**LIST OF PROGRAMS**

**Part - A**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Program** | **CO** |
| 1 | Use Stack operations to do the following:   1. Assign to a variable name Y the value of the third element from the top of the stack and keep the stack undisturbed. 2. Given an arbitrary integer n pop out the top n elements. A message should be displayed if an unusual condition is encountered. 3. Assign to a variable name Y the value of the third element from the bottom of the stack and keep the stack undisturbed.   (Hint: you may use a temporary stack) | CO1 |
| 2 | Write a C program that parses Infix arithmetic expressions to Postfix arithmetic expressions using a Stack. | CO3 |
| 3. | |  | | --- | | Write a C program to simulate the working of Messaging System in which a message is placed in a circular Queue by a Message Sender, a message is removed from the circular queue by a Message Receiver, which can also display the contents of the Queue. | | CO3 |
| 4. | |  | | --- | | Implement a program to multiply two polynomials using single linked list. | | CO2 |
| 5. | |  | | --- | | Write a C program to implement addition of long positive integers using circular single linked list with header node. | | CO4 |
| 6. | |  | | --- | | Design a doubly linked list to represent sparse matrix. Each node in the list can have the row and column index of the matrix element and the value of the element. Print the complete matrix as the output. | | CO4 |
| 7. | |  | | --- | | Write a C program to create Binary Tree and provide insertion and deletion operations and to traverse the tree using In-order, Preorder and Post order (recursively) | | CO3 |
| 8. | |  | | --- | | Given a String representing a parentheses-free infix arithmetic expression, implement a program to place it in a tree in the infix form. Assume that a variable name is a single letter. Traverse the tree to produce an equivalent postfix and prefix expression string. | | CO3 |
| 9. | Write a C program to implement Hashing using Linear probing. Implement insertion, deletion, search and display. | CO3 |
| 10. | |  | | --- | | Write a C program to implement priority queue to insert, delete and display the elements. | | CO4 |

**PART - B**

Student will design, develop and implement an application using the appropriate data structure. Some example applications are listed below:

* Huffman coding
* Dictionary implementation for Indian Languages
* Stemmer implementation for Indian language
* Word frequency finder.
* Bitmap Image Compression.
* Binary Tree (Graphical Implementation)
* To store a set of programs which are to be given access to a hard disk according to their priority
* For representing a city region telephone network.
* To store a set of fixed key words which are referenced very frequently.
* To represent an image in the form of a bitmap.
* To implement back functionality in the internet browser.
* To store dynamically growing data which is accessed very frequently, based upon a key value.
* To implement printer spooler so that jobs can be printed in the order of their arrival.
* To record the sequence of all the pages browsed in one session.
* To implement the undo function.
* To store information about the directories and files in a system.

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. no.** | **Program Name.** | **Date** | **Record Marks**  **(max 6)** | **Viva Voice**  **(max 4)** | **Total Marks** |
| **1.** | Stack Operations |  |  |  |  |
| **2.** | Infix to Postfix using Stack |  |  |  |  |
| **3.** | Messaging system |  |  |  |  |
| **4.** | Multiplication of polynomials |  |  |  |  |
| **5.** | Addition of long positive integers |  |  |  |  |
| **6.** | Sparse Matrix using doubly linked list |  |  |  |  |
| **7** | Binary Tree operations |  |  |  |  |
| **8** | Traversal of trees |  |  |  |  |
| **9** | Hashing using Linear probing |  |  |  |  |
| **10** | Priority queue operations |  |  |  |  |
|  | **Total (100)** |  |  |  |  |

**Part B**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Problem definition (1)** | **Application of relevant data structure with justification**  **(1)** | **Incorporation of suggestions (1)** | **Design (1)** | **Source Code (2)** | **Demonstration (2)** | **Documentation (2)** | **Total (10)** |
|  |  |  |  |  |  |  |  |

**Part – A**

**REDUCED RECORD MARKS = (TOTAL/100) \*\_\_\_\_ = \_\_\_\_\_\_\_\_\_**

**Part A + Part B = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |
| --- | --- | --- |
| **LAB INTERNALS** | | |
| **RECORD** | **Max -** |  |
| **TEST** | **Max -** |  |
| **TOTAL** | **Max - 50** |  |
|  | *Signature of the faculty* |  |