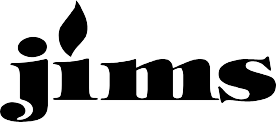
**JAGANNATH INTERNATIONAL MANAGEMENT SCHOOL**



**(Department of Information Technology)**

**Laboratory Manual**

**Data Structure Using ‘C’**

**Bachelor of Computer Applications Semester-IInd**

**Subject Code: BCA 174**

**(Session : Jan 2024 – Jul 2024)**

**Submitted To: Submitted By:**

Ms.Neha Chhabra Name

Assistant Professor–IT Department RollNo.

JIMS, Vasant Kunj, New Delhi Class

# List of Programs



|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Program Statement** | **Date** | **Signature** |
|  | WAP using recursion to check whether a number is prime number or not. |  |  |
|  | WAP using recursion to print fibonacci series upto ‘n’ terms |  |  |
|  | WAP to find GCD of two numbers using recursion. |  |  |
|  | WAP to reverse a number using recursion. |  |  |
|  | WAP to implement following operation on one dimensional array (i) Insertion (ii) Deletion (iii) Traversal (iv) Reverse (v) Merge |  |  |
|  | WAP to search & display the location of an element specified by the user, in an array using (i) Linear Search for unsorted list (ii) Linear Search for sorted list (iii) Binary Search technique using function (iv) Binary Search technique using recursion. |  |  |
|  | WAP to find the frequency of all elements in a 1-D array and delete the duplicate elements keeping only the first copy of the element. |  |  |
|  | WAP to Sort an array using menu driven: (i) BUBBLE SORT (ii) SELECTION SORT (iii) INSERTION SORT (iv) MERGE SORT |  |  |
|  | WAP to implement following operation on matrices (i) Addition (ii) Subtraction (iii) Multiplication (iv) Transponse (v) calculate trace of a matrix (vi) calculate norm of a matrix |  |  |
|  | WAP to store triangular[lower/upper] matrix in1-D array to save memory. |  |  |
|  | WAP to accept a matrix from user, find out matrix is sparse or not and convert into triplex matrix. |  |  |
|  | WAP to implement a Singly Linked List. |  |  |
|  | WAP to implement a Circular Linked Lists |  |  |



|  |  |  |  |
| --- | --- | --- | --- |
|  | WAP to implement Doubly Linked Lists |  |  |
|  | WAP to implement Polynomial addition operation using linked list. |  |  |
|  | Write a C program to create two linked lists from a given list in following way  INPUT List:- 1 2 3 4 5 6 7 8 9 10  OUTPUT:- First List:- 1 3 5 7 9 Second List:- 2 4 6 8 10 |  |  |
|  | WAP to implement student database using linked list with the following structure:  “Name, rollno, marks of 5 subjects, average, result”  If the average is less than 50 then print ‘fail’ otherwise ‘pass’. |  |  |
|  | Write a menu driven program to implement (i) Static  Stack (ii) Dynamic Stack. |  |  |
|  | Write a program to convert Infix to equivalent (i) Prefix expression (ii) Postfix expression |  |  |
|  | Write a program to evaluate (i) Prefix Expression (ii) Postfix Expression using stack. |  |  |
|  | WAP to implement a (i) Static (ii) Dynamic Linear Queue |  |  |
|  | WAP to implement a (i) Static (ii) Dynamic Circular Queue |  |  |
|  | WAP to implement a (i) Static (ii) Dynamic De-Queue |  |  |
|  | Let us assume a Patient's coupon generator for the Doctors’ clinic. The patients are given the coupons on first-come-first-serve basis. After the visit of a patient, patient-ID is kept stack-wise. At the end of the day, the count is generated from the stack. Construct a menu-based program for patients’ coupons generator using an appropriate data structure. |  |  |
|  | Sometimes a program requires two stacks containing the same type of items. Suppose two stacks are stored in separate arrays, then one stack might overflow while there is considerable unused space in the other. A neat way to avoid this problem is to put all spaces in one stack and let this stack grow from one end of the array, and the other stack starts from the other end and grows in the opposite direction, i.e., toward the first stack. In this way, if one stack turns out to be large and the other small, then they will still both fit, and there will be no overflow until all space is used. Declare a new structure that includes these two stacks and perform various stack operations. |  |  |

# Instructions to be Strictly followed



* Outputs of the programs should be yours only and original printout [Not Photocopied].
* At the end of semester you need to spiral bound your file with fresh front page. In between, bring your programs alonwith index in stick file for evaluation purpose.
* **Every student will be responsible for evaluation of their practical file on time as which will be announced time to time during labs failing which their practical file will not be evaluated. To avoid any problem, submit your file for evaluation on or before the mentioned dates. [Do not give any excuse for not submitting your file on time.]**