Project Report

Default Predicates Implementation:

1 – The is_category(C) predicate is implemented by calling the asserted word/2 facts in the database of the compiler. With the words as don't cares (_) and the category as the variable C. (It checks if C is a valid category in the database of the asserted words).

2 – The categories(L) predicate is implemented using Prolog's built-in setof/3 predicate to prevent duplicates, where the things to be put in a list are the things stored in the variable C, the condition being that C is a category, and we store everything in the list L. (True if L is a list containing all categories in the database).

3 – The available_length(L) predicate is implemented by getting a word X from the database of asserted words without considering the category and then we compare X with the given length L using prolog's built-in atom_length/2 predicate. (Checks if L is a valid Length of a word in the database of asserted words).

4 – The pick_word(W, L, C) predicate is implemented by checking that W is a valid word in the asserted words database with category C using the fact word(W, C) and then using the atom_length/2 built-in predicate

to check if the word W has the length L or not. (Checks if W is a valid word in database with category C and Length L).

5 – The correct_letters(L1, L2, L3) predicate is one of six recursive predicates. It checks if L3 is the list containing the common letters present in L1 and L2 without duplicates. It checks if the first element in L1 is a member of L2 or not; if true, it adds the element to L3 and deletes all occurrences of the letter in L2 using prolog's built-in delete/2 predicate then calls itself to check the next element, otherwise it just calls itself without adding the element to L3. (Note L1, L2, and L3 are lists).

6 – The correct_position(L1, L2, L3) is the second of the six recursive predicates. It checks if L3 is a list containing the common elements between L1 and L2, with duplicates of course. It is implemented by taking the first element of L1 and L2 and comparing them with each other; if they match, then the element is added to L3 and the predicate is called to get the next elements in both lists, otherwise the elements are not added to L3 and the predicate is called to get the next elements.

7 – chosen_category(C) is a helper predicate and the third of the six recursive predicates implemented in the project. It writes to the user the available categories and asks him to input a category L; if L is not a valid category, then it outputs to the user that the input is not valid and then calls itself to redo the process above, otherwise it matches C to L and then stops backtracking using the cut predicate (!).

8 – chosen_length(Category, Length) is the second helper predicate and the fourth recursive predicate. It checks if L is a valid length of any word in the category C. It asks the user to input a length. If the length is not valid it tells the user that the length is not valid and then calls itself to repeat the above-mentioned process, otherwise it matches Length to L and prevents backtracking using the built-in cut predicate (!).

9 – the play/0 predicate is the predicate that sets up and prepares everything for the core logic of the game. First it asks the user to enter a valid category C using the chosen_category(C) predicate mentioned above in point 7. Then it asks the user to enter a valid length L in the category C by calling the chosen_length(C, L) predicate mentioned above in point 8. Then it picks the result word R by calling the pick_word(R, L, C) predicate mentioned above in point 4. Then it assigns a value of L + 1 to the variable G to indicate the guesses, then it outputs to the user a message telling him or her that the game has started and that he or she has G guesses. Lastly, it proceeds by calling the play(R, C, L, G) helper predicate to start the game.

10 – Now we come to the core logic of the game. The play(Result, Category, Length, Guesses) helper predicate, which is the fifth recursive predicate. This predicate checks if the number of guesses the user has is zero, and if it is true, then it outputs You Lost to the user, otherwise it does the following. Firstly, it checks if the number of guesses is more than zero. Secondly, it asks the user to enter a word Answer of Length letters from

the Category category. After the input, we use the pick_word/3 predicate to check if the word Answer is indeed a word in the category Category of length Length, then we check if the Answer matches with the word Result, if the above two conditions are true, then we retract the word from the database using the built-in retract predicate, then output to the user you won, then exit out of the predicate using the cut predicate to prevent backtracking. If one of the previous two conditions was false, we do the following. We use the atom_length/2 predicate to check if the word Answer is of length Length, if it is true then we check that the Answer does not match with the result. After that we proceed to get the letters of the word Answer and the word Result in two separate list called AL and RL, respectively, using prolog's built-in atom_chars/2 predicate. Then we proceed to get the correct letters in a list CL and the letters in the correct positions in a separate List PL using the correct_letters/3 and the correct_positions/3 predicates as discussed above in points 5 and 6 and we assign the value of Guesses – 1 to a new variable called NewGuesses. After that we output the correct letters in the word and the CL and PL lists to the user and tell him or her the remaining guesses are NewGuesses and then call the play/4 predicate with the following parameters: play(Result, Category, Length, NewGuesses). However, if the input answer does not have the length Length, we tell the user that the input word Answer is not composed of Length letters and then proceed to call the play predicate with the same parameters: play(Result, Category, Length, Guesses).

11 – The build_kb/0 predicate is the last of the recursive predicates and it is responsible for building the database of the game. It outputs to the user a message asking him or her to enter the word X and its respective category Y on two separate lines then it asserts the word(X, Y) fact using prolog's build-in assert predicate. Then the predicate calls itself to repeat the process until the user enters the word done either in the word or its category, in which case the predicate terminates the recursion.

12 – At last we have the main predicate, which is responsible for building the database of the game and the gameplay itself. It is simply implemented by calling the build_kb/0 predicate, then the cut predicate to prevent backtracking, then the play/0 predicate to start the gameplay.

The Gallery containing both runs is found bellow.

Gallery

The First Run (Winning)

Building The KB:

```
?- main.
Please enter a word and its category on separate lines:
|: father.
|: family.
Please enter a word and its category on separate lines:
|: mother.
|: family.
Please enter a word and its category on separate lines:
|: sister.
|: family.
Please enter a word and its category on separate lines:
|: brother.
|: family.
Please enter a word and its category on separate lines:
|: laptop.
|: collection.
Please enter a word and its category on separate lines:
|: phone.
|: collection.
Please enter a word and its category on separate lines:
|: painting.
|: collection.
Please enter a word and its category on separate lines:
|: lion.
|: animal.
Please enter a word and its category on separate lines:
|: horse.
|: animal.
Please enter a word and its category on separate lines:
|: tiger.
|: animal.
Please enter a word and its category on separate lines:
|: done.

Done building the words database...
```

Entering the Category and Length:

```
The available categories are: [animal,collection,family]
|: test.
This category does not exist.

The available categories are: [animal,collection,family]
|: family.
Choose a length:
|: 4.
There are no words of this length.
Choose a length:
|: 5.
There are no words of this length.
Choose a length:
|: 6.
Game Started. You have 7 guesses.
```

The Gameplay:

```
Enter a word composed of 6 letters
|: mother.
Correct letters are: [t,h,e,r]
Correct letters in correct positions are: [t,h,e,r]
Remaining guesses are 6

Enter a word composed of 6 letters
|: sister.
Correct letters are: [t,e,r]
Correct letters in correct positions are: [e,r]
Remaining guesses are 5

Enter a word composed of 6 letters
|: brother.
Word is not composed of 6 letters. Try again.
Remaining Guesses are 5

Enter a word composed of 6 letters
|: father.
You Won.
```

The Second Run (Losing)

Building The KB:

```
?- main.
Please enter a word and its category on separate lines:
|: octopus.
|: animal.
Please enter a word and its category on separate lines:
|: cheeta.
|: animal.
Please enter a word and its category on separate lines:
|: panda.
|: animal.
Please enter a word and its category on separate lines:
|: ahmed.
|: name.
Please enter a word and its category on separate lines:
|: adam.
|: name.
Please enter a word and its category on separate lines:
|: maged.
|: name.
Please enter a word and its category on separate lines:
|: phone.
|: electronic.
Please enter a word and its category on separate lines:
|: laptop.
|: electronic.
Please enter a word and its category on separate lines:
|: table.
|: furnature.
Please enter a word and its category on separate lines:
|: couch.
|: furnature.
Please enter a word and its category on separate lines:
|: done.

Done building the words database...
```

Entering The Category and Length:

```
The available categories are: [animal,electronic,furnature,name]
|: electronic.
Choose a length:
|: 4.
There are no words of this length.
Choose a length:
|: 6.
Game Started. You have 7 guesses.
```

The Gameplay:

```
Enter a word composed of 6 letters
|: airpod.
Correct letters are: [a,p,o]
Correct letters in correct positions are: [o]
Remaining guesses are 6

Enter a word composed of 6 letters
|: cheeta.
Correct letters are: [t,a]
Correct letters in correct positions are: []
Remaining guesses are 5

Enter a word composed of 6 letters
|: tablet.
Correct letters are: [t,a,l]
Correct letters in correct positions are: [a]
Remaining guesses are 4

Enter a word composed of 6 letters
|: earset.
Correct letters are: [a,t]
Correct letters in correct positions are: [a]
Remaining guesses are 3

Enter a word composed of 6 letters
|: tables.
Correct letters are: [t,a,l]
Correct letters in correct positions are: [a]
Remaining guesses are 2

Enter a word composed of 6 letters
|: locker.
Correct letters are: [l,o]
Correct letters in correct positions are: [l]
Remaining guesses are 1

Enter a word composed of 6 letters
|: iphone.
Correct letters are: [p,o]
Correct letters in correct positions are: []
Remaining guesses are 0

You Lost.
```

Report Prepared By Ziad Elsabbagh.