PREVIOUS YEAR's QUESTIONS AND REFERENCES

Previous Years: 2017,2016, 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008

Model Papers: -

Always follow prescribe text books and reference books

TB1: Fundamentals of Data Structures in C, Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed

TB2: Data Structures using C in Depth, Deepali Srivastava and Suresh Kumar Srivastava

Material: Supporting Material.

ALL THE BEST!!

PREPARED BY VIKAS B, ASSISTANT PROFESSOR, DEPT. of CSE

MODULE 1

Data representation:

1. What is Data Structures? Discuss about various data structure. **[April,2017]**

   (OR)

   Write about primitive and non-primitive data structures. **[April, 2016]**

   Ans: Module 1: 1.1.1 Introduction - Page No. 1 and 2 - Fig 1.2
2. What is an array? Explain its operations. **[April, 2016]**

   Ans: Module 1: 1.1.3 Array Based Representation & Operations - Page No. 3
3. What is indirect addressing? Explain with a suitable example. **[Nov, 2015] [Nov, 2011] [Nov,2008]**

   Ans: Module 1: 1.1.4 Indirect Addressing – Page No. 8
4. Distinguish between linear and non-linear data structures. **[Nov, 2015] [Nov, 2008]**

   Ans: Module 1: 1.1.1 Introduction - Page No. 1 and 2 – Fig 1.2
5. What is an Array? Explain two and three dimensional arrays with examples. **[Nov, 2015] [Nov, 2011]**

   Ans: Module 1: 1.3.1 Arrays, Matrices- Page No. 13. For 3D arrays try writing same as 2D but the size is 3x3.

Searching:

1. Write a C program for Binary Search. **[April,2017]**

   (OR)

   Write a C program for Binary Search. Find the number and position of 36 in the array given below by using Binary Search 2,6,9,10,13,17,32,35,36,58,76,92. **[April, 2016] [Nov, 2014] [Nov, 2010]**

   Ans: Module 1: 1.2.2 Binary Search –Page No. 11(Use algorithm to solve the above)
2. Derive best and worst case time complexity of a linear search. **[April, 2016]**

   Ans: Module 1: Analysis and Time complexity- Page No. 11,12,13
3. What do you mean by time complexity? Discuss about time complexities for linear and binary search. **[April,2017]**

   (OR)

   Compare the differences between linear search and Binary search. **[Nov, 2014]**

   (OR)

   From the given below which is the Best Searching Algorithm for Static Data and Dynamic Data i) Binary Search ii) Linear Search **[Nov, 2013]**

   (OR)

   Compare the Best, Average and Worst case complexities of Linear Search and Binary Search. **[Nov, 2013]**

   (OR)

   Ans: **Time Complexity**: Every algorithm requires some amount of computer time to execute its instruction to perform the task. This computer time required is called time complexity.

   The time complexity of an algorithm is the total amount of time required by an algorithm to complete its execution.

BASIS FOR COMPARISON	LINEAR SEARCH	BINARY SEARCH
Time Complexity	O(N)	O(log 2 N)
Best case time	First Element O(1)	Center Element O(1)
Prerequisite for an array	No required	Array must be in sorted order
Worst case for N number of elements	N comparisons are required	Can conclude after only log2N comparisons
Can be implemented on	Array and Linked list	Cannot be directly implemented on linked list
Insert operation	Easily inserted at the end of list	Require processing to insert at its proper place to maintain a sorted list.
Algorithm type	Iterative in nature	Divide and conquer in nature
Usefulness	Easy to use and no need for any ordered elements	Somehow tricky algorithm and elements must be arranged in order
Lines of Code	Less	More

Both linear and binary search algorithms can be useful depending on the application. If an array is the data structure and elements are organized in sorted order, then binary search is preferred for fast searching. If linked list is the data structure no matter how the elements are arranged, linear search is preferred due to unavailability of direct implementation of binary search algorithm.

Time Complexities: Module 1: Page no.: 11, 13

4. Write an algorithm for Binary Search? What are the implementation issues of binary search? **[Nov, 2014]**

   Ans: Module 1: 1.2.2 Binary Search-Page No.11

5. Write a C program for searching an element using Linear Search. Explain the implementation issues of Linear Search. **[Nov, 2013] [Nov, 2012] [Nov, 2011]**

   (OR)

   What is sequential searching? Explain with a suitable example. **[Nov, 2009]**

   (OR)

   What are linear lists? Write an algorithm which shows the operations on linear lists. **[Nov, 2010]**

   Ans: Module 1: 1.2.1 Linear Search or Sequential Search-Page No. 10

Arrays:

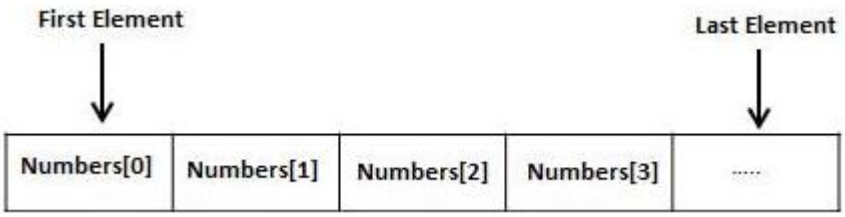
1. Discuss about operations and applications of arrays **[April, 2017]**

   Ans: Operations: http://www.xpode.com/ShowArticle.aspx?ArticleId=142

   Applications: http://www.c4learn.com/c-programming/c-array-application/

2. What is an array? Explain a C program to find largest and smallest of given set using arrays**.** **[April,2017]**

   Ans: Array: Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.




```
#include<stdio.h>
int main()
{
   int a[50],i,n,large,small;
   printf("How many elements:");
   scanf("%d",&n);
   printf("Enter the Array:");
   for(i=0;i<n;++i)
      {scanf("%d",&a[i]);}
   large=small=a[0];
   for(i=1;i<n;++i)
   {
      if(a[i]>large)
         large=a[i];
      if(a[i]<small)
         small=a[i];
   }
   printf("The largest element is %d",large);
   printf("\nThe smallest element is %d",small);
   return 0;
}
```

3. What is an array? Compare and contrast array with lists. **[Nov, 2015] [Nov, 2008]**
   Ans: Module 2: 2.1.1 Differences between Arrays and Linked Lists-Page No.2
4. What is a Sparse Matrix? How the sparse matrix is represented using a Single Linear List? Explain it with an example**. [Nov, 2014] [Nov, 2012] [Nov, 2010] [Nov,2009]**
   Ans: Module 1: 1.3.2 Sparse Matrices – Page No. 14

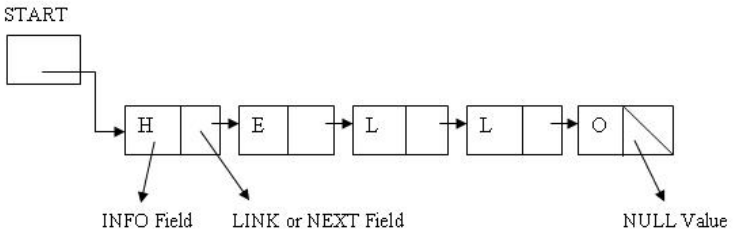
MODULE 2

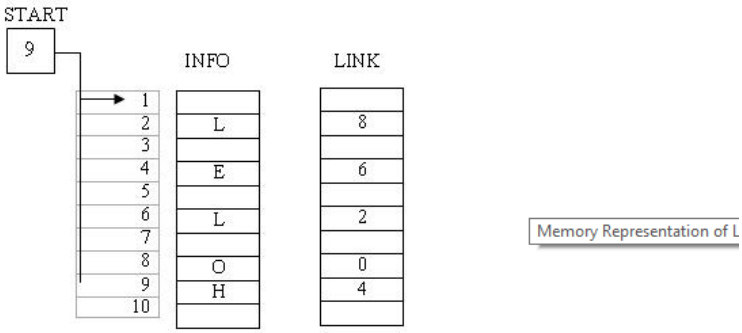
Linked lists:

1. What is single linked list? Explain operations of SLL? **[April, 2017] [April, 2016] [Nov, 2008]**
   Ans: Module 2: 2.2 Single Linked Lists- Page No. 2
2. Explain the operations of Double linked list in detail. **[April,2017] [April, 2016] [Nov, 2009]**
   Ans: Module 2: 2.3 Double Linked Lists-Page No. 12
3. What is circular linked list? Explain operations of a circular linked list with an example program. **[April,2017]**

   (OR)

   Discuss about operations of circular linked list. **[April, 2016]**
   Ans: Module 2: 2.4 Circular Linked List –Page No. 24
4. Explain the memory representation of linked list. **[April, 2016]**
   Let LIST is linear linked list. It needs two linear arrays for memory representation. Let these linear arrays are INFO and LINK. INFO[K] contains the information part and LINK[K] contains the next pointer field of node K. A variable START is used to store the location of the beginning of the LIST and NULL is used as next pointer sentinel which indicates the end of LIST. It is shown below:





Here
START = 9 => INFO[9] = H is the first character.
LINK[9] = 4 => INFO[4] = E is the second character.
LINK[4] = 6 => INFO[6] = L is the third character.
LINK[6] = 2 => INFO[2] = L is the fourth character.
LINK[2] = 8 => INFO[8] = O is the fifth character.
LINK[8] = 0 => The NULL value, so the LIST ends here.

5. Write a C program for searching a circular linked list that has a header node. **[Nov, 2014]**
   https://www.w3resource.com/c-programming-exercises/linked_list/c-linked_list-exercise-29.php
6. Write a C program to join two Doubly Linked List in to a single Doubly Linked List. **[Nov,2013]**

https://www.tutorialspoint.com/learn_c_by_examples/combine_two_doubly_linked_list.htm

7. What is a linked list? What are the different types of linked lists? Explain circular linked list with all the operations that can be performed on it. **[Nov, 2012] [Nov, 2010]**
   Ans: Module 2: Page No. 1
   Module 2: 2.4 Circular Linked List –Page No. 24
8. Write all the operations on single, double and circular linked lists in an algorithm. **[Nov, 2011]**
   Ans: Module 2: Page NO. 2,14,25 (Only list and define with explanation, do not write programs)

MODULE 3

Stacks:

1. Explain array and linked representation of stacks. **[April, 2017] [April, 2016] [Nov, 2008] [Nov, 2010] [Nov, 2011]**
   Ans: Module 3: 3.1.1.1 Operations on Stacks Using Arrays-Page No. 2
   Module 3: 3.1.1.2 Operations on Stacks Using Linked Lists-Page No. 5
2. Explain operations of stacks in detail. **[April, 2016]**
   (OR)
   Write a C program that illustrates Push, Pop and Delete operations in Stack. **[Nov, 2013]**
   Ans: Module 3: 3.1.1.1 Operations on Stacks Using Arrays-Page No. 2
3. What is Stack ADT? Write the applications of Stack. **[Nov, 2015] [Nov 2011] [Nov, 2010] [Nov, 2009]**
   Ans: Module 3: Description and Fig 3.1 and Fig 3.2 - Page No.1 and 2
   Module 3: 3.1.2 Applications of Stack-Page No. 8
4. Discuss about the application of parenthesis matching. **[Nov, 2008]**
   In computer science, stack data structure serves a variety of uses, from operating system function pointer management to compiler construction. We will try to be less ambitious and use the stack to solve the parenthesis matching problem. Instead of using the stack data structure explicitly, we will implicitly use stack operations to show you how to use the stack constructs within the algorithm.
   A typical arithmetic expression is usually written as follows:

   (8 + 3) * (6 + 9) / (1 – 1)

   Here, the parentheses are being used to provide the order in which the operators will be applied in the statement. Also, in various programming languages, we use different types of brackets to represent scope and an incomplete set of brackets will raise a compiler warning.
   For the languages which use them, balancing the sets of opening and closing parenthesis is crucial to manage the scope of variables and methods, as well as explaining the execution context for statements and functions. A matching and balanced instance of parenthesis looks like follows:

   (( )( )( )( ))
   (( )((( ))( )))

   And unbalanced instance will appear like this:

   ((((((( ))

Queues:

1. What is queue? Write a C program on queues with linked list **[April, 2017]**
   Ans: Module 3: Queue Implementation Using LINKED LIST & it's operations – Page No. 21
2. Explain operations of Queue in detail. **[April, 2017] [April, 2016]**

Ans: Module 3: 3.2.1 Operations on Queues Using Arrays- Page No. 16

3. What is a Queue? Explain its applications. **[April, 2016] [Nov, 2015] [Nov, 2011] [Nov, 2010]**
   Ans: http://jcsites.juniata.edu/faculty/kruse/cs240/queues.htm
4. Write algorithms for array and linked representation of Queue operations. **[Nov, 2015] [Nov, 2012] [Nov, 2011]**
   Ans: Module 3: 3.2.1 Operations on Queues Using Arrays- Page No. 16
   Module 3: 3.2.1.2 Operations on Queues using Linked Lists – Page No. 19
5. Write a C program for popping an element from a linked queue **[Nov, 2014]**
   Ans: Module 3: 3.2.1.2 Operations on Queues using Linked Lists – Page No. 20(Deleting only)
6. Discuss any one application of queue and write its implementation **[Nov, 2014]**
   Ans: Ans: http://jcsites.juniata.edu/faculty/kruse/cs240/queues.htm
7. Write the array and linked lists representation of Queues. **[Nov, 2010]**
   Ans: Module 3: 3.2.1 Operations on Queues Using Arrays- Page No. 16
   Module 3: 3.2.1.2 Operations on Queues using Linked Lists – Page No. 19
8. Briefly explain the ADT queue. **[Nov, 2008]**
   Ans: Module 3: Description and Fig 3. - Page No. 15
9. Distinguish between stacks and queues. **[Nov, 2012]**
   Ans: Page No. 22

MODULE 4

Graphs:

1. Write about spanning trees in detail.] **[April, 2017] [April, 2016]**

   (OR)

   Explain the concept of minimum spanning tree with the help of an illustrative example. **[Nov, 2012]**
2. **Ans: Module 4: 4.1.5 Spanning Trees- Page No. – Page no 16**What is a graph? Explain graph traversals in detail. **[April, 2016] [Nov, 2012]**

   (OR)

   Write algorithms for DFS and BFS algorithms and explain with examples. **[Nov, 2008]**

   Ans: Module 4: 4.1.3 Graph Traversals-Page No. 6
3. What are the different types of representation of graphs? Explain each with examples. **[Nov, 2012] [Nov, 2011] [Nov, 2010] [Nov, 2009]**

   Ans: Module 4: 4.1.2 Representation of Graphs – Page No. 4

4. Write Kruskal's Algorithm to search an element in a Graph. **[Nov, 2011]**

   Ans: Module 4: Kruskal's Algorithm- Page No. 17
5. Write Prim's Algorithm to search an element in a Graph. **[Nov, 2010]**

   Ans: Ans: Module 4: Prim's Algorithm- Page No. 21

Introduction to Sorting:

1. Write a C program for merge sort algorithm with an example. Discuss about time and space complexities. **[April, 2017]**

   (OR)

   Write a program for Merge sort with an example. Show that the time complexity of a Merge sort of n elements is O (nlogn). **[Nov, 2011]**

   Ans: Space Complexity: Merge sort on an array has space complexity of O(n), while merge sort on a linked list has space complexity of O(log(n))

   Module 4: 4.2.4 Merge Sort – Page No. 33
2. Explain bubble sort algorithm with example **[April, 2017]**

   (OR)

   Write a C program for bubble sort algorithm. **[April, 2016]**

   Ans: Module 4: C program for Bubble Sort - Page No. 32
3. Explain selection sort algorithm with example. **[April, 2016] [Nov, 2012] [Nov, 2008]**

   Ans: Module 4: 4.2.2 Selection Sort – Page No. 27
4. Discuss the complexity analysis of various sorting algorithms. **[Nov, 2015] [Nov, 2009]**

   Ans: Module 4: 4.2.6 Comparison of various sorting and search techniques – page No.37
5. Write a program for Quick Sort with an example. Show that the average case time complexity of Quick sort is O (nlogn). **[Nov, 2015] [Nov, 2010]**

   Ans: Module 4: 4.2.5 Quick Sort – Page No. 35
6. Compare the Best, Average and Worst case time complexity of the following

   i) Bubble Sort ii) Selection sort iii) Heap Sort

iv) Merge sort v) Insertion sort vi) Quick sort **[Nov, 2015] [Nov, 2013] [Nov, 2008]**

Ans: Module 4: 4.2.6 Comparison of various sorting and search techniques – page No.37

7. Write a C program to implement Quick Sort and sort the following numbers by using Quick sort 46, 56, 47, 11 76, 69, 22, 27, 26. **[Nov, 2015]**

   Ans: Module 4: 4.2.5 Quick Sort – Page No. 35- Sort the above list by looking at the example explained in the material or any video available in the internet.
8. Sort the following sequence by using Merge Sort 88, 74, 98,54,67,32,34,56,90. **[Nov, 2014]**

   Ans: Module 4: 4.2.4 Merge Sort – Page No. 33- Sort the above list by looking at the example explained in the material or any video available in the internet.
9. Sort the following sequence by using bubble sort 50, 36, 11, 9, 55, 24, 27, 57, 22. **[Nov, 2013**

   Ans: Module 4: 4.2.3 Bubble Sort– Page No. 30- Sort the above list by looking at the example explained in the material or any video available in the internet.
10. Write a C program to implement Selection Sort and sort the following numbers by using selection sort 56, 36, 47, 12, 66, 69, 24, 27. **[Nov, 2013]**

    Ans: Module 4: 4.2.2 Selection Sort – Page No. 27 - Sort the above list by looking at the example explained in the material or any video available in the internet.
11. Mention the time and space complexity of selection sort. **[Nov, 2012]**

    Ans: Module 4: Time Complexity - Page No. 29

    Space Complexity: O (1)

MODULE 5

Trees:

1. Write a C program on heap sort with an example**. [April, 2017]**
   Ans: Module 5: C program for heap sort- Page No. 32
2. Write a short note on AVL trees with an example. **[April, 2017]**
   Ans: Module 5: 5.4 AVL Trees: Write an overview of rotations and operations with an example.- Page No. 23
3. Discuss about binary search tree operations. **[April,2017] [April, 2016] [Nov, 2008]**
   Ans: Module 5: 5.3.1 Operations on a Binary Search Tree- Page No. 19
4. What is binary tree? Explain binary tree traversals in detail. **[April, 2016] [Nov, 2012] [Nov, 2011] [Nov, 2009]**

   (OR)

   State and explain all binary tree traversals. **[Nov, 2015] [Nov, 2013] [Nov, 2010]**
   Ans: Module 5: 5.2.3 Operations/Traversals of Binary Trees-Page No. 13
5. What are the properties to be possessed for a binary tree? **[Nov, 2013] [Nov, 2008]**
   Ans: Module 5: 5.2.1 Properties of Binary Tree-Page No. 11
6. Write about heap tree with neat illustration. **[April, 2016]**
   Ans: Module 5: 5.5 Heap Sort- Example –Page No. 29
7. Discuss briefly about the representation of binary trees. **[Nov, 2015] [Nov, 2010] [Nov, 2009]**
   Ans: Module 5:5.2.2 Representations of Binary Trees- Page No. 12
8. Write algorithms for any two Binary tree traversal techniques. **[Nov, 2013]**
   Ans: Module 5: 5.2.3 Operations/Traversals of Binary Trees-Page No. 13
9. Define and explain the following with an example
   a) AVL Tree b) Binary Trees **[Nov, 2015] [Nov, 2012]**
   Ans: Module 5: 5.4 AVL Trees: Write an overview of rotations and operations with an example. - Page No. 23
   Module 5: 5.2.3 Operations/Traversals of Binary Trees-Page No. 13
10. Write the Pseudo code for Abstract data type of binary tree. Write an algorithm to determine the height of a binary tree. **[Nov, 2014]**
    Ans: ADT of Binary Tree

    ADT Binary_Tree (abbreviated BinTree) is
    objects: a finite set of nodes either empty or consisting of a root node, left Binary_Tree, and right Binary_Tree.
    functions:
    for all bt, bt1, bt2 belongs to BinTree, item belongs to element
    Bintree Create ()::= creates an empty binary tree
    Boolean IsEmpty (bt)::= if (bt==empty binary tree) return TRUE else return FALSE
    BinTree MakeBT (bt1, item, bt2)::= return a binary tree whose left subtree is bt1, whose right
    subtree is bt2, and whose root node contains the data item
    Bintree Lchild (bt)::= if (IsEmpty(bt)) return error else return the left subtree of bt
    element Data (bt)::= if (IsEmpty(bt)) return error else return the data in the root node of bt

Bintree Rchild (bt)::= if (IsEmpty(bt)) return error else return the right subtree of bt

Height of a Binary Tree

```
int height( BinaryTree Node t)
{
        if t is a null tree return -1;
        hl = height( left subtree of t);
        hr = height( right subtree of t);
        h = 1 + maximum of hl and hr;
        return h;
}
```

11. What are the steps to be followed to insert an element in to AVL tree? Explain it with an example. **[Nov, 2014] [Nov, 2008]**
    Ans: Module 5: 2. Insertion Operation in AVL Tree-Page No. 26
12. Explain AVL tree rotations. **[Nov, 2011]**
    Ans: Module 5: 5.4.1 AVL Tree Rotations- Page No. 24
13. What is a Balanced Tree? List out and write the algorithms for rotations in AVL Trees. [Nov, 2010]
    Ans: **Balanced Tree**: AVL tree is a self-balanced binary search tree. That means, an AVL tree is also a binary search tree but it is a balanced tree.
    A binary tree is said to be **balanced**, if the difference between the heights of left and right subtrees of every node in the tree is either -1, 0 or +1. In other words, a binary tree is said to be balanced if for every node, height of its children differ by at most one.
    An **AVL tree** is a balanced binary search tree. In an AVL tree, balance factor of every node is either -1, 0 or +1.
    Module 5: 5.4.1 AVL Tree Rotations- Page No. 24
14. Enumerate the operations on AVL trees. **[Nov, 2009]**
    Ans: Module 5: 5.4.2 Operations on an AVL Tree- Page No. 26