**PROJECTS LIST OF DS**

1. Take an **INFIX** algebraic expression from the user then convert it into equivalent **POSTFIX** and **PREFIX** notation.
2. Take an **INFIX** algebraic expression from the user then convert it into binary tree which shows the expression should be solved in specific order.
3. Make **“DICTIONARY”.** User can add words in the dictionary and hence enhance the database of dictionary. If user wants to search any text, it searches and displays the meaning of respective word on screen. The searching algorithm must be **binary search**. (In which all words must be already sorted).
4. Make a game **“TOWER OF HANOI”** for **SIX** plates using STACK in C#. Instead of using plates used Alphabets, A is less than B and so on If you want
5. Make a game **“GAME OF NIM”** for **2 players.**
6. Make a game **“TIC TAC TOE”** for two players but the rows and columns are 7x7.
7. Implement **HASHING** in Data Structures
8. Implement **HUFFMAN CODE**. Take any string from user then convert it to respective HUFFMAN CODE. Show the simulation of all the steps involved.
9. Make a game **“TETRIS”** in C#. There are number of alphabets dropping from top to down in the STACK of game, if user did not press the alphabet while it dropping, then the alphabet dropped into STACK and when STACK becomes full, it will stop and show the score.
10. Row Transposition Cipher (Ciphering and deciphering). All parameters are given by user. Two iterations at least required for ciphering. Show all the steps on screen of ciphering and deciphering.
11. Make a game “**SHOOTER**” in which the shooter shoots the alphabets dropping from top to down. If any alphabet hits the shooter, the shooter will destroy. Maximum 5 lives of shooter are given. There would be certain alphabets that are dropping much faster than other alphabets, if shooter hits that alphabet then the live will increase by 1.
12. Make a **“Typing Tutor”** in C#. There is a paragraph given to user. User types that paragraph. If the user typed wrong character, then it would not display. Only right words typed should display. At the end it shows total number of wrong characters typed by user.
13. Make a small database management system using linked list
14. Make a game “PACMAN” in C#.
15. Implement play fair cipher in C#. Take any Text from user then cipher it with PLAY FAIR cipher. Show each step while Ciphering and Deciphering it.
16. Mines game
17. Simulation of binary tree.
18. Take an array from the user, sort it by performing step by step simulations of insertion, selection sort,bubble sort, merge and quick sort, radix sort
19. Generalized Metrics solver (unlimited dimensions, un-limited matrices)
20. Snake game (if eat bonus food its size should decrease).
21. Make INDEX (How much times a word comes in text on which page)
22. Take an array from the user, apply Boyer Moore algorithm on it for searching. Show full simulation of each process step-by-step
23. Take a paragraph from the user or from file, then apply Horsepool algorithm on it for searching the pattern inside the paragraph. Show full simulation of each process step-by-step
24. Expression evaluator
25. Simulation of each step of heap construction, also simulate the insertion and deletion in heap as well as heapsort.
26. Take any range from user then generate Random Number Generator by Linear Congruential Method within the range, if the user entered less than that value then the message shown” YOU ENTERED SMALLER VALUE” and if the user entered larger value then the message shown “YOU ENTERED HIGHER VALUE”, and if the user entered that same value then display the message “GREAT YOU WON THE GAME”. Make the game level “Easy”, “Medium” and “Hard”. In easy the trials are more for user while in hard the trials are less.
27. Let the user to construct the bidirectional weighted graph, then you simulate step by step construction of Minimum Spanning Tree by Prim’s Algorithm.
28. Develop a search engine by using any efficient searching algorithm. It also suggest searcher the related searches.
29. Let the user to construct the bidirectional weighted graph, then you simulate step by step construction of all Hamiltonian Circuits inside the graph. Also identify the shortest Hamiltonian Circuit in them.
30. Develop your own searching and sorting algorithm. Show step by step simulation of your proposed algorithm.
31. Model that simulate CAR security System.
32. A Calendar for any month for any year entered by user.
33. TEXT EDITOR with all basic functions.
34. Model that simulates two LIFTS in a 10 floor building.
35. Any logical game developed by you.
36. Contact book using double linked list
37. Implement **Stack** and **Queue** using **linked** **list** with following operation, user can add, delete, update and search data in stack and queue.
38. Traverse a directory and process all text files and sub directories found in that directory.and make a tree of that
39. **Knapsack Algorithm**
40. Implementation of Dynamic Programming. Consider yourself a thief and you have Knap with k-capacity, you are in a shop and you have n numbers of item with weight and value you have to fill this knap that give you a maximum profit with k-capacity.
41. **Spell Checking**

Spell checking is the process of verifying that a particular word is spelled properly according to some dictionary. Spell checkers are used in many applications, including word processors (such as Microsoft Word), electronic dictionaries, and optical character recognition (OCR) systems that need to turn images of printed text (or even handwriting) into coherent text. Spell checking itself is trivial, requiring only a simple lookup in a dictionary. However, most applications of spell checking also require that the spell checker provide a list of potentially correct spellings (“near matches”) when the word was spelled improperly. For instance, if I type “spelling” into an online dictionary, it will provide suggestions of similar words that Imay have meant to type, including “spelling”, “spoiling”, “sapling”, and “splendid”. Your task is to implement a spell checker that determines if a given word is spelled correctly based on a dictionary lookup. When the word is not spelled correctly, it will provide a list of similar-sounding words based on your implementation of the Metaphone algorithm, and ordered based on their edit distance from the string that the user typed. We could use only the edit distance to find near matches, but by first using the Metaphone algorithm we achieve better results because it can find the word you are looking for even if you have no idea how to

spell it. Additionally, we can isolate the set of near matches more quickly using the Metaphone algorithm. Read on to learn about the Metaphone algorithm, which will be used to isolate the set of near matches; and an edit distance algorithm, which will be used to rank the results (notice that “spelling” comes before “spoiling”). Fig. 1 gives an overview of the spellchecking process.