Jawaharlal Nehru Engineering College

Laboratory Manual

DATA STRUCTURE

for

SE IT Students

15, Jan 2019 – Rev 00 – IT – ISO 9001 Tech Document

FORWARD

It is my great pleasure to present this laboratory manual for SE 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 9001 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO 9001 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 MANUAL CONTENTS

This manual is intended for the SE students 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 etc. 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

Prof. V.S.Agrawal

List of Assignments

1. Program to implement STACK data type and use it to reverse a string.
2. Program for implementing integer STACK type that grows on demand.
3. Program for appropriate stacks for evaluating infix expression with parenthesis.
4. Program for QUEUE data type to simulate a bank where bank customers served first come first serve basis.
5. Program for implementing singly linked list operations.
    - Concatenate two linked list and create third one
    - Free all nodes In a linked list
    - Reverse a linked list
    - Given two linked list, Create a third list which is intersection of elements in the two.
6. Program for deleting every third element from the linked list.
7. Program to copy given linked list into another (new) list.
8. Programs to implement queue using doubly linked list.
9. Programs for recursive functions for a singly linked NULL terminated list Insert() , traverse(), search() .

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

1. Program to implement STACK data type and use it to reverse a string.

Program

Input: Read a string.
Output: .Print String in reverse way.

Algorithm:
Step1: Input a String.
Step2: Calculate the length of a string.
Step3: Push all the characters one at a time into Stack.
Continue step3 till length of a string
Step4: Pop all the characters one at a time from the stack into same string
Continue step4 till length of a string.
Step5: Print a string.
Step6: End.

Assignment no. 2

Program for implementing integer STACK type that grows on demand.

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: Ask stack size to end user.
Step3: Allocate the memory dynamically.
Step4: Increment top by one.
Step5: Insert an element e. i.e. Stack[top]=ele
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: Display Stack[top] element.
Step6: End.

Algorithm: To display elements from the stack.

Output: It will display all elements from the stack.

Step1: Start.
Step2: Check whether stack is empty or not.
Step3: If it is then print the message stack is empty.
Step4: Otherwise perform step5 from i=0 till counter reaches to top.
Step5: Display Stack[i] element.
Step6: End.

Assignment no. 3

Program for appropriate stack for evaluating infix expression with parenthesis.

1. While there are still tokens to be read in,
   1.1 Get the next token.
   1.2 If the token is:
      1.2.1 A number: push it onto the value stack.

      1.2.2 A variable: get its value, and push onto the value stack.

      1.2.3 A left parenthesis: push it onto the operator stack.

      1.2.4 A right parenthesis:

        1 While the thing on top of the operator stack is not a left parenthesis,

          1 Pop the operator from the operator stack.

          2 Pop the value stack twice, getting two operands.

          3 Apply the operator to the operands, in the correct order.

          4 Push the result onto the value stack.

        2 Pop the left parenthesis from the operator stack, and discard it.

      1.2.5 An operator (call it this Op):
        1 While the operator stack is not empty, and the top thing on the operator stack has the same or greater precedence as this Op,
          1 Pop the operator from the operator stack.

          2 Pop the value stack twice, getting two operands.

          3 Apply the operator to the operands, in the correct order.

          4 Push the result onto the value stack.

        2 Push this Op onto the operator stack.

2. While the operator stack is not empty,
   1 Pop the operator from the operator stack.
   2 Pop the value stack twice, getting two operands.
   3 Apply the operator to the operands, in the correct order.
   4 Push the result onto the value stack.

3. At this point the operator stack should be empty, and the value stack should have only one value in it, which is the final result.

Assignment no. 4

Program for QUEUE data type to simulate a bank where bank customers are served first come first serve basis.

Note:- Insert all bank customers details at the start.

Algorithm: to add customers into the queue

Steps:
Step1:Start
Step2: Ask customer bank account number.
Step3: Insert into the queue by incrementing rear.
Step4: End

Algorithm: to process customers
Steps:

Step1: Start.
Step2: Get the account number from queue by incrementing front to process.
Step3: End

Algorithm: to deposit
Steps:

Step1: Start.
Step2: Enter the amount to be deposited.
Step3: update balance set bal=bal-amt;
Step4: End

Algorithm: to withdraw
Steps:

Step1: Start.
Step2: Enter the amount to be withdrawn.
Step3: update balance set bal=bal+amt;
Step4: End

Assignment no. 5

Program for implementing singly linked list operations.

- Concatenate two linked list and create third one
- Free all nodes In a linked list
- Reverse a linked list
- Given two linked list ,create a third list which is intersection of elements in the two.

Algorithm: Creating first linked list

Steps:
Step1 :Start
Step2: Allocate memory for the new node.
Temp=malloc()
Step3: Assign the value to the data field of the new node.
Temp->info=ele
Step4: Make the link field of the new node to point to the starting node of the linked list.
Temp->next=NULL
Step4: Chk if head is NULL.
If yes then say head = temp
Otherwise perform setp 5.
Step5: Go till r->next is Not NULL.
Then r->next = temp
Step6: End

Algorithm: Creating second linked list

Steps:
Step1 :Start
Step2: Allocate memory for the new node.
Temp=malloc()
Step3: Assign the value to the data field of the new node.
Temp->info=ele
Step4: Make the link field of the new node to point to the starting node of the linked list.
Temp->next=NULL
Step4: Chk if head1 is NULL.
If yes then say head1 = temp
Otherwise perform setp 5.
Step5: Go till r->next is Not NULL.
Then r->next = temp
Step6: End

Algorithm: To Concatenate above two linked list

Steps:

Step1:Set head2=NULL, temp=head
Step2:Go till temp is Not NULL
Allocate memory for the new node.
Temp1=malloc()
Chk head2 is NULL
If yes then head2=temp1;
Otherwise perform setp 3.
Step3: Go till r->next is Not NULL.
Then r->next = temp1
Step4: Assign temp=head1
Step5:Go till temp is Not NULL
Allocate memory for the new node.
Temp1=malloc()
Chk head2 is NULL
If yes then head2=temp1;
Otherwise perform setp 3.
Step6: Go till r->next is Not NULL.
Then r->next = temp1
Step7:End

Algorithm: To delete all node from linked list
Steps:
Step1:Start
Step2: Go till ((r=head)!=NULL)
Say head=head->next;
free(r);
Step3:End

Algorithm: To reverse a linked list
Steps:

Step1: Start
Step2: Set cur = head;
prev = NULL;
nxt= NULL;
Step3: Go till cur != NULL
// Store next
nxt = cur->next;
// Reverse current node's pointer
cur->next = prev;

// Move pointers one position ahead.
prev = cur;
cur = nxt;
Step4:Set head = prev;

Algorithm: Intersection of two linked lists
Note:- pass head of both linked list as input

```
struct Node *getIntersection(struct Node *head1,
                             struct Node *head2)
{
    struct Node *result = NULL;
    struct Node *t1 = head1;

    // Traverse list1 and search each element of it in
    // list2. If the element is present in list 2, then
    // insert the element to result
    while (t1 != NULL)
    {
        if (isPresent(head2, t1->data)) // to find out common element
            push (&result, t1->data);
        t1 = t1->next;
    }

    return result;
}

void push (struct Node** head_ref, int new_data)
{
    /* allocate memory for new node */
    struct Node* new_node =
        (struct Node*) malloc(sizeof(struct Node));

    /* put in the data */
    new_node->data = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

bool isPresent (struct Node *head, int data)
{
    struct Node *t = head;
    Go till  (t != NULL)
    {
        Check  is t->data = data
            return 1;
        t = t->next;
    }
    return 0;
}
```

Assignment 6:

Program for deleting every third element from the linked list.

Algorithm:

Steps:

Step1:Start
Step2: if head is NULL
return NULL;
Step3: set ptr = head, prev = NULL;
Step4:
// Traverse list and delete every 3rd node
int count = 0;
Go till (ptr != NULL)
{
// increment Node count
count++;

// check if count is equal to k
// if yes, then delete current Node
if (count==3)
{
// put the next of current Node in
// the next of previous Node
delete(prev->next);
prev->next = ptr->next;

// set count = 0 to reach further
count = 0;
}

// update prev if count is not 0
if (count != 0)
prev = ptr;

ptr = prev->next;
}

Step5: return head;
Step6: End

Assignment 7: Program to copy given linked list into another (new) list.

Algorithm: Creating first linked list

Steps:
Step1 :Start
Step2: Allocate memory for the new node.
Temp=malloc()
Step3: Assign the value to the data field of the new node.
Temp->info=ele
Step4: Make the link field of the new node to point to the starting node of the linked list.
Temp->next=NULL
Step4: Chk if head is NULL.
If yes then say head = temp
Otherwise perform setp 5.
Step5: Go till r->next is Not NULL.
Then r->next = temp
Step6: End

Algorithm : To copy a linked list

Steps:
Step1:Start
Step2:Set head1=NULL, temp=head;
Step3:Go till temp!=NULL
Allocate the memory
Temp1=malloc()
Chk if head1=NULL
If yes then head1=temp1;
Otherwise
r=head3;
while(r->next!=NULL)
r=r->next;
r->next=temp1;

temp=temp->next

Step4:End

Assignment no. 8
Programs to implement queue using doubly linked list.

Algorithm: Insert a node in liked list.

Steps:

Step1: Allocate memory for the new node.
Temp=malloc()
Step2: Assign the value to the data field of the new node.
Temp->info
Step3: Chk if head=NULL then head=temp
Otherwise
r=head;
while(r->next!=NULL)
r=r->next;
r->next=temp;
Step4:End

Algorithm: Delete a node from linked list.

Step1: Start
Step2: If list is empty then display list is empty
Otherwise
r=head;
no=r->no;
head=head->next;
free(r);

Step3:End

Algorithm: To display a linked list

Step1:Start
Step2: Chk list is empty then display list is empty
Otherwise

```
r=head;
while(r!=NULL)
{
    cout<<r->no<<endl;
    r=r->next;

}
```

Assignment 9:
Programs for recursive functions for a singly linked NULL terminated list Insert() , traverse(), search() .

Note:- 2 parameters passed 1:-head 2:-no to be inserted

```
node* newNode(int no)
{
    node *new_node = new node;
    new_node->no = no;
    new_node->next = NULL;
    return new_node;
}
```

Algorithm: Insert new node .

Step1:Start
Step2: if s is NULL then return newNode(no);
Otherwise

```
    {
       s->next= call same function by passing(s->next,no);
    }
    cout<<"node attached"<<endl;
     return head;
```

Step3:End

Algorithm: Traverse all node

Note:- head passed to the function as argument

Step1:Start
Step2: if (s == NULL)
return ;
Otherwise

```
   cout << s->no << " ";
   call same function by passing(s->next);
```

Algorithm: Search a number in linked list

Note:- Head and number to search passed
Steps:
Step1:Start
Step2: if (s == NULL)
return 0 ;
else if (s->no==no)
return 1;
else
call same function by passing (s->next,no);

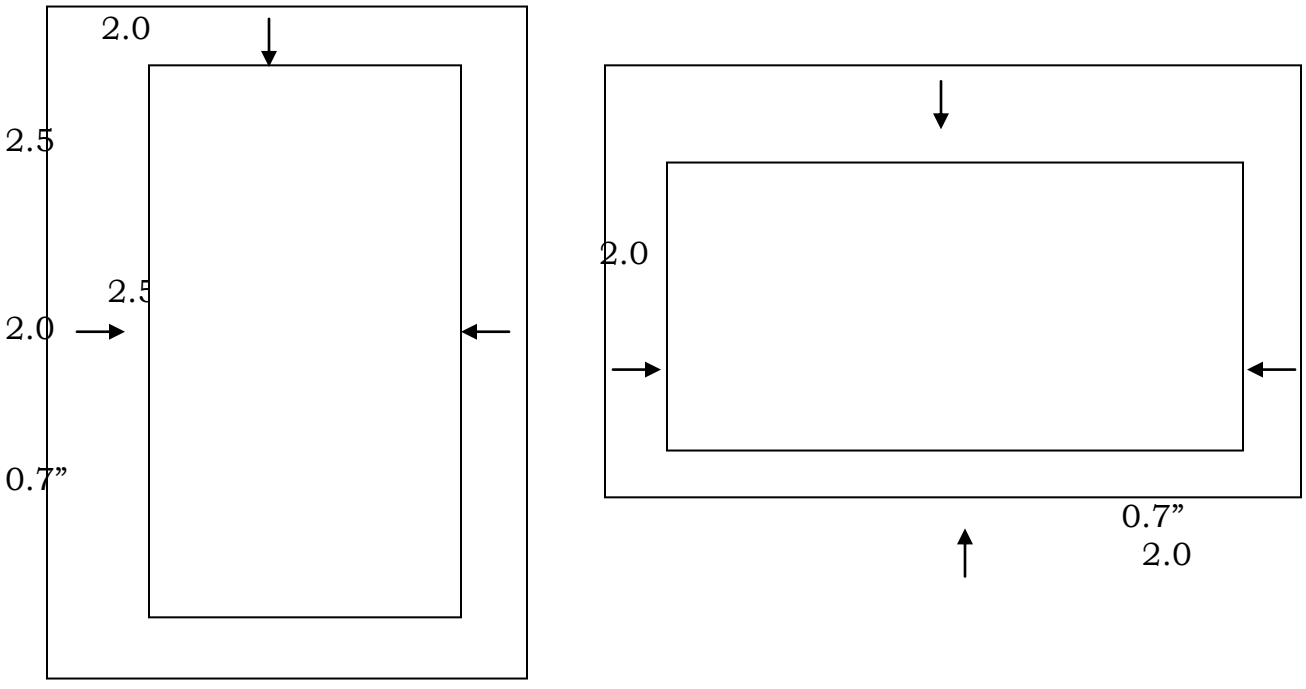
Quiz on the subject:
Quiz should be conducted on tips in the laboratory, recent trends and subject knowledge of the subject. The quiz questions should be formulated such that questions are normally are from the scope outside of the books. However twisted questions and self formulated questions by the 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.