### MiniprojectReport

**on**

“DictonaryCreationusingDataStructure”

By

### PrathameshTiparkar

UndertheGuidanceof

### Prof.ManjiriRaut



**DEPARTMENTOFCOMPUTERENGINEERING**

ShalakaFoundation’sKeystoneSchoolofEngineeringNear Handewadi Chowk, Pune- 412308

SAVITRIBAIPHULEPUNEUNIVERSITY

### 2024-25

A

### MiniProjectReporton

“DictonaryCreation using Data Structure”is Successfully Completed

by

### PrathameshTiparkar



**DEPARTMENTOFCOMPUTERENGINEERING**

Shalaka Foundation’sKeystoneSchoolofEngineeringNear Handewadi Chowk, Pune-412308

SAVITRIBAIPHULEPUNEUNIVERSITY

### 2024-25

**Prof.ManjiriRaut Prof.SagarRajebhosale**

ProjectGuide HOD

Shalaka Foundation’sKeystoneSchoolofEngineering

**Departmentof ComputerEngineering**



### Certificate

This is to certify that the Project entitled"Dictonary Creation using Data Structure"Submittedby**PrathameshTiparkar**isarecordofbonafiedworkcarriedoutbyhim,inthepartialfulfilmentofthe requirementsfortheawardof Degree ofBachelor ofEngineering(Computer Engineering) at Keystone School of Engineering, Pune under the Savitribai Phule Pune University. This work is done during year 2024-2025.

### Prof.ManjiriRaut Prof.SagarRajebhosale

Guide HOD

DepartmentofComputerEngineering DepartmentofComputerEngineering

Dr.SandeepKadam

**Principal**

# ACKNOWLEDGEMENT

It takes immense pleasure in presenting the complete Mini-Project report on"DictonaryCreation using Data Structure". I extend my deepest gratitude to my guide,**Prof. Manjiri Raut**for their unwavering support and guidance. I also wish to thank my Head of Department,**Prof. Sagar Rajebhosale**, for his invaluable insights. Myheartfelt appreciation goes to**Dr. Sandeep Kadam**, Principal of my institution, and**Prof. Y.R. Soman**, Director, for their encouragement and cooperation throughouttheMini-Project.ThisMini-Project wouldnothave been possiblewithout the supportof myfaculty members, dear parents, and friends.

PrathameshTiparkar

**ABSTRACT**

This Python program serves asa dictionarymanagement tool thatenables users to store, retrieve, and display word-related data such as meanings and synonyms. The core functionality is built aroundahashtableusing**linearprobing**forcollisionresolution.Itfeaturesapre-definedstructure toholdwordsalongwiththeirrespectivehashvalues,meanings,andsynonyms,storedinparallel arrays.Userscaninteractwiththesystemthroughacommand-lineinterfacetoinsertnewentries, search for existing words using their hash values, and visualize the entire table. Additionally, the program allows data export in CSV, DOCX, and TXT formats for enhanced usability. This tool simulates the working of a dictionaryusing essential data structure concepts like hashing, arrays, andfilehandling,andhelpsusersunderstandhowdatacanbeefficientlyorganized,accessed,and managed using linear probing in hash tables.

**Keywords**:-Hashing,linearprobing,dictionarydata,wordstorage,hashtable,collisionresolution, data insertion, file export, parallel arrays, data structure implementation.

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTERS** | **Topic** | | **PageNo.** |
| **Chapter-1** | **INTRODUCTION** | |  |
|  | 1.1 | Introduction | 1 |
| **Chapter-2** | **PROBLEMDEFINITIONANDOBJECTIVES** | |  |
|  | 2.1 | ProblemDefinition | 2 |
|  | 2.2 | Objectives | 2 |
| **Chapter-3** | **METHODOLOGYANDREQUIREMENTS** | |  |
|  | 3.1 | Methodology | 3 |
|  | 3.2 | Requirements | 4 |
|  | 3.2.1 | FunctionalRequirement | 4 |
|  | 3.2.2 | Non-FunctionalRequirements | 5 |
| **Chapter-4** | **ADVANTAGESANDDISADVANTAGES** | |  |
|  | 4.1 | Advantages | 6 |
|  | 4.2 | Disadvantages | 6 |
| **Chapter-5** | **RESULTANDOUTPUT** | |  |
|  | 5.1 | Result&Output | 7 |
|  | **CONCLUSION** | | 12 |
|  | **REFERENCES** | | 13 |

**CHAPTER 1INTRODUCTION**

**CHAPTER2**

**PROBLEMDEFINITION ANDOBJECTIVES**

**CHAPTER 3 METHODOLOGYANDREQUIREMENTS**

**CHAPTER 4 ADVANTAGESANDDISADVANTAGES**

**CHAPTER 5 RESULTANDOUTPUT**

## Introduction

Modernapplications oftenusedatastructuresto manageand organizeinformation efficiently. Thisproject presentsaPython-basedimplementationofadictionarysystemusinghashingwithlinearprobing.Itfeatures apredefinedstructureforstoringwords,alongwiththeirmeaningsandsynonyms,usingahashtable.Users can insert new entries, retrieve stored data, and search for specific words using hash values. The program handles collisions through linear probing and provides a command-line interface for user interaction. Additionally, it supports exporting the dictionary data into files formats, showcasing practical use of data structures like arrays, hashing, and file handling in real-world applications.

## Keyelementsofthisprograminclude:

* + - **Data Storage and Lookup:**A hash table is used to efficiently store and retrieve words along with their meanings and synonyms using hashing with linear probing.
    - **CollisionHandling:**Linearprobingisimplementedtoresolvecollisionswhenmultiplewordshash to the same index in the table.
    - **Case-Insensitive Handling:**User inputs for words are managed effectively, allowing consistentstorage and search regardless of input case.
    - **FileExport Support:**Theprogramallows exporting the dictionarydata into files formats for easy sharing and offline access.

This program demonstrates fundamental concepts of Python data structures, hashing techniques, and file handling, offering a practical example of how programming can be used to build efficient, interactive dictionary systems that enhance data accessibility and user interaction.

## ProblemDefinition

Theproblemis tocreatea simple text-based dictionarysystemthat:

* + - **Takesuserinput**foraword,alongwithitsmeaningandsynonym.
    - **Validatesandstorestheinpu**tusingahashingtechniquewithlinearprobing.
    - **Displaystheword**detailsalong withtheircorrespondinghashvalues.
    - **Providesoptions**tosearch fora wordbased onits hashvalue.
    - **Allowsexporting**ofthestoreddictionarydataintoCSV,DOCX,andTXTformatsforexternaluse.

## Objectives

Themainobjectivesoftheprogramareto:

1. **Simulate a Dictionary System:**Provide a tool for users to insert words along with their meanings and synonyms using a hash table.
2. **ImplementHashingwithCollisionHandling:**Usehashingwithlinearprobingtoefficientlystore and retrieve word data.
3. **EnableWordSearch:**Allowuserstosearchforspecificwordsusingtheirhashvaluesanddisplay detailed information.
4. **SupportFileExport:**GiveuserstheabilitytoexportthedictionarycontentintoCSV,DOCX,and TXT formats.
5. **CreateaUser-FriendlyInterface:**Utilizesimpleconsole-basedmenusandcoloredoutputto improve the user experience.

This project, overall, simulates a basic dictionary management system, providing an introduction to data structures such as hash tables and arrays, along with linear probing for collision resolution. The program utilizesPython’sstandardlibrariesforstringhandling,datastorage,fileoperations,andinteractiveconsole input/output, making itan effective example of how fundamentaldata structure concepts can be applied to real-world scenarios.

## Methodology

Themethodologytocreate thisdictionarysysteminvolvesthefollowingkeysteps:

### DataCollectionandStorage:

* + Storewords,meanings,andsynonymsforasetofpredefinedlocationsinahashtable,which providesfastaccessbasedonwordhashvalues.Thekeysarethehashvalues,andthevalues are the corresponding words, meanings, and synonyms.
  + The choice of hash table ensures quick lookup of words when a user inputs a hash value, allowing for efficient retrieval of corresponding details.

### UserInput andProcessing:

* + Prompt the user to enter a word along with its meaning and synonym, which allows for aloop to handle multiple words.
  + In each iteration, take the user’sinput for a word. Convert the word to lowercase using ahelper function to ensure case-insensitive matching.
  + Implement a backtracking mechanism by checking for a special input value,"back,"which allows the user to go to the last entered word by popping it from a stack.

### WordLookup:

* + Afterreceiving theuser’s input,checkif thewordexists inthe hashtable.Iffound, retrieve its meaning and synonym from the table.
  + Ifnotfound, notifythe userthattheword isunavailable in thedictionary.

### DisplayingWordDetailsandExporting:

* + Ifthewordexists,displayitsmeaningandsynonymto theuser.
  + Prompttheusertoaskifthey wanttoexportthedata.Iftheychoose"yes,"theprogram constructs the data and saves it in the selected format.

1. **Navigation History Management:**Use a**stack**data structure to manage a history of user inputs, enablinguserstogobacktopreviouslyvisitedwords.Eachtimetheuserentersanewword,pushit ontothestack.Whentheusertypes"back,"popthetopentryfromthestackandretrievetheprevious word details.

## Requirements

### FunctionalRequirements

* + - 1. **UserInputHandling:**
         * Prompttheusertospecifyhowmanywordstheywishtothenavigationthroughtheprocess.
         * Acceptaword inputfromthe userduring eachiteration.
         * Allowtheuser totype"back"toreturn to theprevious wordand alsoisefficientin itswork

### DataStorage:

* + - * + Storepredefinedwordsandtheircorrespondingmeaningsandsynonymsina**hashtable**

efficientlookups.

### DisplayandNavigation:

* + - * + Displaythemeaningandsynonymoftheselectedword.
        + Provideanoption toexportthedatain the selectedformat.

### NavigationHistoryManagement:

* + - * + Keep track of previously entered words using a stack. Allow the user to go back to the last entered word by typing"back."

### Case-InsensitiveWordSearch:

* + - * + Converttheuser'sinputtolowercasefor case-insensitive matchingwiththestoredwords.

### ProgramTermination:

* + - * + Exitthe programafter displayinga closingmessagethanking theuser.

### Non-FunctionalRequirements

* + - 1. **Performance:**
         * The**unordered\_map**data structure ensures quick lookups, making word searches efficient even as the number of dictionary entries increases.

### PlatformCompatibility:

* + - * + The program is designed to be compatible with systems that support Python and standardlibraries. For specific file formats the system should have the necessary software installed.

### UserExperience:

* + - * + Usecoloredtextoutputforbetter readabilityand amoreuser-friendlyinterface.
        + Provideclearprompts,errormessages,andfeedback.

### Scalability:

* + - * + Thesystemisscalablebysimplyaddingmorewordsandtheirrespectivedatatothe

**unordered\_map**,withoutsignificantchanges tothe programstructure.

### ErrorHandling:

* + - * + Handleinvalidwordlookupsbydisplayingameaningfulerrormessagewhenthewordisnot found in the dictionary.

### Security:

* + - * + Theprogramusesthesystem()functiontoexecutefileopeningcommands.Whilethiscould be a security concern in some contexts, it's safe in this instance as it’s limited to handling specific predefined operations, minimizing any risk.

### RequiredToolsandLibraries

1. **PythonInterpreter**:
   * AcompatiblePythoninterpreter(3.xorlater).

### StandardLibraries:

* + **csv**:ForreadingandwritingdatatoCSVfiles.
  + **docx**:Tocreateand savedatainDOCXformat.
  + **os**:Forinteractingwiththeoperatingsystem.

## Advantages

### EfficientDataInsertion:

* + Using linear hashing ensures quick and efficient insertion of words, meanings, and synonyms into the hash table, even when handling a large number of entries.

### User-FriendlyDataNavigation:

* + Thestack-basedhistoryfeatureallowsuserstoeasilybacktracktopreviouslyinsertedwords, providing flexibility for navigation and management of the hash table.

### Case-InsensitiveDataSearch:

* + Byensuringcase-insensitivesearch,theprogramallowsuserstosearchforwordsregardlessof capitalization, simplifying the search process.

### DataExportto MultipleFormats:

* + The program provides a simple wayto export hash table data into different formats (CSV, DOCX, TXT), enhancing usability without requiring complex user interfaces.

### Simpleand InteractiveConsoleInterface:

* + The program uses colored text and clear, structured prompts to make the user interface interactive and easy to navigate, even in a text-based console environment.

### Expandability:

* + Thehashtablecanbeeasilyextendedwithmorewordsoradditionalfeatures(e.g.,dynamicresizing, more search options) without major modifications to the existing code structure.

## Disadvantages

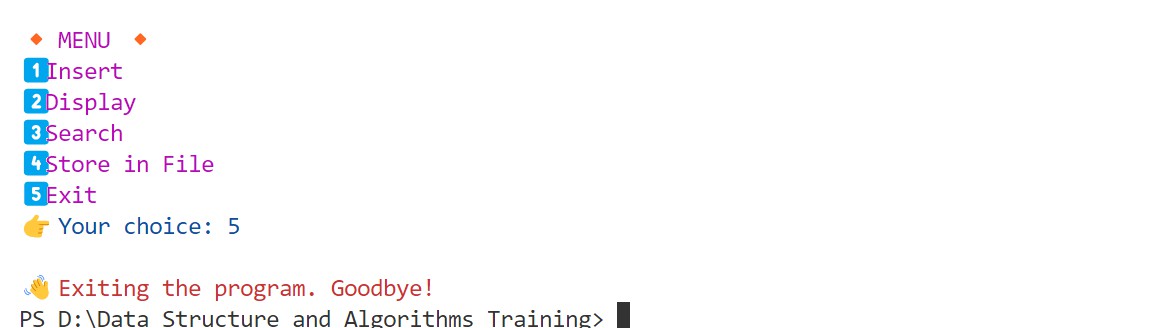
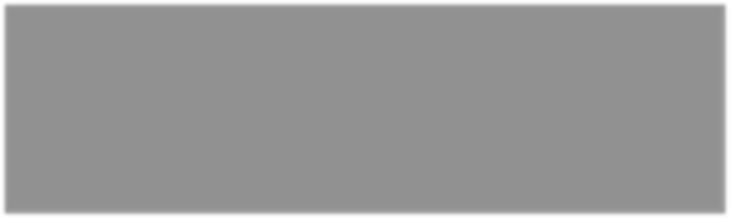
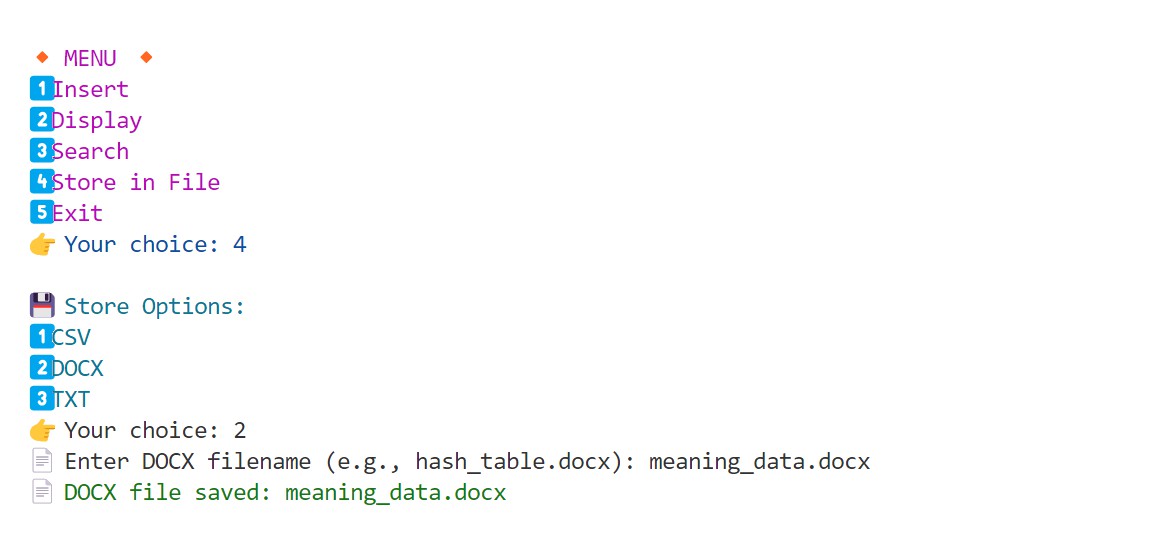
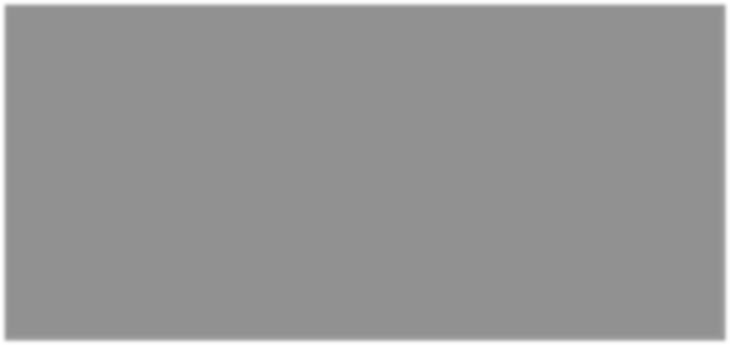
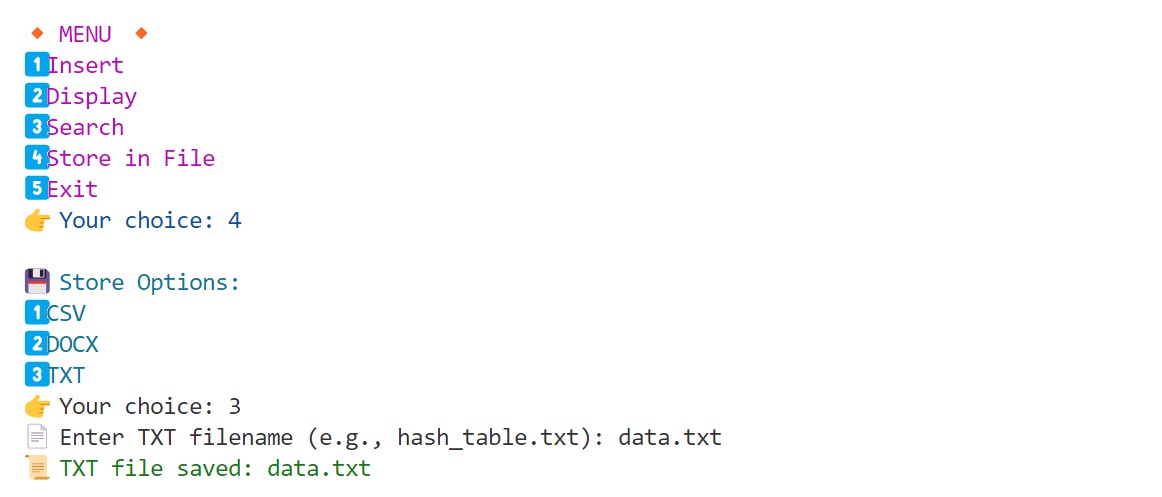
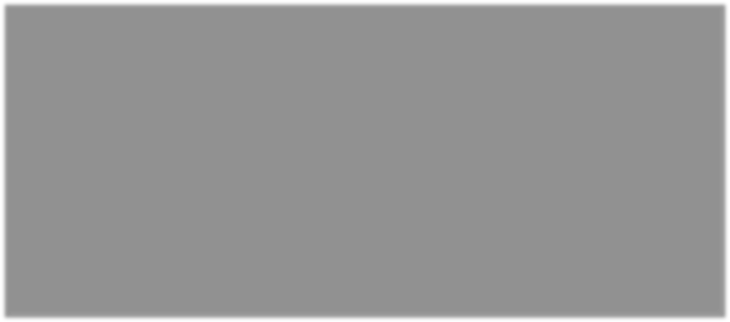
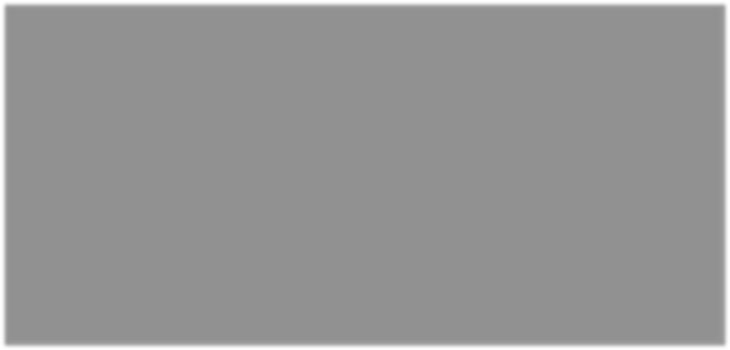
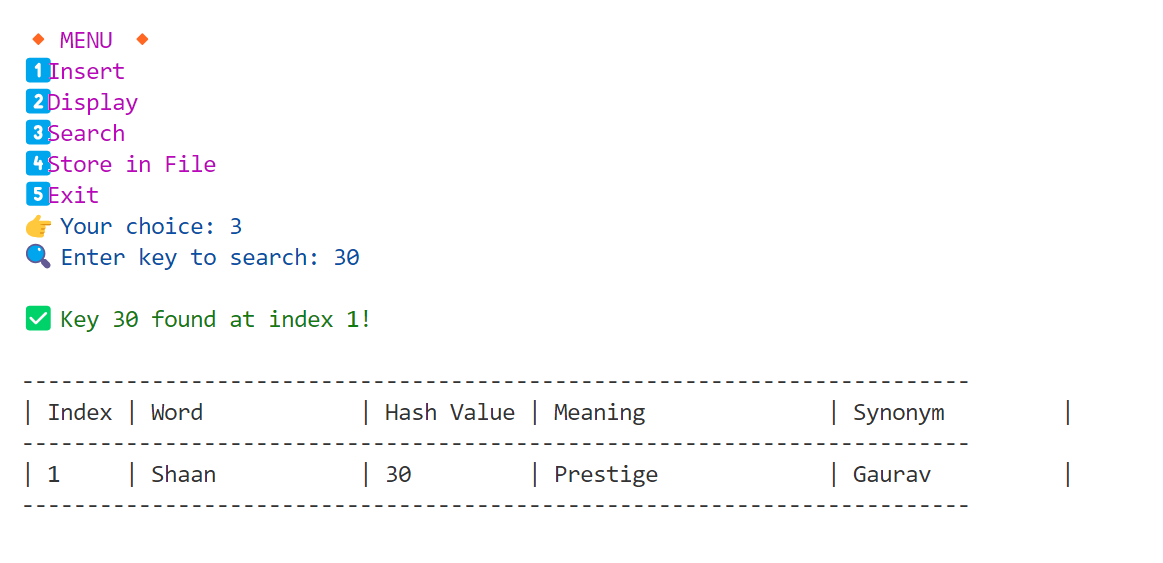
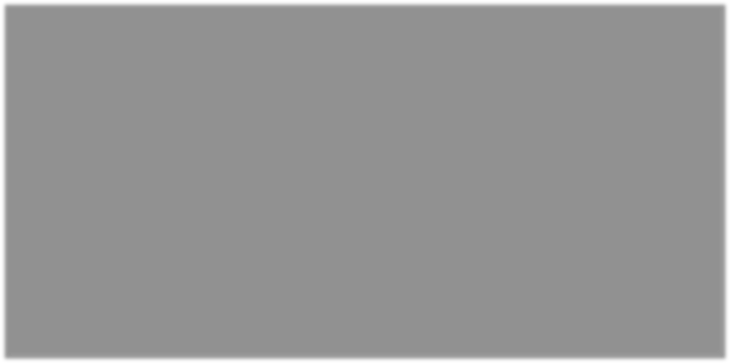
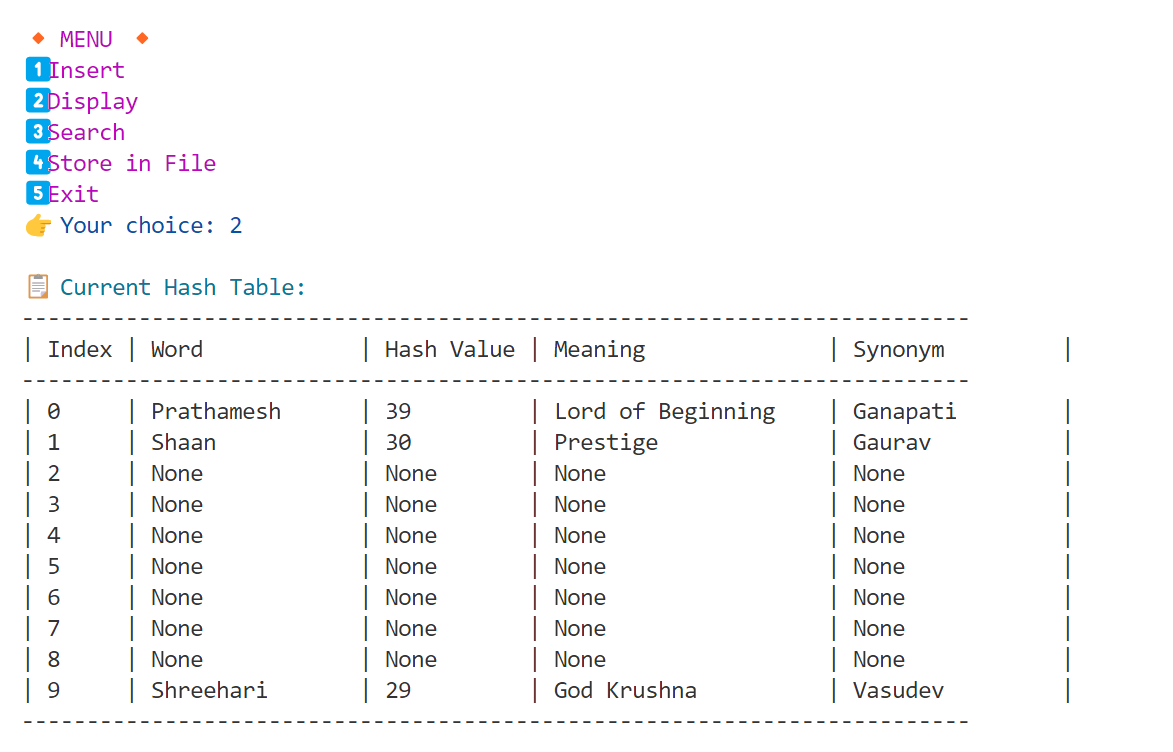
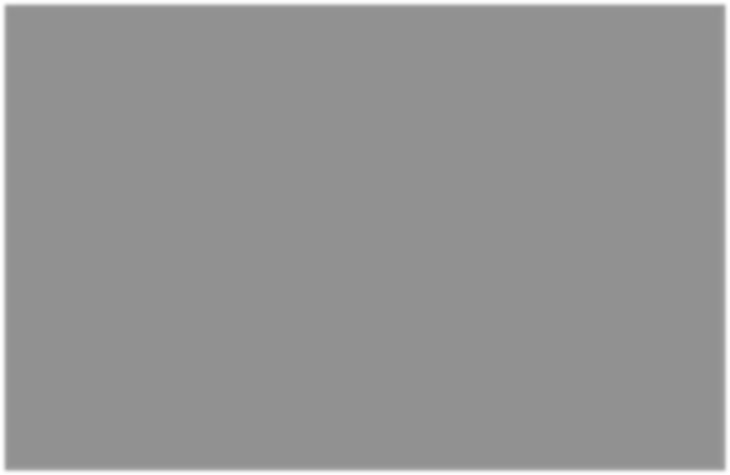
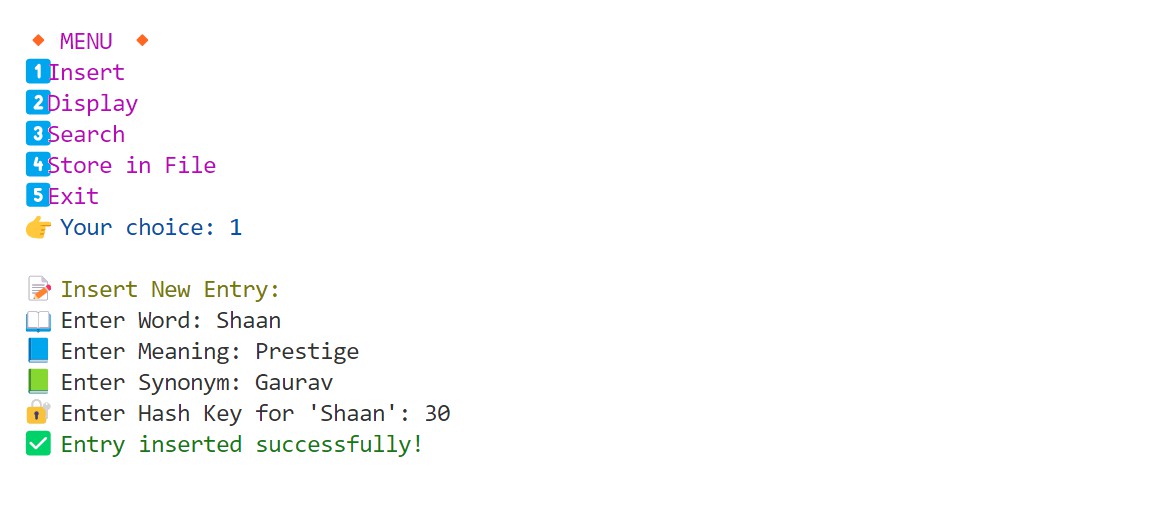
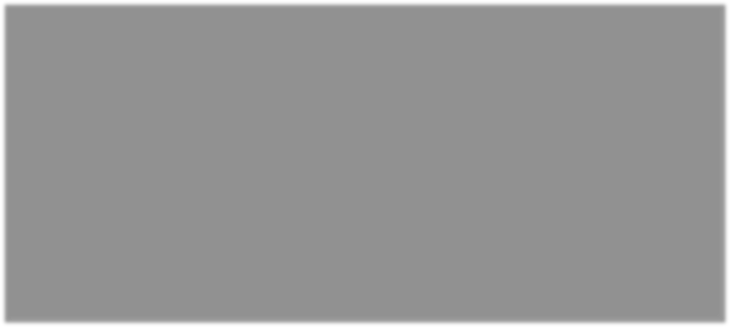
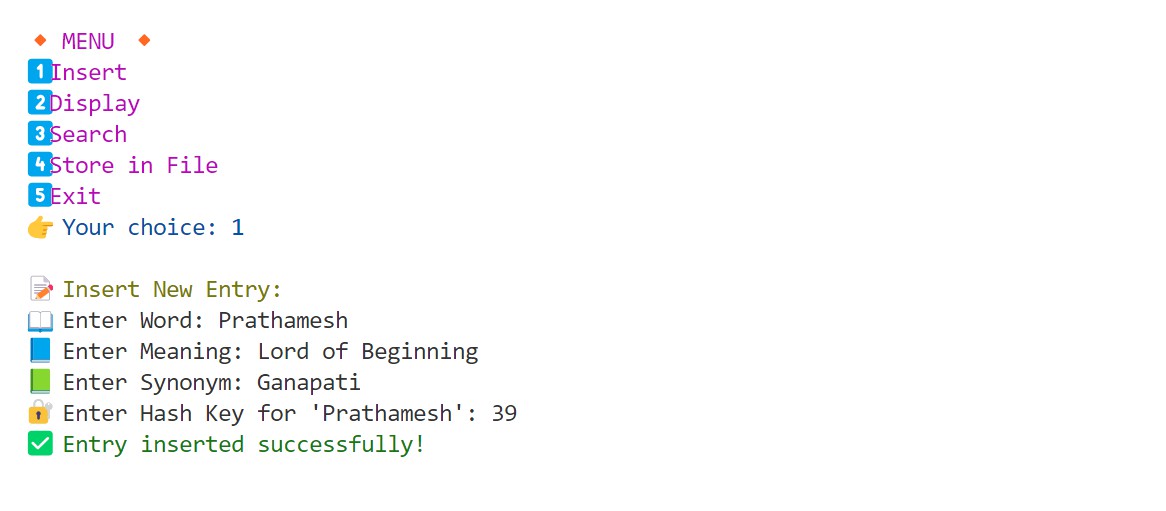
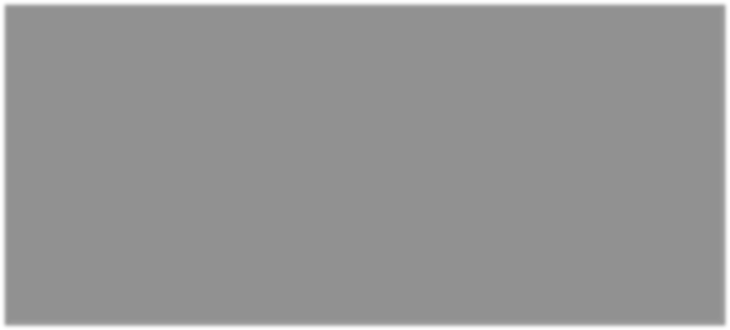
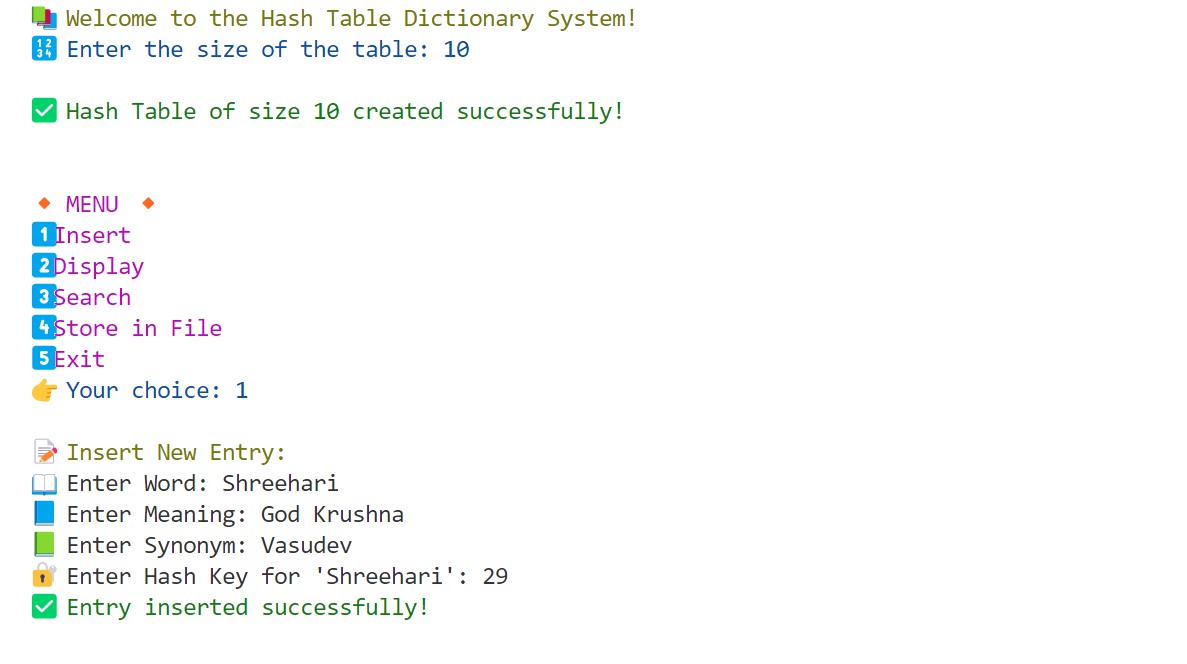
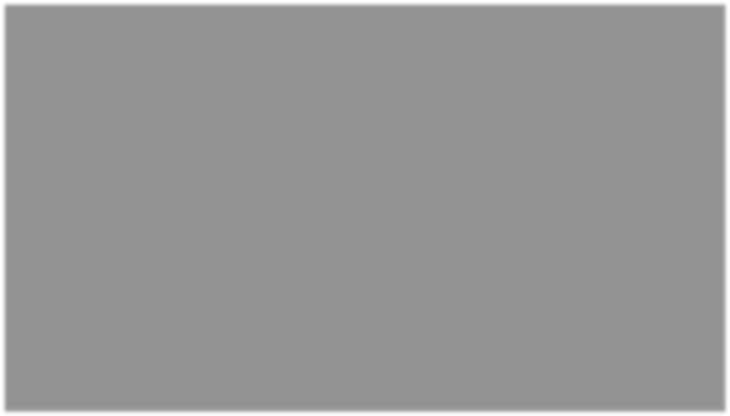
### LimitedtoPredefinedData:

* + - * Theprogramreliesonapredefinedlistofwordsanddata.Iftheusersearchesforawordnot already in the hash table, the system cannot retrieve or display relevant information dynamically, limiting its flexibility.

### PlatformDependencyforFileOperations:

* + - * Theprogramusescertainplatform-specificmethodsforfilehandling(e.g.,fileoperationson Windows),makingitlessportableacrossdifferentoperatingsystems.Adjustmentswouldbe needed for compatibility with other platforms, reducing overall portability.

# 5.1RESULT&OUTPUT



**CONCLUSION**

This program effectively demonstrates key concepts in hash table implementation and data management, including handling collisions through linear probing, efficient data storage, and search operations. It also includes features for storing dictionary entries in various file formats, enhancing the program’s utility. However, the program has limitations: it uses a basic hashing technique and doesn't support dynamic resizing, which could impact performance as the dictionary size grows. To improve performance and scalability, optimization for dynamic resizing, better collision handling techniques, and implementation of amoreadvancedfilestoragemethodcouldfurtherenhanceitsefficiencyandflexibilityonmodernsystems

# REFERENCES

1. **Functional Programming:**E. S. Bainbridge, P. J. Freyd, A. Scedrov, and P. J. Scott, Functorial polymorphism.InG.Huet,editor,LogicalFoundationsofFunctionalProgramming,Austin,Texas, 1987. Addison-Wesley, to appear.
2. **Semantics of Datatypes:**K. B. Bruce and A. R. Meyer, The seman- tics of second-order polymorphiclambdacal-culus.InKahn,MacQueen,andPlotkin,editors,SemanticsofDataTypes, Sophia- Antipolis, France, 1984, pp. 131–144. LNCS 173, Springer-Verlag.
3. **Models for Polymorphism:**V. Breazu-Tannen and T. Coquand, Exten- sional models for polymorphism. Theoretical Computer Science, 59:85-114, 1988.
4. **MathematicalAspects:**M.Sheeran,Categoriesfortheworkinghard-waredesigner.InWorkshop onHardwareSpecification,Verification, andSynthesis:MathematicalAspects,Cornell,July1989.
5. **Mapfusion:**Making Haskell225%faster.
6. **LISPProgramming:**J.McCarthy,K.Maling,S.Russell,N.Rochester,S.Goldberg,J.Slagle.

LISPProgrammer'sManual.March-April,1959

1. **GraceHopper**:Pioneeredhigh-levelprogramminglanguagesandcompilerdesign.
2. **Donald Knuth**: Developed foundational work in algorithms and data structures like stacks andmaps.
3. **JohnvonNeumann**:Developedcomputerarchitecturefundamentalsusedinprocessing.
4. **ClaudeShannon**:Laidthegroundworkfor datastorage andbinarysystemsincomputers.