Data Structures Using C

QUESTION BANK

REVIEW OF STRUCTURES AND POINTERS, INTRODUCTION TO SPECIAL FEATURES OF C

OBJECTIVE:

- Learn :
- Usage of structures, unions - a conventional tool for handling a group of logically related items.
- Unions - sharing of memory.
- Pointers - the real power of C.

1.	What are structures? Explain how the structures are useful by giving various examples.	
2.	If you have to solve the searching problem for a list of n numbers, how can you take advantage of the fact that the list is known to be sorted? Give answer for lists represented as arrays.	
3.	Bring out the differences between structures and unions.	
4.	What is a pointer and why is it used?	
5.	With an example explain how bit fields can be defined in C.	
6.	What are enumerated data types? Explain with an example	
8.	What is the difference between macros and inline functions?	
9.	Explain the differences between malloc() and calloc() functions used for memory allocation	
10.	What is type casting? Explain with an example.	
11.	Write a short note on Enumeration.	
12.	What are the storage classes available in C? Explain .	
13.	What is union? How is it different from structure? With suitable example show how union is declared and used in C?	
14.	What is a Macro definition? Write a C program to print the cube of a given number using a Macro definition.	
15.	What do you understand by dynamic memory allocation? Explain any three functions that support dynamic allocation?	
16.	What is the difference between macros and inline functions?	
17.	Show the output of the following assuming a = 4, b = 3, c = 10 a) d = a & b b) d = a/b c) d = a^b d) d = ~c e) d = a >> 2 f) d = a << 2	
18.	What are files? Create a file called sem.txt to store semester performances of all branches. Also display the semester and the branch, which has performed the best.	
19.	Create a file for storing employee database. To this file include operations to add new employees, delete. Employees and search for a particular employee.	
20.	Explain the various access modes used for files in C.	
21.	Explain call by value and call by reference. Give two examples. Define formal and actual parameters.	
22.	Create a structure pointed by a pointer variable and allocate memory using malloc() and calloc() separately. Execute the code and note your observations.	

THE STACK

OBJECTIVE:

- The most important concept in data structures.
- To study the prominent role in the area programming is stack.
- To expertise into a concrete and valuable tool in problem solving.

32. Define stack. Explain the implementation of various stack operations using an array, using structures, using singly link list and using doubly link list. 4*4
33. Explain how the stack is used in parameter passing.

34. Show the stack after each operation of the following sequence that starts with the empty stack: push(a), push(b), pop, push(c), push(d), pop.
35. Write the prefix and postfix form of the following infix expressions
    a) ((A + B) * C - (D-E)) $(F+G) b) (A+B)*(C+D-E)*F
36. Write a C function to convert a valid arithmetic infix expression into its equivalent postfix expression. Stack implementation is to be assumed
37. Show the detailed concept of the stack to evaluate the following positive expression
    632-5*+1^7+ *
38. Write short notes on dynamic stack representation.
39. What is a stack? How it can be represented in "C" using arrays?
40. Define Stack as a data structure and discuss its applications.
41. Obtain prefix and postfix expression
    a. (A+B)*(C+D)$(A+B)
    b. A+B*C-D/E*H
    c. (A+B^C^D)*(E+F/D)
42. List the various ways of representing the arithmetic expressions by giving suitable examples
43. Show the detailed contents of the stack for given postfix expression to evaluate
    6 2 3 + - 3 8 2 / + * 2 $ 3 +
44. List applications of stacks. Using stack write an algorithm to determine if a given string is palindrome and print suitable message as output.
45. Implement stack using static and dynamic memory allocation. Discuss the relative merits of these two implementations.
46. Write a program to check whether a given expression is a valid postfix expression or not. If valid evaluate a given postfix expression otherwise display a suitable message.
47. Write a program to check whether a given expression is a valid prefix expression or not.
    If valid evaluate a given prefix expression otherwise display a suitable message.
48. Write an algorithm to convert a given infix expression into prefix expression.
49. Write an algorithm for converting infix expression to post-fix expression. Trace the algorithm indicating content of stack for expression (a-b)/(c*d)+e.
50. Show that , how stack operations takes place for the conversion of infix to postfix expression. ((A-(B+C)) *D) $(E+F)
51. Convert the following infix expression into a postfix expression
    a. (((A/(B*C))+(D*E))-(A*C))
    b. A$B*C-D+E/F/(G+H)
52. Write an algorithm to implement a stack of size N using an array. The elements in the stack are to be integers. The operations to be supported are PUSH , POP and DISPLAY. Take into account the exceptions of Stack overflow and underflow.
53. What are STACK? Explain operation performed on STACK. Discuss how the stack structure can be used for Tower of Hanoi problem.

RECURSION

OBJECTIVE:

- Recursion the most powerful and one of the best programming tools of data structures.
- Advantages and disadvantages of the same.

54. What is recursion? Indicate its properties?
55. What is a recursion? Compare the recursive programs with iterative programs.
56. Write a note on Efficiency of Recursion.
57. Write a "C" recursive program to solve the tower of Hanoi problem. Give the trace for 3 disks.
58. Bring out the differences between recursion and iteration.
59. Write a program to find GCD of 2 nos. using iterative and recursive techniques
60. Write a program to find the following using recursion:

i) Factorial of a given number
ii) Fibonacci series up to a given number

61. Write a recursive program to solve the Binary Search problem.
62. Define recursion. Write a recursive program to find the GCD of two numbers.
63. Write recursive c routines
    i) For computing the binomial coefficient of k term of an n degree polynomial
    ii) That accepts a non negative decimal integer as a parameter and writes out Its binary representation.
64. What is recursion? Discuss an example which can be represented both recursively and iteratively.
65. Write recursive function for:
    i) Checking a number is palindrome or not
    ii) Printing a number or a string in reverse order
66. Write a recursive C function to find the n fibonacci number. Explain with an example

QUEUES AND LISTS

OBJECTIVE:

a. QUEUES:

- To make a considerable exhaustive study of yet another data structure QUEUE.
- To study various forms of implementation techniques of the Queues and their different types.
- To study the applications.

b. LISTS:

- To study exhaustive about lists viz. singly linked list, doubly linked list, circular list, circular doubled linked list, their implementation techniques and applications.

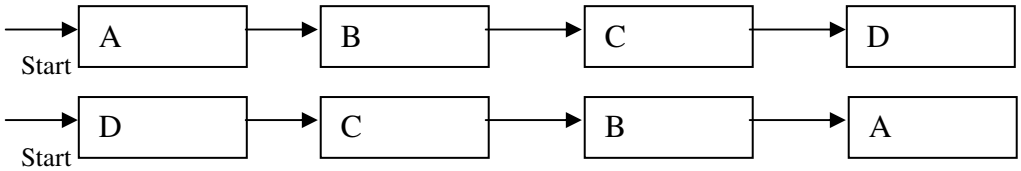
QUEUES

67. Distinguish between Ordinary queues and Circular queues.
68. What are the advantages and disadvantages of linked lists over arrays?
69. How Priority queues can be implemented?
70. Write a program to implement a linear queue using arrays. Take into account the exceptions like Queue Full and Queue Empty.
71. Write a program to implement a circular queue using arrays. Take into account the exceptions like Queue Full and Queue Empty.
72. Write a program to implement a double-ended queue using arrays.
73. A circular queue has a size of 5 and has 3 elements 10,40 and 20,where F=2 and R=4. After inserting 50 and 60,what is value of F and R. Trying to insert 30 at this stage what will happen? Delete 2 elements from the queue and insert 100. Show the sequences of steps with necessary diagrams with the value of F and R.
74. Write a C Program to perform the following operations on a queue
    Insert Delete Display
75. Write a c function
    i) to insert an element at the rear end of a queue
    ii) to delete an element from the front end of the queue

LISTS

76. Understanding list operations like concatenation, searching for a node etc.
77. Understanding the array implementation of list
78. What is a SLL? Write a program to implement stack as a singly linked list.
79. Write a program to create an on ordered link list.
80. Write an algorithm to find whether a given list is ordered or not.

81. Write a recursive search routine to search for a node in a SLL.
82. Write a program to find for a particular node (based either on the position or on info of the node) in a SLL.
83. Write an algorithm to concatenate two lists (assume both are existing)
84. What do you understand by a linked list? Write a C function SEARCH (F, X) that accepts a printer P to a list of integers and an integer X, and returns a pointer to a node containing X, if it exists, and the NULL pointer otherwise.
85. What is a single linked list? Explain with an example how a single linked list can be used for sorting a set of N numbers.
86. What are different types of linked list? Write a C function to count number of elements present in single linked list.
87. Write advantages of doubly linked list over singly linked list. Write C function that will insert a given integer value into an ordered doubly linked list.
88. Write a C function to
    a. To count number of nodes using singly linked list
    b. To concatenate two singly linked list, and then to sort the resultant list
    c. To reverse direction of singly linked list(as shown below)



89. Explain merging of two lists which have been represented as
    i) Array ii) Linked list
90. Write a program to represent a polynomial of single variable using linked list and perform the following functions
    a. Evaluation of polynomial
    b. Display the polynomial

DOUBLY LINKED LIST

91. Define and implement doubly link list.
92. Write a program to find for a particular node (either the position or info of the node can be given) in a DLL or circular DLL.
93. What are the advantages and disadvantages of doubly linked list? Also give its applications.
94. Write a C function to insert and delete a node from the front end in case of doubly linked list.

CIRCULAR LIST

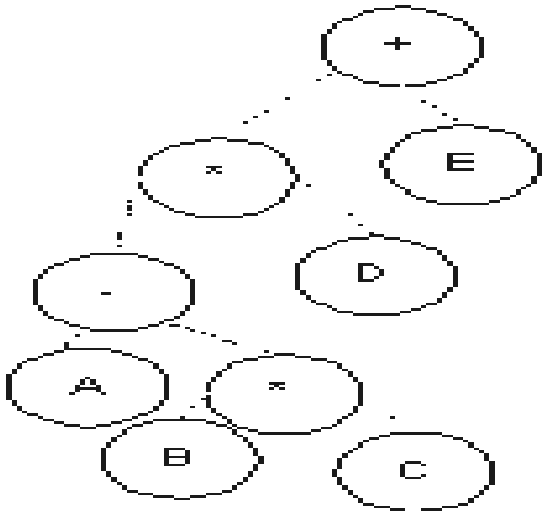
95. Implement a circular SLL with the header node containing the total no of nodes in the list. The node itself consists of student details like reg no, name, marks etc.
96. What are the advantages of circularly linked list over a linear list? Write a C routine that concatenates two circular lists.
97. Write C functions to perform the following operations:
    a. Create a circular singly linked list
    b. Display Circular singly linked list

TREES

OBJECTIVE:

- To study exhaustively regarding a tree, types of trees and focus only on BST (Binary search tree).
- To study the implementation of BST.
- About applications.

98. What is binary tree?
99. What are the differences between strictly binary tree and complete binary tree
100. What are the two conditions under which a binary tree becomes an almost complete binary tree?
101. Discuss the applications of trees
102. Explain with suitable example the following traversals of a tree
i) Preorder ii) In order iii) Post order
103. Give examples to show the father, son, descendent and ascendant nodes of a binary tree.
104. Given the following traversals
In order: E I C F J B G D K H L A
Preorder: A B C E I F J D G H K L
Construct a Binary Tree
105. What is a TREE? Define the following
a. Ancestor
b. Descendants of node with respect to the TREE
106. Explain various type of tree traversal with simple example.
107. Write C functions for following tree traversals
i) Inorder ii) Preorder iii) Post order
108. Write a C recursive program to find out the height of a binary tree.
109. Explain the different methods of representation of binary trees in "C".
110. Write recursive C routines to traverse a binary tree in different ways. Assume the dynamic node representation of a binary tree
111. Define the following
a. Strict binary tree
b. Almost complete binary tree
c. Ordered tree
d. Right in threaded binary tree
112. Construct a binary tree for the expression A + (B-C)*(E+F)/G and draw the diagram showing each step.
113. Explain one-way and two-way threading of binary trees
114. Implement the sequential representation of a binary search tree
115. Write an algorithm to traverse the tree using father field
116. List the various tree traversal techniques giving examples
117. Implement the recursive tree traversal of in, pre and post order traversals
118. Write an algorithm for the iterative tree traversal.
119. What is a threaded binary tree?
120. Implement the in order threaded binary tree.
121. Define the following terms with appropriate sketches
a) Tree b) Directed tree c) Ordered tree d) Binary tree e) Spanning tree
122. Explain the different methods of binary tree representation
123. Write an output after traversing of a given tree by
i) in order
ii) Preorder traversing methods



124. Write an algorithms for deleting node in a binary search tree for all the three cases

125. Write a "C" program to i) Create; and ii) traverse a binary tree in three orders.

126. Write a C Function to create expression TREE using postfix expression, and discuss how a postfix expression can be transformed to its equivalent prefix expression