMMING FUNDAMENTALS

PART-A (2 MARKS)

1. Define class and object.
2. Define object.
3. When do we declare a member of a class static?
4. How is a class declared in C++?
5. What is a scope resolution operator and how can it be used for global variable?
6. What is meant by binding?
7. How the pointer is implemented in C++?
8. What does ‘this’ pointer point to?
9. Define encapsulation.
10. Write the properties of static member function.
11. What is an inline function?
12. State the characteristics of procedure oriented programming.
13. What are the features of Object Oriented Programming?
14. Distinguish between Procedure Oriented Programming and Object Oriented Programming.
15. List out the basic concepts of Object Oriented Programming.
16. State Polymorphism.
17. List and define the two types of Polymorphism.
18. Define Message Passing.
19. List out some of the benefits of OOP.
20. List out the applications of OOP.
21. List out the four basic sections in a typical C++ program.
22. State the use of void in C++.
23. List out the conditions where inline expansion doesn’t work.
24. Why do we use default arguments?
25. State the advantages of default arguments.
26. Define friend function.

PART B (16 MARKS)

1. Write short notes on comparison of conventional programming and OOPS.
2. Explain the control structures of C++ with suitable examples.
3. Write a menu driven program to accept 2 integers and an operator (+,-,*,%,/) and to perform the operation and print the result.
4. Give the syntax and usage of the reserved word inline with two examples.
5. Explain the importance of constructors and destructors with example.
6. Define friend class and specify its importance. Explain with suitable example.
7. Explain the merits and demerits of object oriented paradigm.
8. What are the difference between pointer to constants and constant to pointers?
9. 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.
10. Write a program to demonstrate how a static data is accessed by a static member function.

UNIT II

OBJECT ORIENTED PROGRAMMING CONCEPTS

PART-A (2 MARKS)

1. Why is it necessary to overload an operator?
2. What is the need to declare base classes as virtual?
3. What is the use of virtual functions in C++?
4. What is inheritance?
5. What does multiple inheritance mean?
6. List out the operators which cannot be overloaded.
7. What is constructor?
8. Define default constructor.
9. Define parameterized constructor.
10. Define copy constructor.
11. Define dynamic constructor.
12. Define const object.
13. Define destructor.

14. Write some special characteristics of constructor.
15. List some of the rules for operator overloading.
16. What are the types of type conversions?
17. What is an abstract class?
18. What is the ambiguity between default constructor and default argument constructor?
19. What is meant by casting operator and write the general form of overloaded casting operator.
20. Is it possible to overload a constructor? How**.**

PART B (16 MARKS)

1. Define a class ‘string’. Use overload ‘= =’ operator to compare two strings.
2. What is a parameterized constructor? Explain with example.
3. Describe the syntax of multiple inheritance. When do we use such an inheritance?
4. What is a virtual function? When do we make a virtual function ‘‘pure’’?
5. What is operator overloading? Overload the numerical operators ‘+’ and ‘/’ for complex numbers “addition” and “division” respectively.
6. Define friend class and specify its importance. Explain with suitable example.
7. Explain the concept of inheritance by considering an example of “vehicle”.
8. Explain the operators used for dynamic memory allocation with examples.
9. Write a C++ program to define overloaded constructor and to perform string initialization and string copy.
10. Illustrate the use of copy constructor and function overloading with C++ program.
11. What are the different forms of inheritance supported by C++? Explain with relevant example code.
12. Explain protected data with private and public inheritance.
13. Write a C++ program for to solve eight queens problem with friend functions.

14. Write a C++ program that contains a class String and overloads the following operators on Strings.
    + to concatenate two strings
    - To delete a substring from the given string
    = = to check for the equivalence of both strings.

UNIT III
C++ PROGRAMMING ADVANCED FEATURES

PART-A (2 MARKS)

1. What is template?
2. How is an exception handled in C++?
3. What is file mode? List any four file modes.
4. What are the file stream classes used for creating input and output files?
5. List out any four containers supported by Standard Template Library.
6. List five common examples of exceptions.
7. What are the three standard template library adapters?
8. What is ‘throw’()? What is its use?
9. What is meant by abstract class?
10. What are streams? Why they are useful?
11. What is a namespace?
12. What is a manipulator?
13. How are exception classified?
14. What do you mean by synchronous exception?
15. What is asynchronous exception?
16. What is File?
17. What is String?
18. What are the ways for creating String object?
19. What are class templates?
20. What are function templates?

PART B (16 MARKS)

1. Explain with an example, how exception handling is carried out in C++.
2. Write a class template to insert an element into a linked list.

3. Write a class template to implement a stack.
4. What is a user defined exception? Explain with an example.
5. Write a C++ program to store set of objects in a file and to retrieve the same.
6. Highlight the features of STL.
7. List the different stream classes supported in C++
8. Write a C++ program to read the contents of a text file.
9. Explain the use of any six manipulators with example.
10. Discuss in detail the unformatted I/O operations.

UNIT IV
ADVANCED NON-LINEAR DATA STRUCTURES

PART-A (2 MARKS)

1. In an AVL tree, at what condition the balancing is to be done?
2. What do you mean by a heap and mention its types?
3. What is a binary search tree?
4. What is a disjoint set? Define the ADT for a disjoint set.
5. Define non linear data structure.
6. Discuss the application of trees.
7. What is meant by depth and height of a tree?
8. Write code for disjoint set find.
9. Show the result of inserting 2;1;4;5;9;3;6;7 into an initially empty AVL tree.
10. Write any two advantages of binary heap.
11. Why is always a red node inserted into a red-black tree?
12. Distinguish between the constraints of shape property and heap property.
13. What is the purpose of Red black trees?
14. Give an example of Fibonacci heap and define Fibonacci heap.
15. Define binomial heap with appropriate diagram.
16. Does the sequence <23,17,14,6,13,10,1,5,7,12> represents a heap?
17. Is the height of every tree in a Binomial heap that has n elements O (log n)? If not what is the worst-case height as a function of n?
18. Compare the worst case height of a red-black tree with n nodes and that of an AVL tree with the same number of nodes?

19. List out any two applications of splay trees.
20. Define amortized analysis.

PART B (16 MARKS)

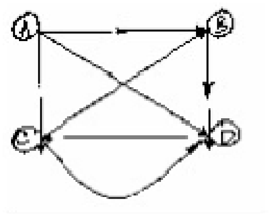
1. Discuss, compare and contrast Binomial heaps and Fibonacci heaps in terms of insertion, deletion operations and applications.
2. Describe any one scheme for implementing Red-Black trees. Explain insertion and deletion algorithm with details. How do these algorithms balance the height of the tree?
3. Write algorithm to construct Fibonacci heap with suitable example.
4. Write algorithm to construct Binomial heap with suitable example.
5. What are the properties of Red black trees?
6. What is the need for splay trees? Give an example.
7. Illustrate with an algorithm to show the insertion of data into an AVL tree.
8. Write an algorithm to merge two AVL trees T1 and T2 to obtain new AVL tree pointed out by NT.
9. Write C++ member function to implement the following Fibonacci Heap operations
   (i) Create an empty F-heap. (8)
   (ii) Insert element x into F-heap. (8)
10. Explain Fibonacci Heap Deletion Key operation using Cascading-Cut procedure with example.
11. Explain insertion procedure in Red-Black tree and insert the following sequence: { 20,10,5,30,40,57,3,2,4,35,25,18,22,21}
12. Show the result of inserting 10,17,2,4,9,6,8 into an empty AVL tree.
13. Write the procedure to implement single and double rotation while inserting nodes in AVL tree.

UNIT V

GRAPHS

PART-A (2 MARKS)

1. Define graph.
2. When does a graph become tree?
3. What is a spanning tree?
4. What is degree of a graph?
5. Define indegree and out degree of a graph.
6. What is a minimum spanning tree?
7. What is Euler circuit?
8. What are the two ways of representing a graph? Give examples.
9. Does either prim's or Kruskal's algorithm work if there are negative weights?
10. List out the applications of graph.
11. Does the minimum spanning tree of a graph give the shortest distance between any two specific nodes? Justify.
12. What is meant by digraph? Define the terms in-degree and out-degree with respect to a digraph.
13. Write the adjacency matrix for the following graph.



14. What is topological sort?
15. Define bi-connected graph.
16. What is meant by biconnectivity and articulation point with respect to undirected graphs?
17. What do you mean by depth first traversal?
18. What is a forest?
19. What do you mean by breadth first traversal?
20. Explain prim's and Kruskal's algorithm.

PART B (16 MARKS)

1. Explain the method of constructing minimum cost spanning tree using Kruskal's algorithm.
2. Explain briefly articulation points and biconnected components.
3. Explain the traversals of directed graphs also give its analysis.
4. Explain Prim's algorithm to construct minimum spanning tree from an undirected graph.
5. What is topological sort? Write an algorithm to perform topological sort.
6. Write the pseudo code to find a minimum spanning tree using Kruskal's algorithm.
7. Find the shortest weighted path from A to all other vertices for the graph given below

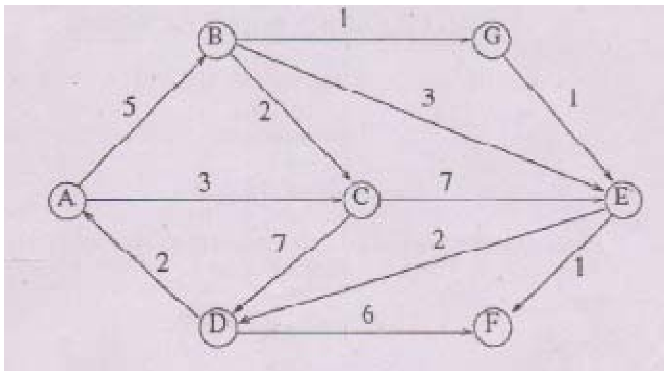
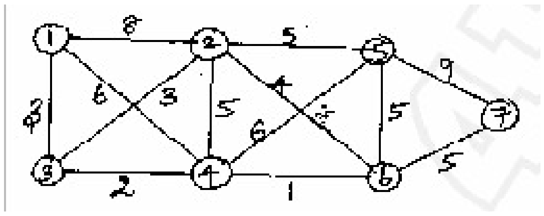


Figure 1

8. Find the shortest unweighted path from B to all other vertices for the graph given in Figure 1.
9. Explain Dijkstra's algorithm and solve the single source shortest path problem with an example.
10. Illustrate with an example, the linked list representation of graph.
11. Find the shortest path from node 1 to 7 using shortest path algorithm.


1
2
3
4
5
6
7
8
5
4
6
3
5
4
6
5
9
2
1
5