Introduction

30! is a **time based word game** where you have to guess as many words as possible from a given square letter grid.
The player will have **30 seconds** to guess words from the grid.
The words have to be at least 3 letters long;
If he guesses **at least 5 words** the player wins.

```
u v w b y w
a b t t l q
g t o b i h
j t w e x p
m e l q y d
p j a n h x
```

The key algorithm which works behind this implementation is **hashing.**
File Handling is instrumental in extracting all possible words which can be formed in the grid by comparing them with a dictionary.

Language Used

C++

Functionality

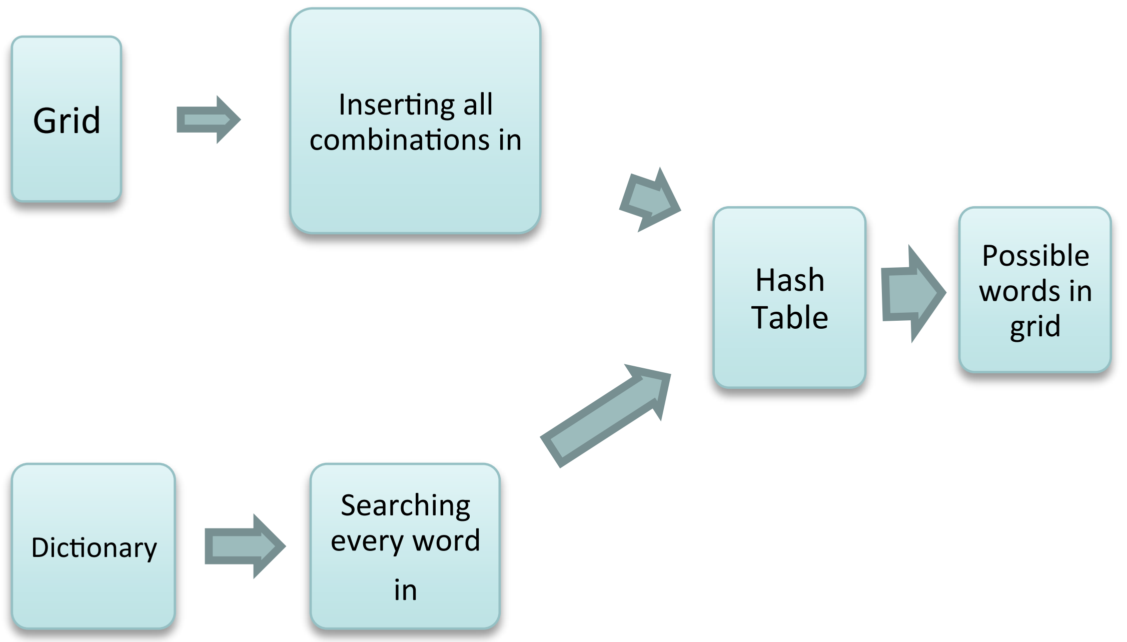
The game is coupled with many **interactive features**.
These are listed in order of appearance in the game.

1. The **size** of the grid is filled by the player. The available options are 5,6,7,8,9.
2. The difficulty of the game is chosen:
   - **Easy**
   - **Moderate**
   - **Difficult**
3. A timer which prompts the time each time a guess is entered.
4. Post-game, the score is shown along with the correct guesses and all the possible answers.

30! a word game!

Algorithm

```
1.  input size of grid, and level of difficulty
2.  creation of randomly generated letter grid
3.  extracting all possible adjacent horizontal, vertical and diagonal
    letter combinations
4.  inserting all these in a hash table
5.  opening the dictionary file
6.  traversing the dictionary and checking for presence of each
    English word in the hash table
    1. if present, push it in a list "PossibleWords"
7.  checking if the solution set is in agreement with the level of
    difficulty,
    1. if not goto step 2
8.  displaying the grid
9.  starting the timer and taking in guesses
    1. if a guess is in list "PossibleWords", increment score by 1
       and add guess in list "CorrectGuesses"
    2. if timer is < 30s goto step 9
10. display the score, your correct guesses list CorrectGuesses and
    all possible solutions from list PossibleWords
11. ask if the player wants to play again
```



Data Structure and Algorithms used

Data Structure

Hash Tables: Hash Table was used in this project primarily for comparing:
- all the grid combinations, and
- English words from dictionary

The hash function used was **FNV hash** because it results in almost no collisions.
Why Hash Tables?
There are huge amounts of data to be compared and hash tables take least time for searching. O(1)

Algorithms

Possible Combinations Extraction From Grid Algorithm:
To enlist all possible letter combinations from the grid.
All the grid elements are visited and all of their combinations are taken.

f g y i	for element z, all combinations are:	
j z c e	zg , zs, zsa	:horizontal
u s b i	zj , zc , zce	:vertical
v a y l	zf , zy , zu , zb , zbl	:diagonal

Future Scope

By combining the app with **daily progress charts, timely notifications and detailed reports** regarding the performance of the user, the app has a potential of becoming a vocabulary building tool which can specially be beneficial for MBA,MS,IAS aspirants.

Also, by adding features like **multiplayer battles, social rewards and online community tournaments and forums**, the project 30! can be expanded into an extensive game app.

The game can be modified to work as a lock screen key during alarm such that the mobile can't be unlocked until the user successfully traces a word from the grid of letters which will help ensuring that the user is no longer in sleep.

Efforts by:
Rishabh Kumar Saini 101453006 COE9