Jawaharlal Nehru Engineering College

Laboratory Manual

DATA STRUCTURE

For

FYMCA Students

30, Oct 2009 – Rev 00 – Comp Sc – ISO 9000 Tech Document

FORWARD

It is my great pleasure to present this laboratory manual for FYMCA engineering students for the subject of Data Structure. As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that MGM has already been awarded with ISO 9000 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO 9000 Certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Dr. H. H. Shinde
Principal

LABORATORY MANNUAL CONTENTS

This manual is intended for the students FYMCA for the subject of DS (Data Structure). In the subject of this manual typically contains practical/Lab Sessions we have dealt with C language.

Data Structure mainly contains STACK, QUEUE, LINKED LIST, TREE and GRAPH. Programs of data structure should be executed by using C language.

Students are advised to thoroughly go through this manual rather than only topics mentioned are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Dr. S.S.Deshmukh
HOD MCA

Prof. K.K. Misal
Asst. Prof.

List of Assignments

1. Program to demonstrate dynamic memory allocation.
2. Program for implementing STACK.
3. Program for implementing QUEUE.
4. Program for implementing singly linked list.
5. Program for implementing doubly linked list.
6. Program for implementing circular queue using linked list.
7. Program for creation and traversing of binary tree.
8. Programs for depth first search and breadth first search.
9. Programs for insertion sort and merge sort.
10. Programs for heap sort and quick sort.

DOs and DON'T DOs in Laboratory:

1. Do not handle any equipment before reading the instructions/Instruction manuals

2. Read carefully the power ratings of the equipment before it is switched on whether ratings 230 V/50 Hz or 115V/60 Hz. For Indian equipments, the power ratings are normally 230V/50Hz. If you have equipment with 115/60 Hz ratings, do not insert power plug, as our normal supply is 230V/50 Hz, which will damage the equipment.

3. Observe type of sockets of equipment power to avoid mechanical damage

4. Do not forcefully place connectors to avoid the damage

5. Strictly observe the instructions given by the teacher/Lab Instructor

Instruction for Laboratory Teachers:

1. Submission related to whatever lab work has been completed should be done during the next lab session. The immediate arrangements for printouts related to submission on the day of practical assignments.

2. Students should be taught for taking the printouts under the observation of lab teacher.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

WARMUP EXCERCISES:

Programs of recursion.

Programs of Arrays.

Programs of Structures.

Programs of Pointers.

Assignment No. 1

The memory allocations of programs are handling automatically while compiling. But we can dynamically allocate memory for variables at run time. This type of memory allocation is called as Dynamic Memory Allocation.
MALLOC (), CALLOC (), REALLOC () and FREE () are the functions used for memory allocation.

Program no. 1
Input: Enter n number.
Output: .Print values of that numbers.

Algorithm:
Step1: Input n no.
Step2: Allocate memory for the numbers using malloc(() function.
Step3: If prt !=null then assign values to variables.
Step4: Print those values.
Step5: Otherwise print memory allocation failed.
Step6: Free the memory by using free ().
Step7: End.

Program no. 2
Output: Print the values of that number.
Algorithm:
Step1: Start.
Step2: Allocate memory for 3 integer no by using calloc().
Step3: If prt !=null then assign 3 values.
Step4: Again allocate memory for two integer no. using realloc().
Step5: If prt !=null then assign 2 values.
Step6: Print those values.
Step7: Otherwise print not enough memory.
Step8: End.

Assignment no. 2

Stack is an ordered collection of items into which items inserted and items deleted only from one end called top of the stack. Operations on stack is like LIFO i.e. Last in first out.

Algorithm: To add an element onto the stack.

Input: Enter the no. of elements in the stack.
Output: Print the elements in the stack.

Step1: Start.
Step2: check whether stack is empty or not.
Step3: If it is then display message stack is full.
Step4: Otherwise increment top by one.
Step5: Insert an element e. i.e. Stack[top]=e
Step6: end.

Algorithm: To delete an element from the stack.

Input: Enter the delete option from menu.
Output: It will delete top element from the stack and display the elements.

Step1: Start.
Step2: Check whether stack is empty or not.
Step3: If it is then print the message stack is empty.
Step4: Otherwise decrement top by one.
Step5: End.

Assignment no. 3

Queue is an ordered collection of items from which items may be deleted from one end i.e. front and items is inserted at other end called as rear.

Queue follows LIFO technique.

Algorithm: To insert an element into queue.
Input: Enter the elements in the queue.
Output: Print the elements in the queue.

Step1: Initialize front=0,rear=-1 and maxsize=5.
Step2: If rear>=maxsize
Print ”Queue overflow” and return.
Else increment rear by one.
Step3: Insert item, q[rear]=item
Step4: For insert more element goto step2.
Step5: End.

Algorithm: To delete an element from queue.
Input: Enter the delete option from the menu.
Output: It will delete the element from queue and display the elements.

Step1: If front < 0 then print "Queue is empty" and return.
Step2: Otherwise delete an element
i.e. item=q[front]
Step3: Increment queue by one.
Step4: End.

Assignment no. 4
If item in the list itself contains address of the next item then that type of the list is called as linked list. Each item in the list contains information field and next field.

Inserting nodes in the linked list has following three parts.

1. Inserting node at the beginning of the list.
2. Inserting node at the end of the list.
3. Inserting node at the specified position within the list.

Algorithm: Insert a node at the beginning of the liked list.

Input: Enter the elements in the list.
Output: Print the elements in the list.

Step1: Allocate memory for the new node.
Temp=malloc()
Step2: Assign the value to the data field of the new node.
Temp->info
Step3: Make the link field of the new node to point to the starting node of the linked list.
Temp->next=start
Step4: Set the external pointer to point to the new node.
Start = temp

Algorithm: Insert a node at the end.
Step1: If the list is empty then create new node
Temp=malloc()
Step2: If list is not empty then go to the last node and then insert new node after the last node.
r=start
Step3: Go till last node
r->next=temp

Algorithm: Insert new node at the specified position.
(Here assume that new node is to be inserted after node r and before node s)
Step1: Allocate memory for the new node
Temp=malloc()
Step2: Assign value to the data field of the new node
Temp->info
Step3: Go till node r, also mark next node as s.
Step4: Make the next field of the node temp to node s and next field of Node r to point to new node.

Deleting node in the linked list has following three instances.
Deleting the first node.
Deleting the last node.
Deleting middle node.

Input: Enter the no which is to be deleted .
Output: Print the values after deletion.
Algorithm: Delete the first node from linked list.
Input: Enter the element which is to be deleted from the list.
Output: Print the elements after deletion in the list

Step1: Mark the first node as temp.
Step2: Mark start point to the next node.
Step3: Mark next field of temp as null.
Step4: Free temp.

Algorithm: Deleting last node
Step1: Mark the last node as temp and last but one node as r.
Step2: Make the next field of r as null.
Step3: Free temp.

Algorithm: Deleting a middle node.
Step1: Mark the node to be deleted as temp.
Step2: Mark the previous node as r and next node as s.
Step3: Make the next field of r point to s.
Step4: Make the next field of temp as null.
Step5: Free temp.

Assignment no. 5
If node in the list is having two pointers then that type of list is called is called as doubly linked list.

Algorithm: Insert a node in doubly linked list.
Input: Enter the elements in the list.
Output: Print the elements in the list

Step1: Allocate memory for a new node,temp.
Step2: If this is first node
    Start=temp
Step3: If temp is to be inserted after, last node, mark last node as r
  r->next=temp
  temp->prev=r
Step4: If this is to be inserted at the starting of the list
  Start->prev=temp
  Temp->next=start
  Start=temp
Step5: Otherwise if the node is to be inserted after the specific node
  mark that node as 'r' and next node as 's' .

Algorithm: Delete a node from doubly linked list.
Input: Enter the no which is to be deleted .
Output: Print the values after deletion.

Step1: Mark the node to be deleted as temp.
Step2: If this is first node then start point to next node.
  Start-start->next
  Make next field of temp null
  Temp->next=null
Step3: If this is last node, mark previous of last node as 'r'. Make next field of 'r' null and previous field of temp null.

```
r->next=s
s->prev=r
temp->next=null
temp->prev=null
free(temp)
```

Assignment no. 6

If next field in the last node node contains a pointer back to the first node rather than a null pointer then such type of list is called as circular list. Queue as circular list.

Algorithm: Inserting a node in linked list.

Input: Enter the elements in the list.

Output: Display the elements in the list.

Step1: Allocate memory for a new node temp.
Step2: If this is first node,make next point to the node itself.
Temp->next=temp
Also make rear and front point to temp.
Step3: Otherwise rear->next=temp.
Step4: Make rear=temp and next field of rear point to first node.
Rear->next=front

Algorithm: Deleting a node from the liked list.

Input: Enter the delete choice from the list

Output: Display the deleted node from the list.

Step1: Make temp point to front node.
Step2: front=front->next
Step3: temp->next=null
Step4: rear->next=front
Free(temp)

Assignment no. 7

Creation and traversing a binary tree.

There are three types of traversals.

1. Preorder traversal
2. Post order traversal
3. Inorder traversal

Algorithm: Preorder traversal of tree.

Input: Insert the elements in the tree and then enter the choice for preorder, Inorder and postorder from the menu.

Output: Display the elements of tree according to traversal selected from the Menu.

Step1: Visit the node n.
Step2: Traverse the left subtree in preorder.
Step3: Traverse the right subtree in preorder.

Algorithm: Postorder traversal of tree.

Step1: Traverse the left subtree in postorder.
Step2: Traverse the right subtree in postorder.
Step3: Visit the node n.

Algorithm: Inorder traversal of tree.

Step1: Traverse the left subtree in inorder.
Step2: Visit root node n.
Step3: Traverse the right subtree in inorder.

Assignment no. 8
Study of DFS and BFS algorithms
Input: Enter the no of nodes in the graph then enter 1 if there is edge from one node to another otherwise 0.
Output: Display the visited nodes from the graph.
Algorithm: Depth first search (DFS) of a graph.

Step1: Select any unvisited node in the graph. Mark this node as visited, Push this on to the stack.
Step2: Find the adjacent node to the node on top of stack and which is not yet visited, mark this node as visited and push onto stack.
Step3: Repeat step2 untill no new adjacent node to top of stack node can be found, when no new adjacent node can be found, pop top of stack.
Step4: Repeat step2 and step3 till stack becomes empty.
Step5: Repeat step1 till any more nodes which are still unvisited. Use adjacency matrix for finding adjacent nodes.

Algorithm: Breadth first search of the graph.
Input: Enter the no of nodes in the graph then enter 1 if there is edge from One node to another otherwise 0.
Output: Display the adjacency matrix and visited nodes of the graph.

Step1: Select any unvisited node in the graph. Mark this node as visited. Insert this node into the queue.
Step2: Find all the adjacent nodes or node present in the front end of the Queue and which is not yet visited, mark this new node as visited and insert it into the queue.
Step3: Repeat step2 till queue becomes empty.
Step4: Repeat step1 till any more nodes which are still unvisited.

Assignment no. 9

Insertion sort: Insertion sort algorithm sorts a set of values by inserting values into an existing sorted file.

Algorithm:

Input: Enter the elements which are to be sorted.

Output: Display the elements in sorted form.

Algorithm:

Let A be an array of n elements A[1],A[2],……,A[n].Pos is the control variable to Hold the position of each pass.

Step1: Input an array A of n elements.

Step2: Initialize i=1 and repeat through steps 4 by incrementing I by one.

If (i<=n-1),temp=A[i]

Pos=Pos-1

Step3: Repeat step3 if(temp<A[Pos] and (Pos>=0))

A[Pos+1]=A[Pos]

Pos=Pos-1

Step4: A[Pos+1]=temp

Step5: Exit.

Merge sort: Merge sort combines two or more sorted files into a third sorted file. Divide the file into n subfiles of size one and merge adjacent pairs of files. Repeat this process until there is only one file remaining of size n.

Algorithm:

Let A be an array of size n,A[1],A[2],….A[n]. low1 and up1 represent lower and upper bound of the first subarray and low2 and up2 represent the lower and upper bound of the second subarray.temp is an temporary array of size n. Size is the sizes of merge files.

Step1: Enter an array of n elements to be sorted.

Step2: Start with file of size n.

Step3: while (size<n) repeat step 4 to 9.

n is total no of elements.

Set low1=0,k=0

Step4: While ((low1+size) <n)

low2=low1+size

up1=low2-1

Step5: If((low2+size-1)<n)

up2=low2+size-1

else

up2=n-1

Step6: Repeat step 7 till (i<=up1) and (j<=up2)

Set i=low1,

Set j=low2

Step7: If (A[i] <=A[j])
Temp[k]=A[i++]
Else
Temp[k]=A[j++]
Step8: Copy remaining elements of either of the two subfiles into temporary Array.
8.1 Repeat step 8.1 by incrementing the value of k until (j<up1)
temp[k] =A[i++]
8.2 Repeat step 8.2 by incrementing the value of k until (j<up2)
temp[k]=A[j++]
Step9: low1=up2+1
Step10: copy any remaining single file.
Initialize i=low1and repeat the step till(i<n)by incrementing I by one
A[i]=temp[i]
Step11: size=size*2
Step 12: Stop.

Assignment no. 10

Heap sort: Heap is a collection of elements.For heap sort first we have to create a heap and then sort the elements.

There are two types of heat

Max heap: It is a complete binary tree in which the value of each node is greater than or equal to the value of its children.

Min heap: It is a complete binary tree in which the value of each node is smaller than or equal to the value of its children.

Algorithm: Algorithm to insert a new element into the heap.

Pos is the position of the newly added node element, ele. To insert an element into the heap, element is added at the bottom of the heap and then compares it with parent, grandparent, great grandparent and so on, until it is less than or equal to one of these values.

Step1: Enter an element in the heap ht.
Step2: Set pos=n and ele=ht[n].
Step3: Repeat step4 and step5, while (pos>1) and (ht [pos/2] <ele)
Step4: ht [pos] = ht [pos/2]
Step5: Set pos=pos/2
Step6: Set ht [pos] =ele
Step7: Stop.

Algorithm: Adjust the elements to satisfy heap property.

Step1: Set pos1=2*pos, ele=ht [pos]
Step2: Repeat step3 to step5 while (pos1<=n)
Step3: Compare left and right child and let the larger child be at position, Pos1.
If ((pos1<n) && (ht [pos1]<ht[pos1+1]) then
Set pos1=pos1+1
Step4: If (ele>=ht [pos1]) then stop It means the correct position for the element is found.

Step5: Set ht[pos1/2]=ht[pos1],pos1=2*pos1
Step6: Set ht[pos1/2]=ele

Algorithm: Delete the max element from heap tree and store it in x.
Step1: If n=0 then display "heap is empty".
Otherwise set x=ht[1]
Step2: ht[1]=ht[n]
Step3: call algorithm adjust to adjust the remaining of the tree.
Step4: Stop.

Algorithm : To sort the elements.
Step1: Repeat step2 and step3
Ie For i=n to 1, decrement I by 1.
Step2: call algorithm delete
Step3: set ht[i]=x.

Quick sort: Quick sort algorithm works by partitioning the array to be sorted. And each partition is internally sorted recursively.

Input: Enter the elements in the list.
Output: Display the sorted list.
Algorithm: Let A be an array of n elements A(1),A(2),A(3).....A(n),lb represents the lower bound pointer and ub represents the upper bound pointer.Key represents the first element of the array.
Step1: Enter the n elements in an array.
Step2: Initialize lb=2,ub=n.
Step3: Repeat upto step8 while(lb<ub)
Step4: Repeat step5 while(A[lb]>key)
Step5: lb=lb+1
Step6: Repeat step 7 while(A[ub]<key)
Step7: ub=ub-1
Step8: If(lb<=ub)
1> temp=A[lb]
2> A[lb]=A[ub]
3> A[ub]=temp
4> lb=lb+1
5> ub=ub-1
Step9: If(1<ub)Quicksort(A1,ub)
Step10: If(lb<n)Quicksort(A,lb,n)
Step11: Return

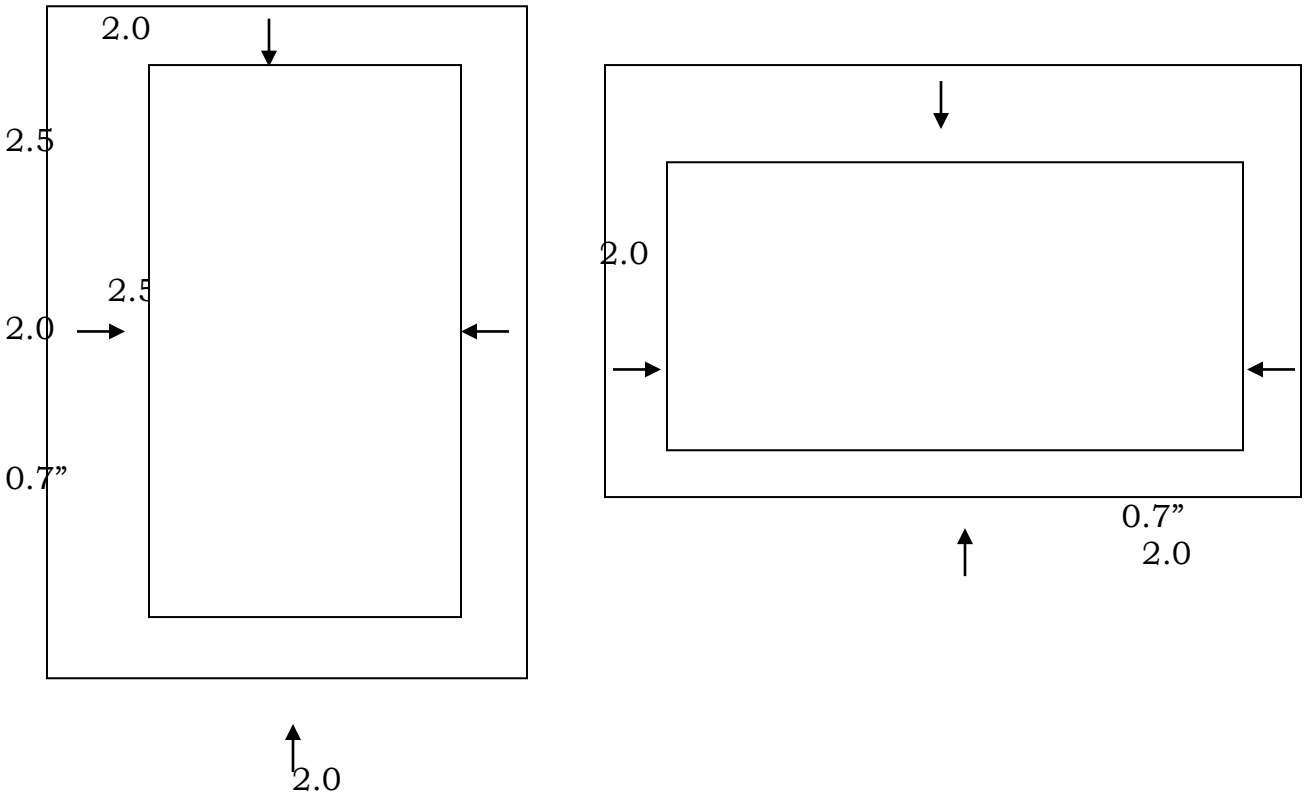
Quiz on the subject:
Quiz should be conducted on tips in the laboratory, recent trends and subject knowledge of the subject. The quiz questions should be formulated such that questions are normally are from the scope outside of the books. However twisted questions and self formulated questions by the faculty can be asked but correctness of it is necessarily to be thoroughly checked before the conduction of the quiz.

Conduction of Viva-Voce Examinations:

Teacher should oral exams of the students with full preparation. Normally, the objective questions with guess are to be avoided. To make it meaningful, the questions should be such that depth of the students in the subject is tested Oral examinations are to be conducted in co-cordial environment amongst the teachers taking the examination. Teachers taking such examinations should not have ill thoughts about each other and courtesies should be offered to each other in case of difference of opinion, which should be critically suppressed in front of the students**.**

Submission:

Document Standard:
A] Page Size A4 Size
B] Running text Justified text
C] Spacing 1 Line
D] Page Layout and Margins (Dimensions in Cms)
Normal Page Horizantal



Desription	Font	Size	Boldness	Italics	Underline	Capitalize
College Name	Arial	24	-----	------	Yes	----------
Document Title	Tahoma	22	-----	------	---------	----------
Document Subject	Century Gothic	14	-----	------	---------	Capital
Class	Bookman old Slyle	12	-----	------	---------	----------
Document No	Bookman old Slyle	10	-----	------	---------	----------
Copy write inf	Bookman old Slyle	9	-----	------	---------	----------
Forward heading	Bookman old Slyle	12	-----	------	Yes	Capital
Forward matter	Bookman old Slyle	12	-----	------	---------	----------
Lab man Contents title	Bookman old Slyle	12	-----	------	Yes	Capital
Index title	Bookman old Slyle	12	Yes	------	Yes	Capital
Index contents	Bookman old Slyle	12	-----	------	---------	----------
Heading	Tahoma	14	Yes	Yes	Yes	----------
Running Matter	Comic Sans MS	10	-----	------	---------	----------

Evaluation and marking system:

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination system to become popular amongst the students. It is a wrong approach or concept to award the students by way of easy marking to get cheap popularity among the students to which they do not deserve. It is a primary responsibility of the teacher that right students who are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking patterns should be justifiable to the students without any ambiguity and teacher should see that students are faced with unjust circumstances.