



DICTIONARY USING BST

A MINI PROJECT REPORT

Submitted by

ABINAYA M (REG.NO:9517202109002)

BHUVANIKA S (REG.NO:9517202109011)

RAJA KUMARI S (REG.NO:9517202109042)

*In partial fulfillment for the award of the degree
of*

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

ANNA UNIVERSITY: CHENNAI 600 025

DECEMBER 2022

BONAFIDE CERTIFICATE

Certified that this project report “**DICTIONARY USING BST**” is the bonafide work of “**ABINAYA.M(9517202109002), BHUVANIKA.S(9517202109011), RAJAKUMARI.S(9517202109042)**” who carried out the mini project work under my supervision.

SIGNATURE

Dr. P.Thendral, M.E.,Ph.D

Assistant Professor(SG)
Artificial Intelligence and Data Science
Mepco Schlenk Engineering College
Sivakasi – 626 005
Virudhunagar District.

SIGNATURE

Dr.J.Angela Jennifa Sujana,M.E.,Ph.D

Associate Professor(SG) & Head
Artificial Intelligence and Data Science
Mepco Schlenk Engineering College,
Sivakasi – 626 005.
Virudhunagar District.

Submitted for the project viva-voce examination to be held on _____. .

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Dictionary can be implemented using binary search tree. A binary search tree is a binary tree such that each node stores a key of a dictionary. Key 'k' of a node is always greater than the keys present in its left sub tree. Similarly, key 'k' of a node is always lesser than the keys present in its right sub tree. The defining relationship between the key values for nodes in a bst. All nodes stored in the left subtree of a node whose key value is KK have key values less than or equal to KK . All nodes stored in the right subtree of a node whose key value is KK have key values greater than KK . A simple implementation for the dictionary ADT can be based on sorted or unsorted list. When implementing the dictionary with an unsorted list, inserting a new record into the dictionary can be performed quickly by putting it at the end of the list. However, searching an unsorted list for a particular record requires $\Theta(n)\Theta(n)$ time in the average case. For a large database, this is probably much too slow.

TABLE OF CONTENTS

CH.NO	TITLE	PG.NO
	ABSTRACT	3
1	INTRODUCTION	6
	1.1 Overview	
	1.2 Architecture	
	1.3 Application	
	1.4 issues	
2	LITERATURE REVIEW	11
	2.1 Implement dictionary functionality in c	
	2.2 Advantage	
	2.3 Disadvantage	
3	SYSTEM DESIGN	13
	3.1 description	
4	SYSTEM REQUIREMENT	14
	4.1 Hardware requirement	
	4.2 Software requirement	
5	SYSTEM IMPLEMENTATION	15
6	OUTPUT	19
7	CONCLUSION	34

ACKNOWLEDGEMENT

First and foremost we **praise and thank “The Almighty”**, the lord of all creations, who by his abundant grace has sustained us and helped us to work on this project successfully.

We really find unique pleasure and immense gratitude in thanking our respected management members, who is the backbone of our college.

A deep bouquet of thanks to respected Principal **Dr. S. Arivazhagan M.E.,Ph.D.**, for having provided the facilities required for our mini project.

We sincerely thank our Head of the Department **Dr. J. Angela Jennifa Sujana M.E.,Ph.D.**, Associate Professor(SG) & Head, Department of Artificial Intelligence and Data Science, for her guidance and support throughout the mini project .

We also thank our guide **Dr. P. Thendral M.E, Ph . d** Assistant Professor Department of Artificial Intelligence and Data Sciencefor their valuable guidance and it is great privilege to express our gratitude to them.

We extremely thank our project coordinator **Dr. P. Thendral ,M.E.,Ph.D.**, Assistant Professor(SG) Department of Artificial Intelligence and Data Science, who inspired us and supported us throughout the mini project.

We extend our heartfelt thanks and profound gratitude to all the faculty members of Artificial Intelligence and Data Science department for their kind help during our mini project work.

We also thank our parents and our friends who had been providing us with constant support during the course of the mini project work.

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Dictionary can be implemented using binary search tree. A binary search tree is a binary tree such that each node stores a key of a dictionary. Is there some way to organize a collection of records so that inserting records and searching for records can both be done quickly? We can do this with a BINARY SEARCH TREE (bst). The advantage of using the BST is that all major operations (insert, search, and remove). A binary tree that imposes the following constraint on its node values: The search key value for any node AA must be greater than the (key) values for all nodes in the left subtree of AA , and less than the key values for all nodes in the right subtree of AA . Some convention must be adopted if multiple nodes with the same key value are permitted, typically these are required to be in the right subtree. If we use a sorted array based list to implement the dictionary, then binary search can be used to find a record. However, insertion will now require $\Theta(n)$ time on average because, once the proper location for the new record in the sorted list has been found, many records might be shifted to make room for the new record. If the dictionary is implemented using trees, then each node will hold unique keys. Here for each node u in the tree Every key is $U.I$ is strictly smaller than $U.K$. a tree is organized. According to this invariant as referred to as a binary search Tree. one of the major advantage of this variant is that, sorted List of keys can be found in linear time using in-order Traversal. This can be defined recursively as follows – one empty Tree, do nothing, otherwise recurs on the left subtree first, take the root, and report it. Then recur to the right subtree. If the list is implemented using a linked list, then no speedup to the search operation will result from storing the records in sorted order. On the other hand, if we use a sorted array based List to implement the dictionary, then binary search can be used to find a record. A simple implementation for the dictionary ADT can be based on sorted or unsorted list. When implementing dictionary with an unsorted list, inserting a new record into the dictionary can be performed quickly by putting it at the end of the list. However, searching an unsorted list for a particular record requires $\Theta(n)$ time in the average case. For a large

database, this is probably much too slow. Alternatively, the records can be stored in a sorted list. If the list is implemented using a linked list, then no Speedup to the search operation will result from storing the records in sorted order

1.2 ARCHITECTURE

BST is a collection of nodes arranged in a way where they maintain BST properties. Each node has a key and an associated value. While searching, the desired key is compared to the keys in BST and if found, the associated value is retrieved. Following are the basic operations of a tree – Searches an element in a tree, Inserts an element in a tree, Traverses a tree in a pre-order manner, Traverses a tree in an in-order manner, Traverses a tree in a post-order manner.

Whenever

n element is to be searched, start searching from the root node. Then if the data is less than the key value, search for the element in the left subtree. Otherwise, search for the element in the right subtree. Whenever an element is to be inserted, first locate its proper location. Start searching from the root node, then if the data is less than the key value, search for the empty location in the left subtree and insert the data. The value of the key of the left sub-tree is less than the value of its parent (root) node's key. The value of the key of the right sub-tree is greater or equal to the value of its parent (root) node's key.

1.3 APPLICATION

1) A Self-Balancing Binary Search Tree is used to maintain sorted stream of data.

For example, suppose we are getting online orders placed and we want to maintain the live data (in RAM) in sorted order of prices. For example, we wish to know number of items purchased at cost below a given cost at any moment. we wish to know number of items purchased at higher cost than given cost.

2) A Self-Balancing Binary Search Tree is used to implement doubly ended priority queue. With a Binary Heap, we can either implement a priority queue with support of extract Min () or with extract Max (). If we wish to support both the operations, we use a Self-Balancing Binary Search Tree to do both in $O(\log n)$

3) There are many more algorithm problems where a Self-Balancing BST is the best suited data structure, like count smaller elements on right, smallest greater on right side.

4) When data is stored in different nodes and arranged in a pattern, it is easy to remember the organized structure of data and this is the main advantage of BST. We can arrange the data in hierarchical fashion and make it useful for different aspects so that it explains the structural relationships between different nodes and how they are correlated with each other. While considering a particular dataset, it is taken as BST so that left and right nodes are arranged which depicts the relationship between the nodes and gives solution to any particular algorithm in the structure.

5) BSTs are used for indexing. It is also used to implement various searching algorithms. It can be used to implement various data structures.

6) BSTs are used for indexing in databases. It is used to implement searching algorithms. BSTs are used to implement Huffman coding algorithm. It is also used to implement dictionaries.

7) helps student transition from classroom to practice, where they can become of the health care team.

1.4 ISSUES

1)Malware – short for malicious software which is specifically designed to disrupt, damage, or gain authorized access to a computer system. Much of the malware out there today is self-replicating: once it infects one host, from that host it seeks entry into other hosts over the Internet, and from the newly infected hosts, it seeks entry into yet more hosts. In this manner, self- replicating malware can spread exponentially fast.

Virus – A malware which requires some form of user’s interaction to infect the user’s device. The classic example is an e-mail attachment containing malicious executable code. If a user receives and opens such an attachment, the user inadvertently runs the malware on the device.

2)Worm – A malware which can enter a device without any explicit user interaction. For example, a user may be running a vulnerable network application to which an attacker can send malware. In some cases, without any user intervention, the application may accept the malware from the Internet and run it, creating a worm.

3)Botnet – A network of private computers infected with malicious software and controlled as a group without the owners’ knowledge, e.g. to send spam.

4)Vulnerability attack: This involves sending a few well-crafted messages to a vulnerable application or operating system running on a targeted host. If the right sequence of packets is sent to a vulnerable application or operating system, the service can stopper, worse, the host can crash.

5) Bandwidth flooding: The attacker sends a deluge of packets to the targeted host—so many packets that the target’s access link becomes clogged, preventing legitimate packets from reaching the server.

6) Connection flooding: The attacker establishes a large number of half-open or fully open TCP connections at the target host. The host can become so bogged

down with these bogus connections that it stops accepting legitimate connections.

7)DDoS (Distributed DoS) – DDoS is a type of DOS attack where multiple

Compromised systems, are used to target a single system causing a Denial of Service (DoS) attack. DDoS attacks leveraging botnets with thousands of comprised hosts are a common occurrence today. DDoS attacks are much harder to detect and defend against than a DoS attack from a single host.

8)Packet sniffer – A passive receiver that records a copy of every packet that flies by is called a packet sniffer. By placing a passive receiver in the vicinity of the wireless transmitter, that receiver can obtain a copy of every packet that is transmitted! These packets can contain all kinds of sensitive information, including passwords, social security numbers, trade secrets, and private personal messages. some of the best defenses against packet sniffing involve cryptography.

9)IP Spoofing – The ability to inject packets into the Internet with a false source address is known as IP spoofing, and is but one of many ways in which one user can masquerade as another user. To solve this problem, we will need end-point authentication, that is, a mechanism that will allow us to determine with certainty if a message originates from where we think it does.

CHAPTER 2

PROBLEM STATEMENT

2.1) IMPLEMENT DICTIONARY FUNCTIONALITY IN C

Generally, the C standard library does not include a built-in dictionary data structure, but the POSIX standard specifies hash table management routines that can be utilized to implement dictionary functionality. Namely, `h create`, `h search` and `h destroy` provide the features like creating a hash table, inserting items into it, searching the item in a table, and deallocating the whole data structure. Even though the implemented functionality is the bare minimum, it can solve many problem scenarios. At first, the `h create` function needs to be called to create a hash table with a given maximum number of elements, which is passed as the only argument. Note that capacity can't be modified after the `h create` call; thus, the user needs to factor this issue into the code structure. Once the table is created, the `h search` function can be called to add items into it. It takes 2 arguments; the first one is of type `ENTRY`, defined as struct of `char*` for the key and `void*` for corresponding data. The second parameter is of type `ACTION`, which is essentially an enum consisting of two elements, `FIND` and `ENTER`. The latter values are used to indicate what operation to conduct on the table. The `h destroy` function is used to free hash table memory, but it does not free the buffers in the `ENTRY` objects, which is usually created by the user. Thus, if pointers in these objects point to dynamic memory, the caller is responsible for keeping the handles for them and deallocating before the program exit. Mind that individual entries cannot be deleted from the table.

ADVANTAGE:

- 1) Searching is very efficient we have all the nodes of bst in a Specific order, hence searching for a particular item is very Efficient and faster. This is because we need not search the Entire tree and compare all the nodes
- 2) Efficient working when compared to arrays and linked list
- 3) Insert and delete are faster
- 4) BSTs are easy to implement compared to hashing, we can easily Implement Our own customized BST. To implement Hashing, We generally rely on libraries provided by programming languages

DISADVANTAGE:

- In binary tree traversals, there are many pointers that are null and hence useless.
- The access operation in a Binary Search Tree (BST) is slower than in an array.
- A basic option is dependent on the height of the tree.
- Deletion node not easy.
- A basic option is based on the height of tree.

CHAPTER 3

3.1) DESCRIPTION

The binary search tree is an advanced algorithm used for analyzing the node, its left and right branches, which are modeled in a tree structure and returning the value. The BST is devised on the architecture of a basic binary search algorithm; hence it enables faster lookups, insertions, and removals of nodes. This makes the program really fast and accurate. Digital dictionary of Computer and Network Engineering can be used in the learning process of computer and network engineering students. Dictionaries are generally book-shaped is hard to carry because of their thick and heavy, now can be accessed anywhere by development of web technology. Digital dictionaries can be accessed through computers, laptops dan cellphones this is make student easily in learning process. The digital dictionary can not only be used by students of computer and network engineering, but can also by general public, as long as you have an internet connection, anyone can be use this digital dictionaries. The main process in this digital dictionary is the search process. Binary search algorithm used in the search process of binary search algorithm. Binary search algorithm search is applied to word search in this digital dictionary, because this algorithm is intended for sequential data

CHAPTER 4

SYSTEM REQUIREMENT

4.1) HARDWARE REQUIREMENT

COMPUTER WITH WINDOWS OPERATING SYSTEM

DEV C++

WIFI MODULE

LAPTOP

4.2) SOFTWARE REQUIREMENT

DEV C++

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>

struct dict
{
    struct dict *left,*right;
    char word[10],meaning[20];
}*Root=NULL;

typedef struct dict dictionary;
int check(char[],char[]);
void insert(dictionary *);
void Search();

void view(dictionary *);

int check(char a[],char b[])
{
    int i,j,c;
    for(i=0,j=0 ; a[i]!='\0'&&b[j]!='\0' ; i++,j++)
    {
        if(a[i]>b[j])
        {
            c=1;
            break;
        }
        else if(b[j]>a[i])
        {
            c=-1;
            break;
        }
        else
            c=0;
    }
}
```

```

    }
    if(c==1)

        return 1;
    else if(c==-1)

        return -1;

    else
        return 0;
    }
    void Search()
    {
        int flag=0;
        dictionary *ptr;
        ptr=Root;
        char w[10];
        printf("\nEnter word");
        scanf("%s",w);
        while(ptr!=NULL && flag==0)
        {
            if(check(w,ptr->word)>0)
                ptr=ptr->right;
            else if(check(w,ptr->word)<0)
                ptr=ptr->left;
            else if(check(w,ptr->word)==0)
            {
                flag=1;
                printf("\n%s",ptr->meaning);
            }

        }
        if(flag==0)
            printf("\nWord not found");
    }

    void insert(dictionary *temp)
    {

        int flag=0;
        dictionary *ptr,*par;
        ptr=Root;

        if(Root==NULL)
            Root=temp;
    }

```



```

else
{
    while(ptr!=NULL)
    {
        if(check(temp->word,ptr->word)>0){
            par=ptr;
            ptr=ptr->right;
        }

        else if(check(temp->word,ptr->word)<0)
        {
            par=ptr;
            ptr=ptr->left;
        }
        else if(check(temp->word,ptr->word)==0)
        {
            flag=1;
            printf("\nWord exists!!");
            break;
        }
    }
}
if(flag==0 && ptr==NULL)
{
    if(check(par->word,temp->word)==1)
        par->left=temp;
    else if(check(par->word,temp->word)==-1)
        par->right=temp;
}

}

}

void view(dictionary *ptr)
{
    if(Root==NULL)
        printf("\nEmpty dictionary\n");

    else if(ptr !=NULL)
    {
        view(ptr->left);

        printf("\nWord:%s\n",ptr->word);
    }
}

```

```

        printf("\nMeaning:%s\n",ptr->meaning);

        view(ptr->right);
    }

}

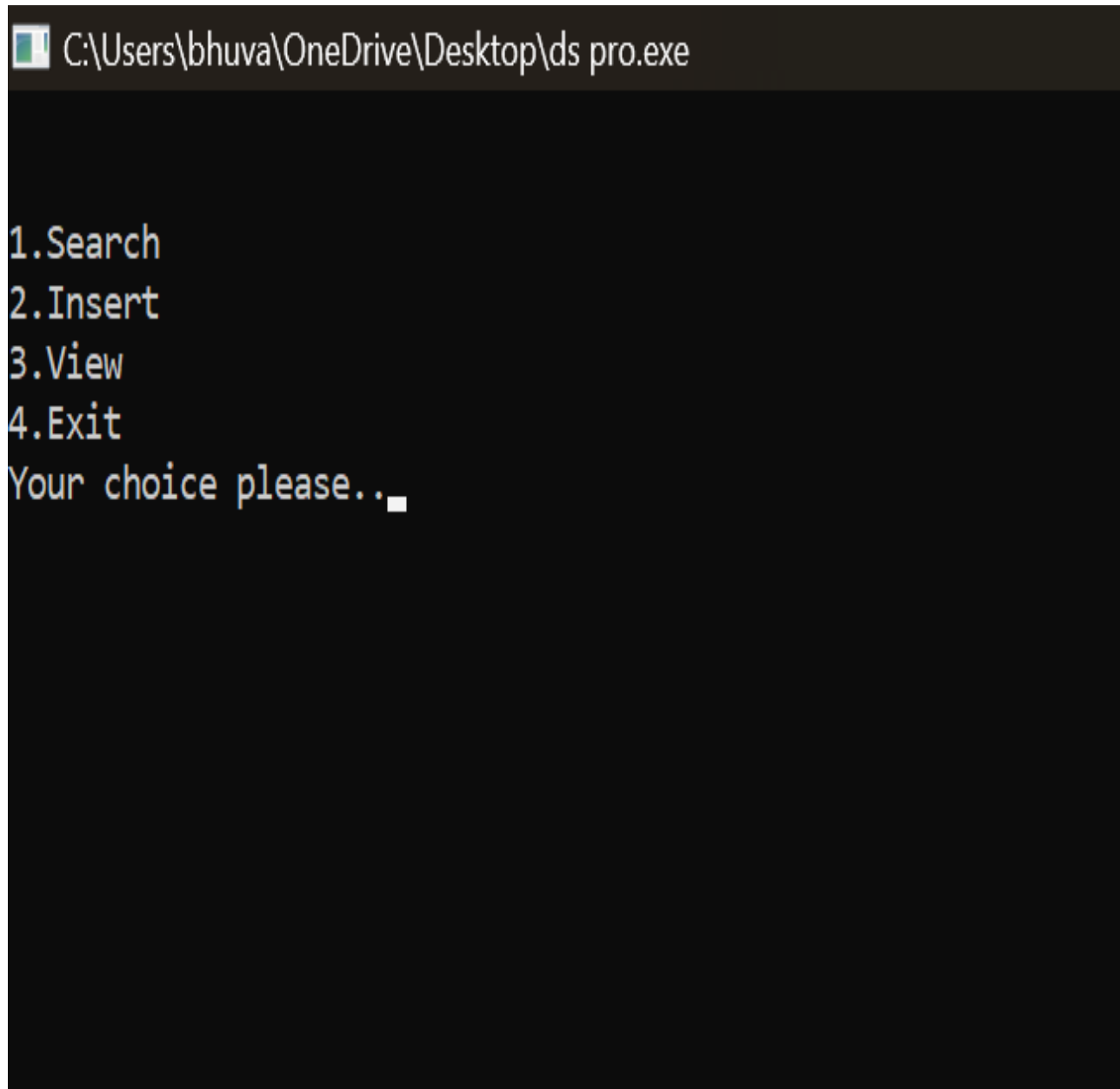
int main(int argc, char const *argv[])
{
    int ch;
    dictionary *temp;
    while(ch!=4)
    {
        printf("\n1.Search\n2.Insert\n3.View\n4.Exit\nYour choice please..");
        scanf("%d",&ch);
        switch (ch)
        {
            case 1: Search();
                    break;
            case 2:
                    temp=(dictionary*)malloc(sizeof(dictionary));
                    temp->left=NULL;
                    temp->right=NULL;
                    printf("\nInsert word:\n");
                    scanf("%s",temp->word);
                    printf("\nInsert meaning:\n");
                    scanf("%s",temp->meaning);
                    insert(temp);
                    break;
            case 3:
                    view(Root);
                    break;
            case 4:
                    exit(0);
        }
    }

    return 0;
}

```

CHAPTER 6

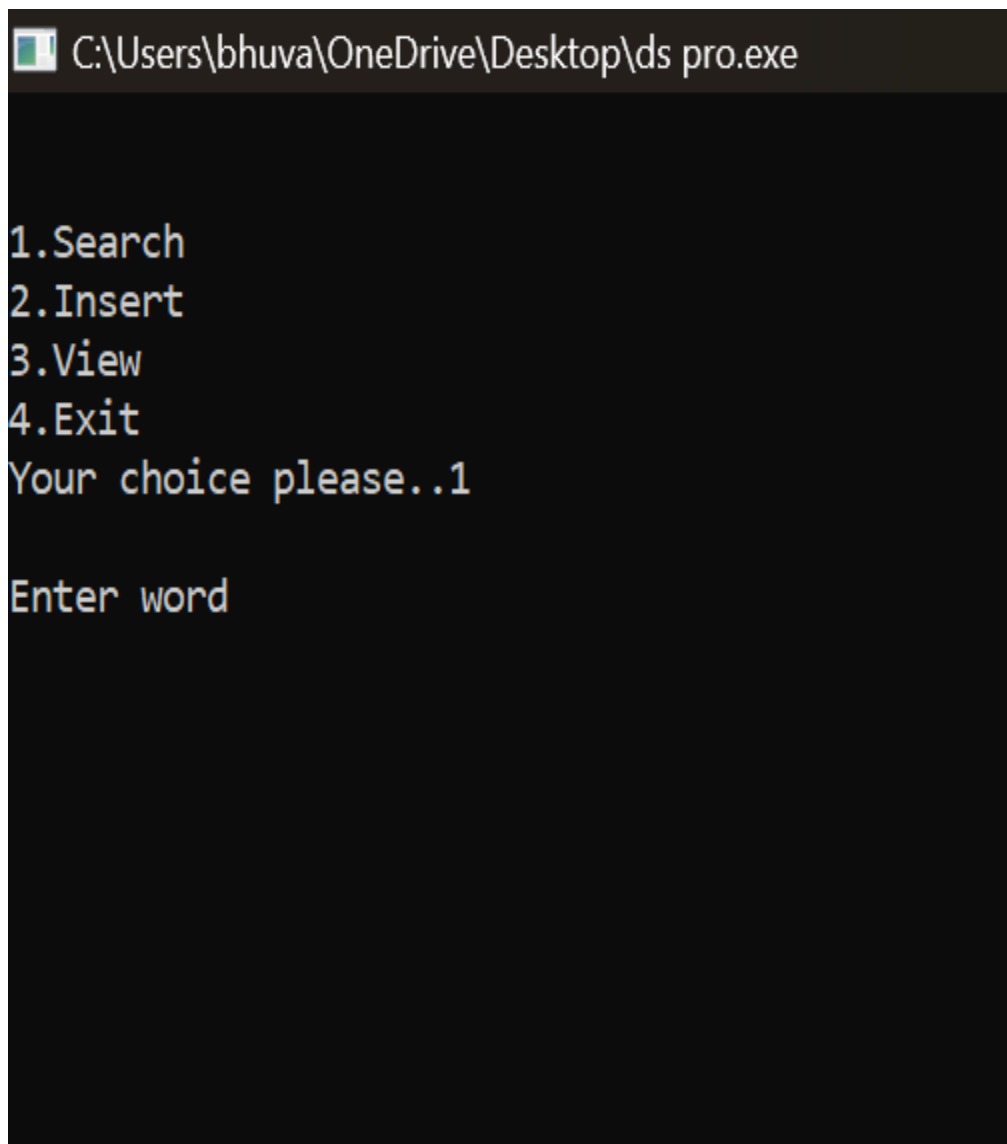
OUTPUT



```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..
```

Figure no.6.1 Result 1

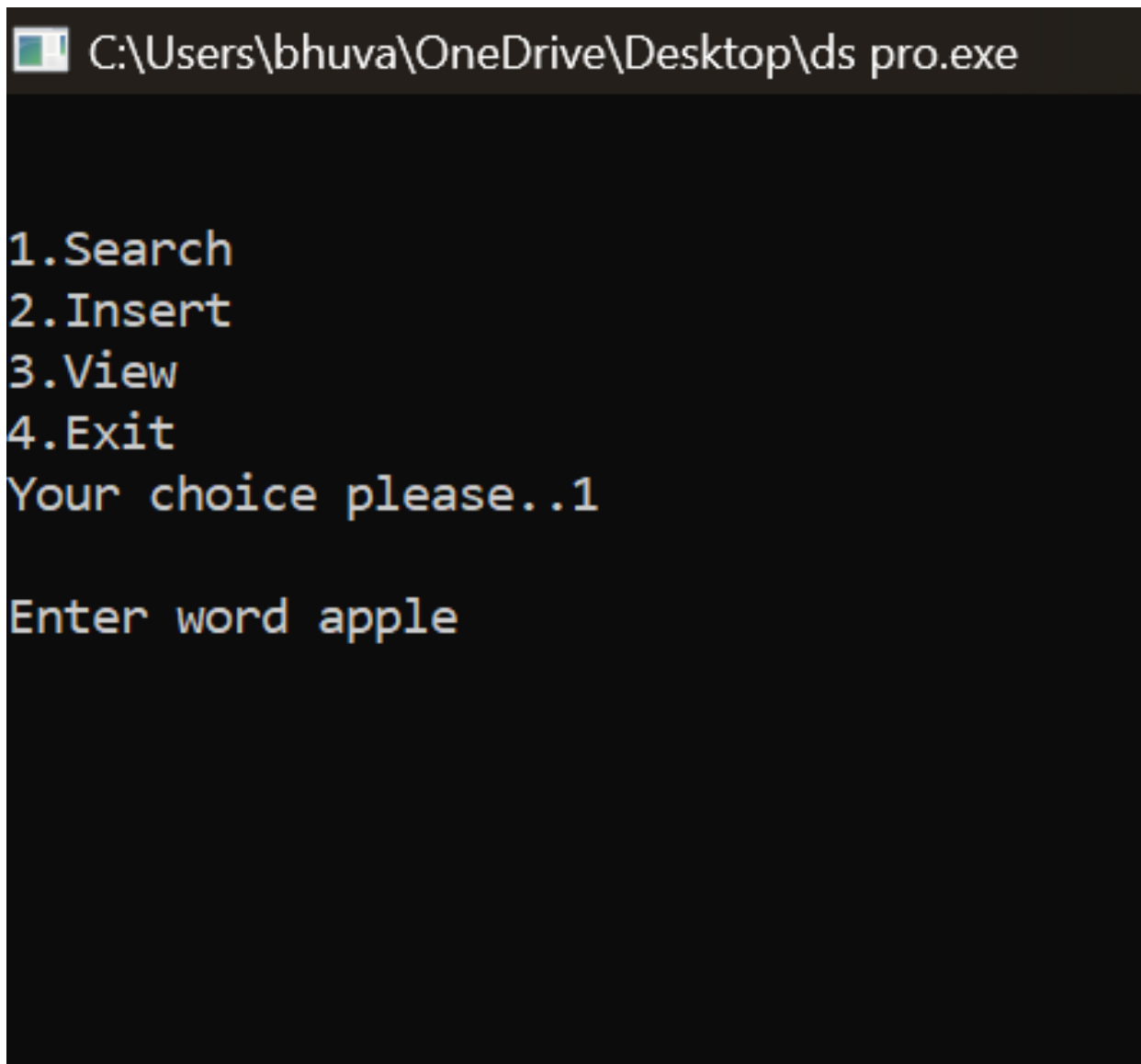


```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word
```

Figure no. 6.2 Result 2

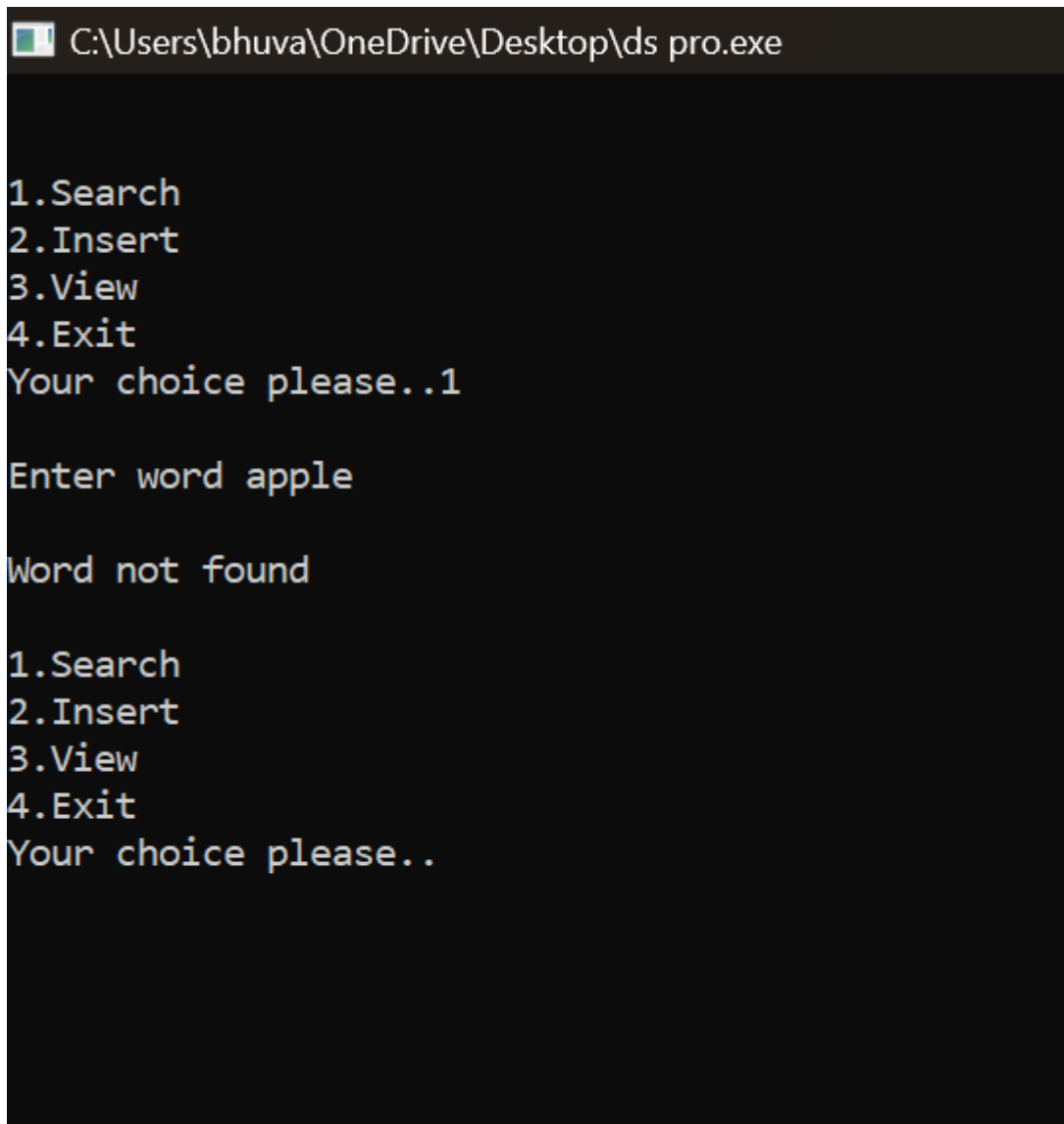


```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word apple
```

Figure no. 6.3 Result 3



```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word apple

Word not found

1.Search
2.Insert
3.View
4.Exit
Your choice please..
```

Figure no. 6.4 Result 4

```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

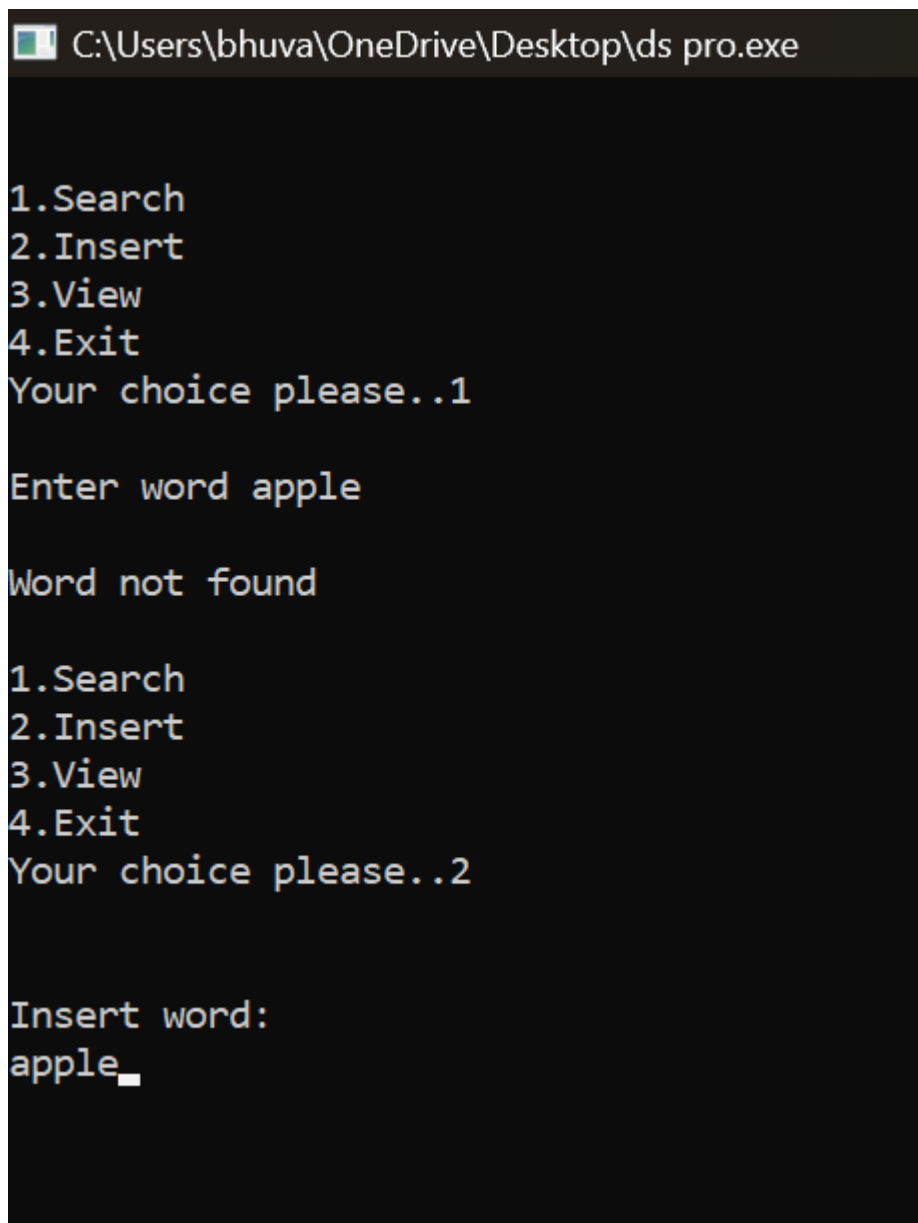
Enter word apple

Word not found

1.Search
2.Insert
3.View
4.Exit
Your choice please..2

Insert word:
_
```

Figure no. 6.5 Result 5



```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word apple

Word not found

1.Search
2.Insert
3.View
4.Exit
Your choice please..2

Insert word:
apple_
```

Figure no. 6.6 Result 6


```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word apple

Word not found

1.Search
2.Insert
3.View
4.Exit
Your choice please..2

Insert word:
apple

Insert meaning:
fruit

1.Search
2.Insert
3.View
4.Exit
Your choice please.._
```

Figure no. 6.7 Result 7

```
1.Search
2.Insert
3.View
4.Exit
Your choice please..2
```

```
Insert word:
red
```

```
Insert meaning:
colour
```

```
1.Search
2.Insert
3.View
4.Exit
Your choice please..2
```

```
Insert word:
new
```

```
Insert meaning:
recent
```

```
1.Search
2.Insert
3.View
4.Exit
Your choice please..2
```

Figure no. 6.8 Result 8

```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..2

Insert word:
brave

Insert meaning:
bold

1.Search
2.Insert
3.View
4.Exit
Your choice please..2

Insert word:
target

Insert meaning:
aim

1.Search
2.Insert
3.View
4.Exit
Your choice please..unnecessary

Insert word:

Insert meaning:
needless
```

Figure no. 6.9 Result 9

```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..discourage

Insert word:

Insert meaning:
depress

1.Search
2.Insert
3.View
4.Exit
Your choice please.._
```

Figure no. 6.10 Result 10

```
1.Search
2.Insert
3.View
4.Exit
Your choice please..3

Word:apple

Meaning:fruit

Word:brave

Meaning:bold

Word:discouragedepress

Meaning:depress

Word:new

Meaning:recent

Word:red

Meaning:colour

Word:target

Meaning:aim

Word:unnecessarneedless

Meaning:needless
```

Figure no. 6.11 Result 111

```
1.Search
2.Insert
3.View
4.Exit
Your choice please..3

Word:apple
Meaning:fruit

Word:brave
Meaning:bold

Word:discouragedepress
Meaning:depress

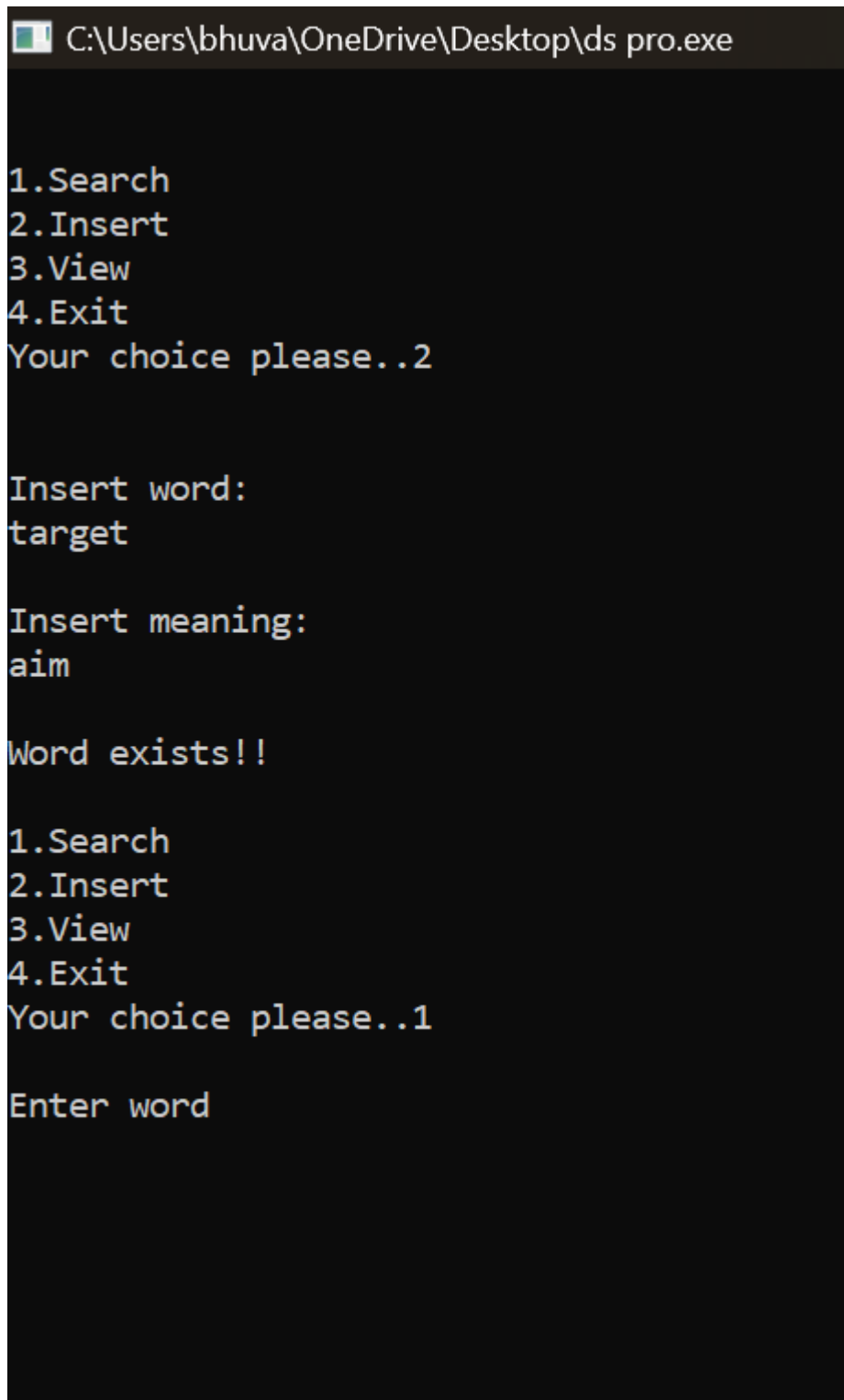
Word:new
Meaning:recent

Word:red
Meaning:colour

Word:target
Meaning:aim

Word:unnecessarneedless
Meaning:needless
```

Figure no. 6.12 Result 12



```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe

1.Search
2.Insert
3.View
4.Exit
Your choice please..2

Insert word:
target

Insert meaning:
aim

Word exists!!

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word
```

Figure no. 6.13 Result 13

```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe
1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word discourage

depress

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word apple

fruit

1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word target

aim
```

Figure no. 6.14 Result 14


```
C:\Users\bhuva\OneDrive\Desktop\ds pro.exe
fruit
1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter word target

aim
1.Search
2.Insert
3.View
4.Exit
Your choice please..1

Enter wordcar

Word not found

1.Search
2.Insert
3.View
4.Exit
Your choice please..4

-----
Process exited after 790.1 seconds with return value 0
Press any key to continue . . .
```

Figure no. 6.15 Result 15

CONCLUSION

From the results of the discussions that have been carried out, it can be concluded follows:

- a. The search process on the “Kamus Digital Teknik computer dan Jaringan” web-based built using binary search algorithms works well.
- b. The Binary Search Algorithm successfully finds every word that is searched if the word is available in the dictionary database.
- c. The basic principle of the binary search algorithm is to repeat the search space repeatedly until data is found or until the search space cannot be shared (data may not exist).
- d. The purpose of searching using a binary search algorithm is to reduce the number of operations that must be compared between the data sought and data in the database, especially for large amounts of data.
- e. This digital dictionary is built to facilitate students, especially computer and network engineering students in finding foreign computer vocabulary without having to carry a dictionary in book form everywhere.
- f. This digital dictionary is also made to make it easier for people to find computer vocabulary that they don't understand.
- g. This Digital Dictionary can be easily used by users who want to find computer vocabulary.