Name: Stuti Bimali

USN : 1NH18CS210

Semester: 3rd

Sec: C

Department: CSE

Topic: Word Bank for Recognizing

Chapter 1

1. INTRODUCTION:

This is offline software which does not need internet connectivity. You can know the meaning of words without internet connection. Here you can add words with their meanings, search the meaning of words, and delete the word from the dictionary. There are some people who do not recognize the word properly.

A simple implementation of word bank can be based on sorted or unsorted lists. In this project, the binary search tree is used as in tree it is easy to traverse. Binary search tree makes it easier to traverse the tree. When searching the word, using a tree, we do not need to search all the data. It makes work easier.

Similarly in deletion the word is easily found and to delete the word we do not need to traverse till the end of the tree. It checks whether which part is the word left or right and according to that if it is in the left only left side is traversed which saves time. Hence using tree if we create a word bank, compile time is less as in linked list we need to traverse all data one by one to search a word or insert it whereas in tree the time is average.

1.1 PROBLEM DEFINITION:

A word bank commonly known as dictionary helps people to know the meaning of a word. It helps to improve vocabulary. It is very helpful for everyone to have an offline software where you can find the meaning of words or if a person wants to create his own dictionary, it is the best software as it can be used everywhere irrespective of the internet connection or carrying big books trouble. It was discovered for making people know meanings of most of the word to improve English vocabulary.

Word bank is a simple program in which it allows a person who wanted to create his own dictionary. It can be used to search for words whose meaning he doesn't remember. The problem is that we forget simple and simple things. We get confused in meaning of different words. Hence to specify a word to its own meaning, this code

helps. The person will be able to search the meaning of the specific word, if sometimes it is not there, he can add the word and meaning and delete the words he is familiar with.

1.2 OBJECTIVES:

In a normal world, we forgot simple and simple things. We get confused in meaning of different words. To specify the word with its own and correct meaning, I developed this project. We can search words, but sometimes some words are not there and we probably want to add them. And sometimes there are some words which we are already familiar with and we want to delete them as they occupy memory. The purpose of the project is to provide good and proper meaning of the word. The project's name is itself "Word Bank for Recognizing", so it means to distribute the correct meaning of that word. To specify the word with its own and correct meaning.

1.3 METHODOLOGY TO BE FOLLOWED:

- The insert function is created to insert the word and its meanings,
- A delete function is created to delete the unwanted words and their meanings,
- A search function is created which helps user to search for words whose meaning he wants to know,
- A display function is created which helps to display all the words and their meanings available in the word bank.

1.4 EXPECTED OUTCOMES:

- By using this, we can add a word into the software with its meanings,
- Search and display the meaning of the words,
- We can delete the meanings of the words we are familiar with or unwanted words,

- We can give not only one but two meanings for one word, which is very informative and helpful,
- Using this software you can create your own little word bank where you will be able to edit the contents.

1.5 HARDWARE AND SOFTWARE REQUIREMENTS:

1.5.1 HARDWARE SYSTEM CONFIGURATION:

Processor	- Intel Core i5
Speed	- 1.8 GHz
RAM	- 256 MB (min)
Hard Disk	- 10 GB

1.5.2 SOFTWARE SYSTEM CONFIGURATION:

Operating System	- Windows 7 / 8.1 / 10
Programming Language	- C ++
Compiler	- Turbo C++

Chapter 2

2. DATA STRUCTURES

Data structure is a logical or mathematical organization of data; it describes how to store the data and access data from memory.

Data structures are structures programmed to store ordered data, so that various operations can be performed on it easily. It represents the knowledge of data to be organized in memory. It should be designed and implemented in such a way that it reduces the complexity and increases the efficiency.

Primitive data structures : Integer, Float, Boolean, char etc.

Abstract data structures, which are used to store large and connected data. Examples : Linked List, Tree, Graphs, Stack, Queue etc.

Depending upon the organization of the elements, data structure are classified into two types:

1. **Linear data structure:**

   Elements are accessed in a sequential order but it is not compulsory to store all elements sequential. Examples : linked lists, stacks, and queues.

2. **Non – linear data structure:**

   Elements of this data structure are stored/accessed in a non – linear order. Examples : trees, and graphs.

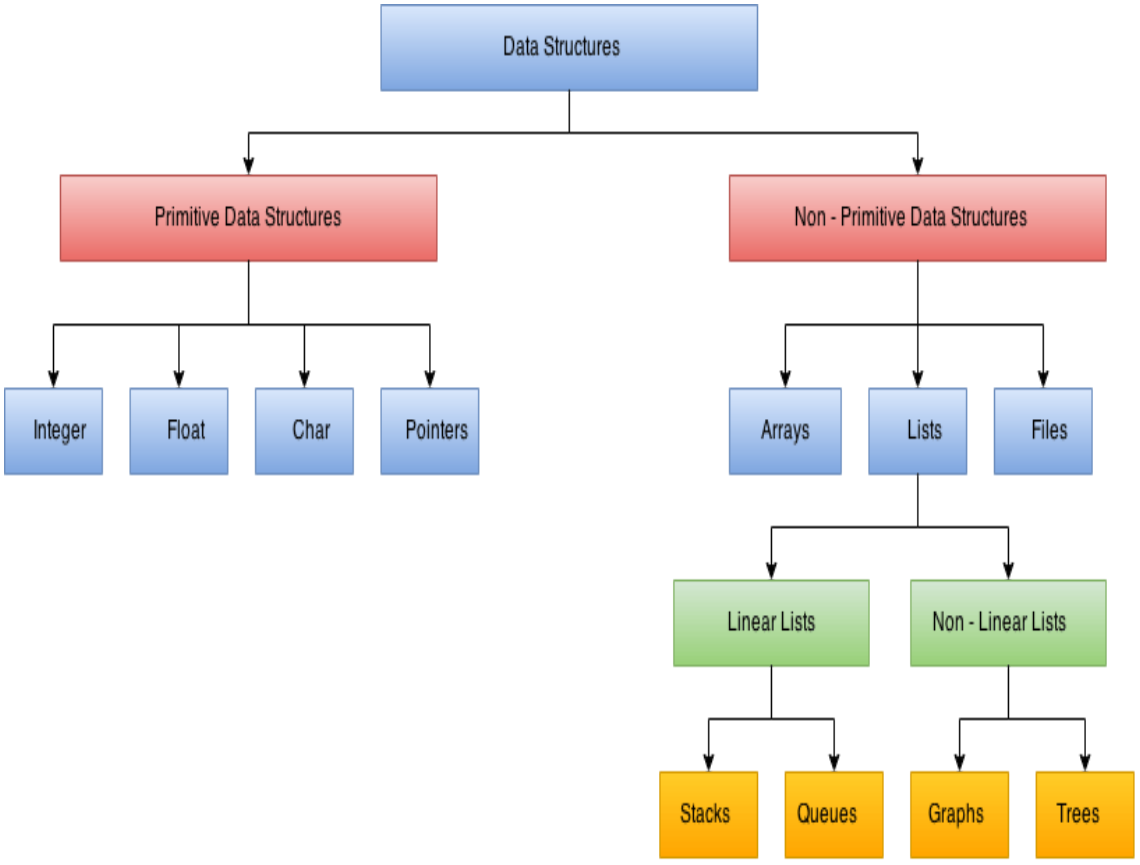

Figure 1: TYPES OF DATA STRUCTURE

2.1 STACK :

A stack is a container of objects that are inserted and removed according to the last-in first out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure – elements can be added and removed from stack only at the top. Push adds an item to the top of the stack, pop removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.

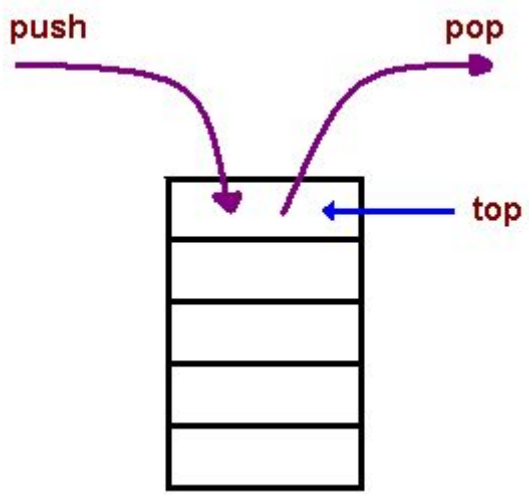

Figure 2: STRUCTURE OF STACK IMPLEMENTATION

2.2 QUEUE:

A queue is a linear list of elements in which deletion of an element can take place only at one end called the front and insertion can take place on the other end which is termed as the rear. In the concept of queue, the first element to be inserted in the queue will be the first element to be deleted or removed from the list. So queue is said to follow the FIFO (First In First Out) structure. The most basic queue operations in the data structure are enqueue and dequeue where enqueue adds an element at the beginning of the queue and dequeue deletes an element at the end of the queue.

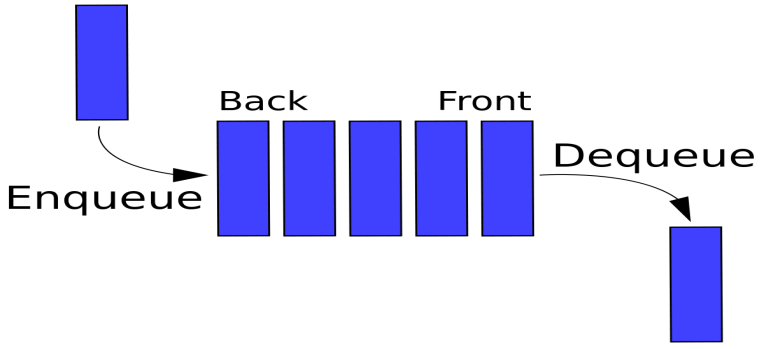

Figure 3: STRUCTURE OF A QUEUE IMPLEMENTATION

2.3 LINKED LIST:

Linked list is a linear data structure that consists of a sequence of elements where each element (usually called a node) comprises of two items – the data and a reference (link) to the next node. The last node has a reference to null. The entry point in the linked list is called the head (start) of the list. It should be noted that the head is not a separate node, but the reference to the first node. If the list is empty then the start is a null reference. The list with no nodes – empty list or null list. There are three types of linked list. They are:

1. **Singly linked list**
2. **Doubly linked list**
3. **Circular linked list**

In the singly linked list, each node except the last one contains a single pointer to the next element. The last node has a null pointer to indicate the termination of the list. In doubly linked list, each node contains two fields called links, that are references to the previous and next link, respectively, point to NULL, to facilitate traversal of the list. In circular linked list first and final nodes are linked together. There is no null at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.

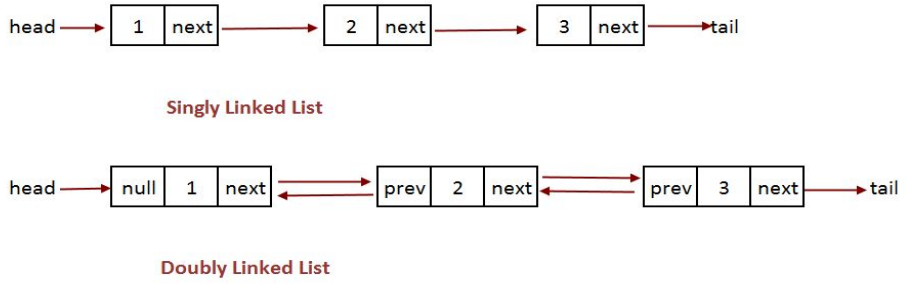

Figure 4: STRUCTURE OF SINGLY LINKED LIST AND DOUBLY LINKED LIST

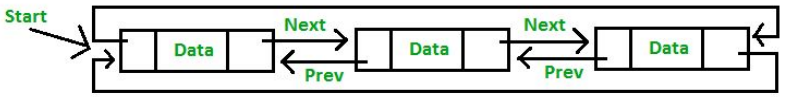

Figure 5: STRUCTURE OF CIRCULAR LINKED LIST

2.4 TREES:

Tree is a non-linear data structure which organizes data in hierarchical fashion and the tree structure follows a recursive pattern of organizing and storing data. Every individual element is called as Node. Node in a tree data structure, stores the actual data of that particular element and link to the next element in hierarchical structure. If there are N number of nodes in a tree structure, then there can be a maximum of N-1 number of links.

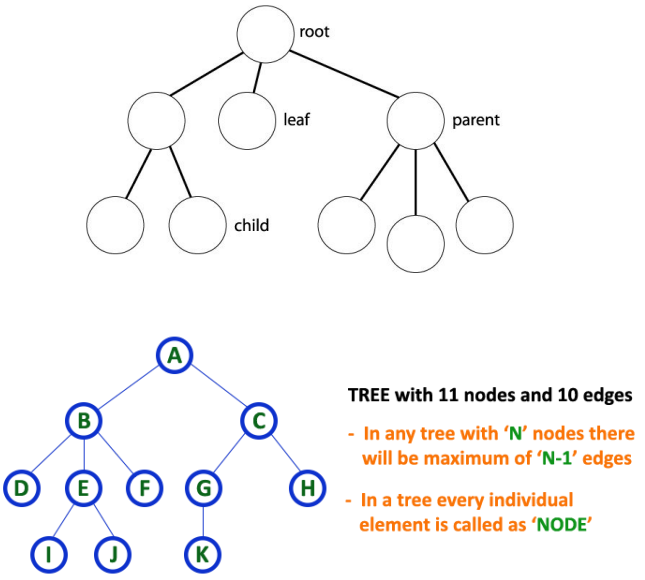

Figure 6: STRUCTURES OF A TREE

For implementing the program, I have used tree. In this program tree helps us to recursively following a pattern of organizing and storing data. Traversing is done as inorder traversal in this project. In tree, actual data of the particular element and link to next element is stored in a hierarchical structure which in turn makes the code easy.

Tree is a data structure similar to a linked list but instead of each node pointing simply to the next node in a linear fashion each node points to a number of the node. Unlike linked list, tree has a link to its sub trees.

2.4.1 TERMINOLOGY:

- **Root**

  Root node is the node which do not have parent node. In a tree, there is only one root node.

- **Child**

  Child node are the nodes having parent node. In a tree data structure, the node which is a descendent of any node is called as the child node. In a tree, any parent node can have any number of child nodes. In a tree, all the nodes except root are child nodes.

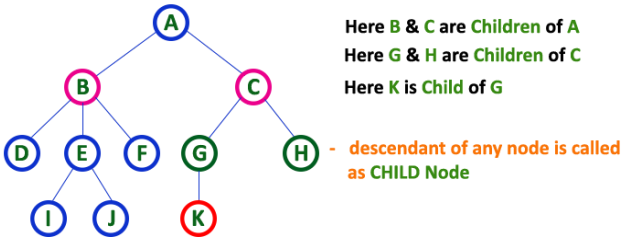

Figure 7: DIAGRAM TO DEMONSTRATE CHILD NODE

- **Edge**

  In a tree data structure, the connecting link between any node are called as edge. In a tree with **N** number of nodes there will be a maximum of **N-1** number of edges.

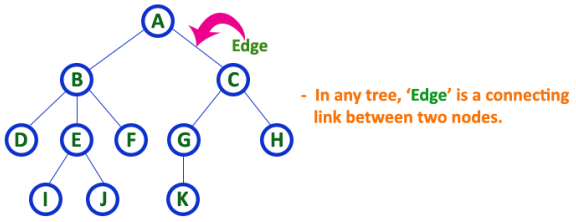

Figure 8: DIAGRAM TO DEMONSTRATE EDGES

- **Parent**

In a tree data structure, the node which is a predecessor of any node is called as parent node. In simple words, the node which has a branch from it to any other node is called a parent node. Parent node can also be defined as the node which has child/children.

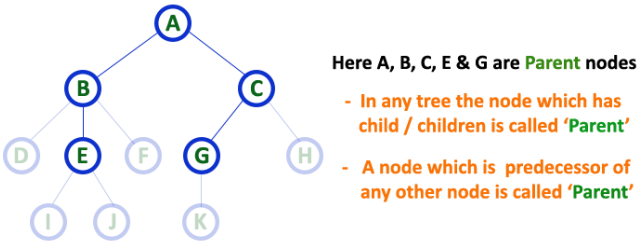

Figure 9: DIAGRAM TO DEMONSTRATE PARENT NODE

- **Leaf**

In a tree data structure, the node which does not have a child is called a leaf node. In simple words, a leaf is a node with no children. In a tree data structure, the leaf nodes are also called as external node. External node is also a node with no children. In the tree, leaf node is also called as terminal node.

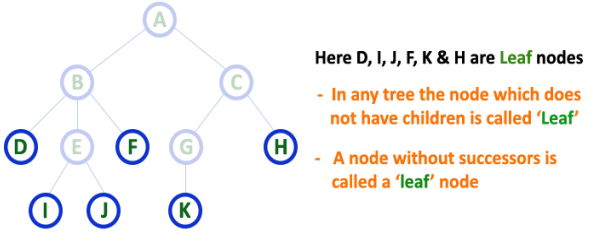

Figure 10: DIAGRAM TO DEMONSTRATE LEAF NODE

- **Siblings**

  In a tree data structure, nodes in the same level belonging to the same parent are called siblings. In simple words, the node with the same parent are called sibling node.

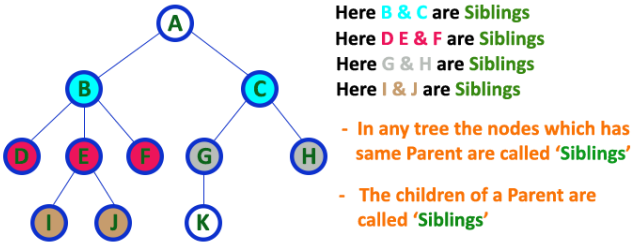

Figure 11: DIAGRAM TO DEMONSTRATE SIBLINGS

- **Internal node**

  In a tree data structure, the node which has at-least one child is called as internal node. In simple words, an internal node is a node with at least one child. In a tree data structure, nodes other than leaf nodes are called as internal nodes. The root node is also said to be internal node if the tree has more than one node. Internal nodes are also called as non terminal node.

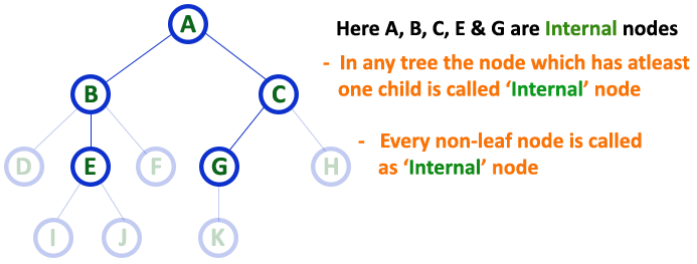

Figure 12: DIAGRAM TO DEMONSTRATE INTERNAL NODE

- **Degree**

  In a tree data structure, the total number of children of a node is called as degree of that node. In simple words, the degree of a node is the total number of children it has. The highest degree of a node among all the nodes in a tree is called as degree of tree.

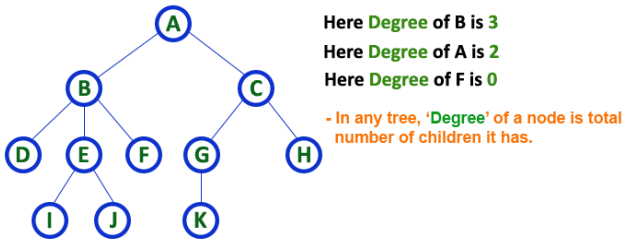

Figure 13: DIAGRAM TO DEMONSTRATE DEGREE

- **Height of tree**

In a tree structure, total number of edges from leaf node to a particular node in the longest path is called the height of the tree. In a tree, height of the root node is said to be the height of the tree. In a tree, height of all leaf nodes is 0.

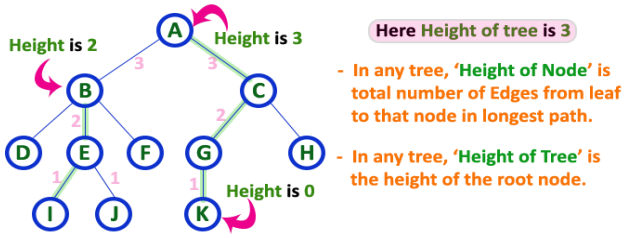

Figure 14: DIAGRAM TO DEMONSTRATE HEIGHT OF TREE

- **Depth**

In a tree structure, the total number of edges from root to particular node is called as the depth of the node. In a tree, the total number of edges from the root node to a leaf node in the longest path is said to be the depth of the tree. In simple words, the highest depth of any leaf node in a tree is said to be depth of that tree. In a tree depth of root node is 0.

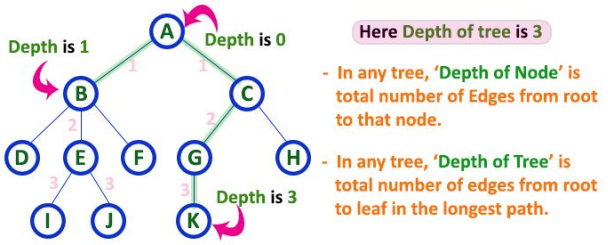

Figure 15: DIAGRAM TO DEMONSTRATE DEPTH

- **Path**

  In a tree data structure, the sequence of nodes and edges from one to another node is called path between that two nodes. Length of path is total number of nodes in that path.

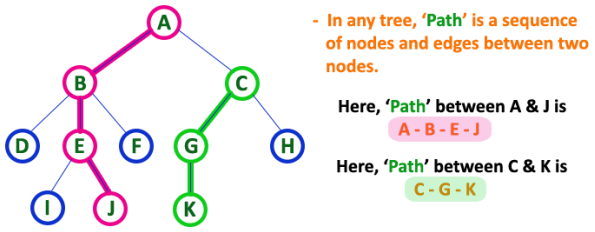

Figure 16: DIAGRAM TO DEMONSTRATE PATH

- **Sub tree**

  In a tree data structure, each child from a node forms a subtree recursively. Each child node will form a subtree on its parent node.

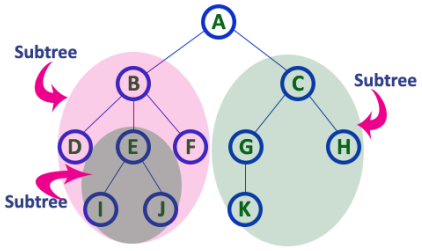

Figure 17: DIAGRAM TO DEMONSTRATE SUB TREE

- **Level**

  In a tree data structure, the root node is said to be of level 0 and the children of the root node are at level 1 and the children of the nodes which are at level 1 will be at level 2 and so on. In simple words, in a tree each step from top to bottom is called a level and the level count starts with 0 and incremented by one at each level.

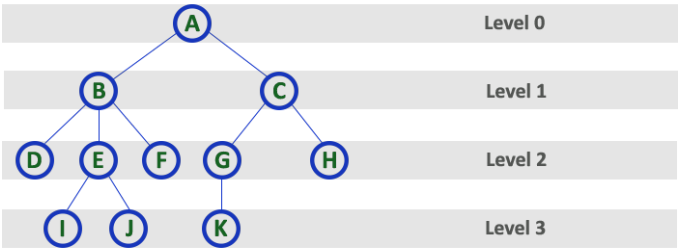

Figure 18: DIAGRAM TO DEMONSTRATE LEVEL

2.4.2 TYPES OF TREE:

- **Skewed tree**

  Skewed tree is a tree data structure where each parent node have only one node.

- **Binary tree**

  Binary tree is a special type of tree data structure in which every node can have a maximum of 2 children. One is known as left child and the other is right known as right child. A tree in which every node can have a maximum of two children is called as binary tree. In binary tree, every node can have either 0 children or 1 child or 2 children but not more than 2 children.

2.4.3 TYPES OF BINARY TREE:

- **Strictly binary tree**

  In a binary tree, every node can have a maximum of two children. But in strictly binary tree, every node should have exactly two children or none. That means every internal node must have exactly two children. A strictly binary tree is also called as full binary tree or proper binary tree.

- **Full binary**

In a full binary tree, every node can have exactly two children.and all leaf nodes are at the same level.

- **Complete binary**

In a binary tree, every node can have a maximum of two children. But in strictly binary tree, every node should have exactly two children or none and in complete binary tree all the nodes must have exactly two children and at every level of complete binary tree there must be 2^{level} number of nodes.

2.4.4 TYPES OF TREE TRAVERSAL:

- **Inorder traversal**

1. Traverse the left subtree in inorder.
2. Visit the root.
3. Traverse the right subtree in inorder.

```
void InOrder(struct BinaryTreeNode *root){
    if(root) {
        InOrder(root→left);
        printf("%d",root→data);
        InOrder(root→right);
    }
}
```

- **Preorder traversal**

1. Visit the root.
2. Traverse the left subtree in preorder.
3. Traverse the right subtree in preorder.

```
void PreOrder(struct BinaryTreeNode *root){
    if(root) {
        printf("%d",root→data);
        PreOrder(root→left);
        PreOrder (root→right);
    }
}
```

- **Postorder traversal**

1. Traverse the left subtree in postorder.
2. Traverse the right subtree in postorder.
3. Visit the root.

```
void PostOrder(struct BinaryTreeNode *root){
    if(root)        {
            PostOrder(root→left);
            PostOrder(root→right);
            printf("%d",root→data);
    }
```

2.4.5 BINARY SEARCH TREE:

A binary tree in which, for each node, the value of all the nodes in the left sub-tree is lesser or equal and values of all the nodes in right sub-tree is greater.

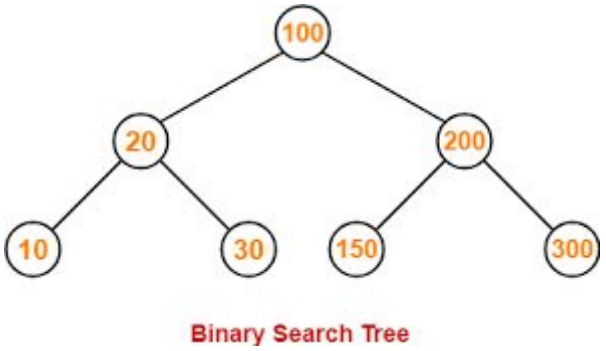

Figure 19: BINARY SEARCH TREE

For example in the above tree, the value of the root node (100) is greater than left node (20) and lesser than right node (200). As moving to the next level, 20 is greater than 10 and less than 30 and 200 is greater than 150 and less than 300. As looking at the level 0 and level 2 we can see root node value is greater than the far left node value i.e. 10 and lesser than the far right node value i.e. 300. Hence above tree is a binary search tree where all the values at the left are less than root node and all the values at right are greater than the root node.

2.4 GRAPHS:

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges. Formally, a graph is a pair of sets **(V, E)**, where **V** is the set of vertices and **E** is the set of edges, connecting the pairs of vertices.

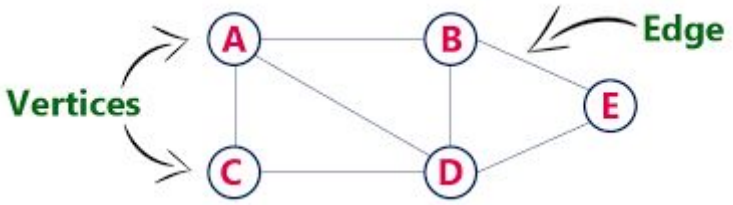

Figure 20: STRUCTURE OF A GRAPH

Chapter 3

3. SYSTEM DESIGN

3.1 DESIGN GOALS:

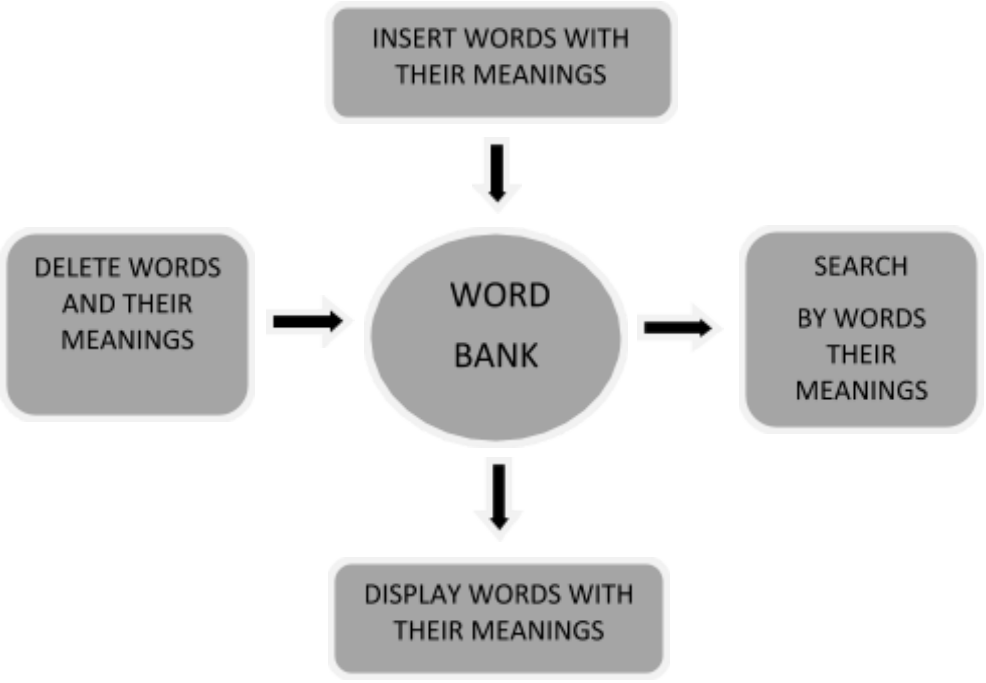

Figure 21: DIAGRAM FOR DATAFLOW

3.2 ALGORITHM:

AT INSERT:

- parent, current, and new node pointer are set to null
- root is checked
- if root is null, no node is created so create function is called where a node is created
- Dictionary is traversed and new word is compared with stored word
- If same duplicated entry is printed
- After traversing last node is set as current node
- A node is created and inserted using binary search method

AT SEARCH:

- Root is checked if null it is return back displaying directory empty
- Root data are copied to a temporary variable
- Word is taken from user and is compared with each word present
- If true, word and meanings are displayed
- And pointer value are selected using binary search method

AT DISPLAY:

- Root is checked if null no word present
- Else all the words are displayed inorder
- Traverse the left subtree in order
- Visit the root
- Traverse the right subtree in order.

AT DELETE:

- Root is checked, if null no word to display
- Current pointer is set to root
- Word asked to be deleted and every word in the dictionary are compared
- If present, parent is set to current and according to the compared value result in the current pointer is set
- If current pointer points to root then the word is deleted by changing root to null
- Else if the nodes first are checked if they are child node or parent node
- then accordingly the value of parent or root pointer is set to value of current pointer value and current is deleted

3.3 FLOWCHART:

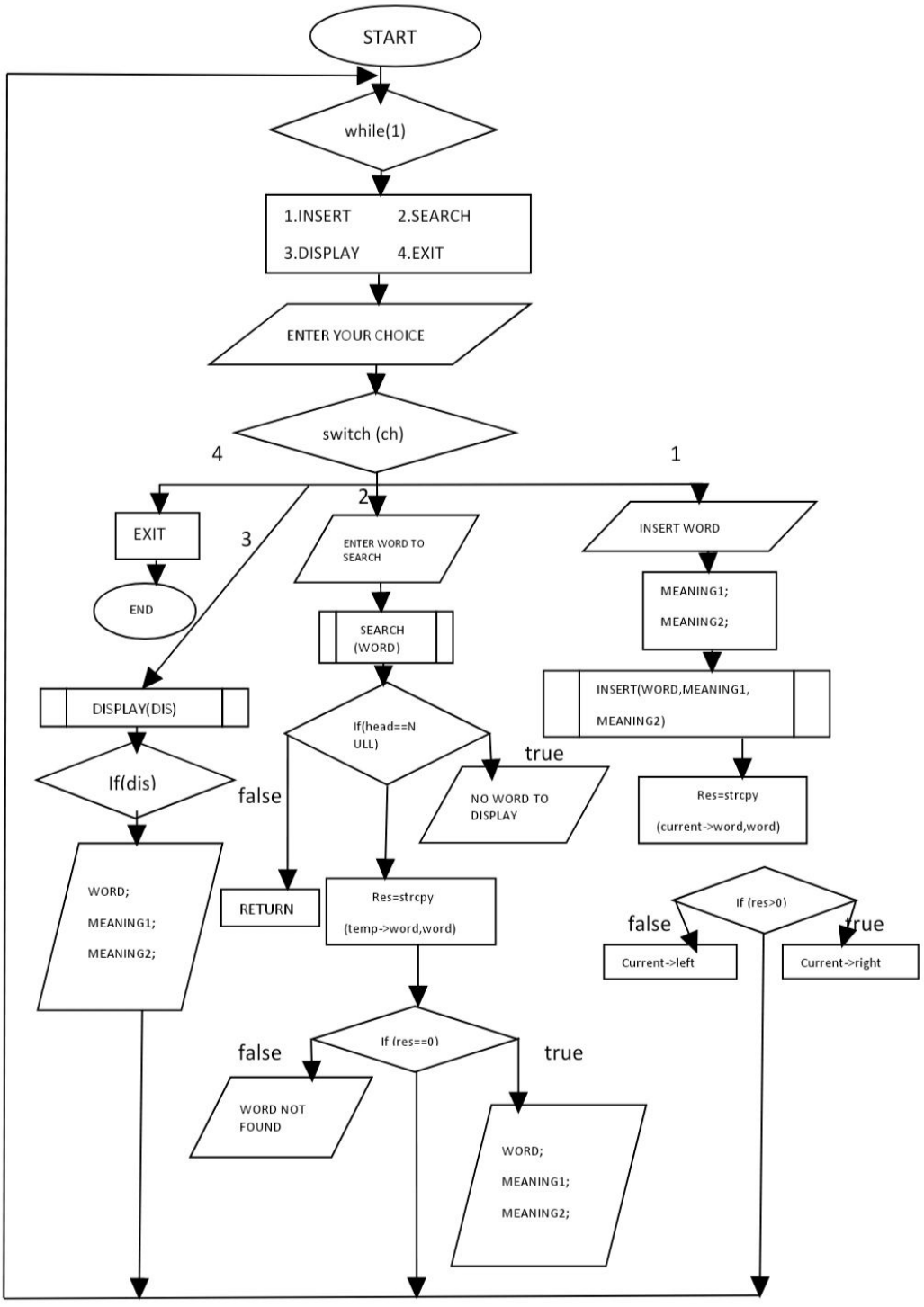

Figure 22: FLOWCHART

Chapter 4

4. IMPLEMENTATION

4.1 MODULE 1 FUNCTIONALITY:

INSERTION MODULE

In this program, insertion of word and its two meanings is done through an insert function inside which a create function is called new data i.e. new words and its meaning is entered by user. In other words, a node is created in the tree. The words are again passed to the insert function where the new node is added to the dictionary. Insertion is executed when the user, in the main function selects the insertion case. Insertion module, help the user to add words they want to insert in the dictionary with their meanings which later can be read. When inserted it performs binary search method.

4.2 MODULE 2 FUNCTIONALITY:

DELETION MODULE

In this program, deletion of word and its two meanings is done through a delete function. In main function, the user is asked for the word that he/she wants to delete from the memory. The word given is then passed to the delete function where the words are compared with each and every words present in the dictionary. If the words matches the given word then the word is deleted from the memory. In the program, words of matching node is deleted and the previous pointer is pointed to the value of the current pointer. Here when the word is deleted its meaning also is deleted thus frees the space.

4.3 MODULE 3 FUNCTIONALITY:

SEARCH MODULE

In this program, searching of word and its two meanings is done through a search function. In the main program, the user is asked for the word he/she wants to search. The word is then passed to the search function where the entered word is compared with each and every word present in the dictionary. If the word matches, the function return or displays the word with its two meanings. This makes easy for the user to search for a particular word's meaning.

4.4 MODULE 4 FUNCTIONALITY:

DISPLAY MODULE

In this program, display of word and its two meanings is done through a display function. In the main function, when the user asks to display, display function is called. In display function, the dictionary is traversed in inorder. It displays, every word with it's meanings in an alphabetical order. This function makes easier for the user to look at more number of words and their meanings at once. Display function displays all the content of the dictionary.

Chapter 6

CONCLUSION

This project helps to identify as something previously seen, known, etc. To identify from knowledge of characteristics. It provides option with word to insert, recognize that word by adding their meaning and another option is to display that word.

Also, we can search, what we had been entered. Using this program, we can delete words too.

Add	Meaning1(m1)	Meaning2(m2)	Display	Search	Delete
Love	Affection	Care	m1 m2	love	Null

Figure 1: EXAMPLE OF DATA

REFERENCES

For my project, the following references are useful to know for better understanding:

[1] Data structure using C

- Aaron M. Tananbaum
- Yedidyah Langham
- Moshe J. Augenstein

[2] www.youtube.com