Course code: CSL 201 Course Name : Data Structures Lab

Faculty In Charge: Dr Binu V P

L-T-P-Credits 0-0-3-2

Pre-requisite: CST 201 Data Structures, EST 102 C programming skills.

Operating System to Use in Lab : Linux

Compiler/Software to Use in Lab : gcc

Programming Language to Use in Lab : Ansi C

Preamble: The aim of the Course is to give hands-on experience for Learners on creating and using different Data Structures. Data Structures are used to process data and arrange data in different formats for many applications. The most commonly performed operations on data structures are traversing, searching, inserting, deleting and few special operations like merging and sorting.

Lab Cycle-3

Learning Outcome: Learn stack and its applications

Date of submission: on or before 24-12-2021

(Write these programs in fair record-show the output in lab and get it signed by the staff in charge. There will be viva voce in every lab.)

Stack

Stack is a data structure having LIFO structure. PC uses stacks at the architecture level, which are used in the basic design of an operating system for interrupt handling and operating system function calls. Recursive programs uses stack extensively. The design of compilers, language recognizers, expression evaluation etc uses stack. Many languages has a class called "Stack", which can be used by the programmer.

1) Implement a stack using an array.

2) Implement multiple stacks(2 stacks) using an array. Consider memory efficient implementation

3)Find the minimum element in a stack in O(1) time using an auxiliary stack which keeps track of the minimum element.

4) implement a sorted push so that stack is always maintained in sorted order.

Using the stack do the following programs

4) Convert a given decimal number into binary and hex.

5) Check whether a string is palindrome.

Expression evaluation and syntax parsing

Calculators employing reverse Polish notation use a stack data structure to hold values. Expressions can be represented in prefix, postfix or infix notations. Conversion from one form of the expression to another form needs a stack. Many compilers use a stack for parsing the syntax of expressions, program blocks etc. before translating into low level code. Most of the programming languages are context-free languages allowing them to be parsed with stack based machines.

6) Check whether the parenthesis are balanced in an expression.

7) Convert a given infix expression to postfix/prefix

8) Evaluate a postfix/prefix expression.