KB PHASE:

```
1 ?- main.
Welcome to Pro-Wordle!
----------------------

Please enter a word and its category on separate lines:
|: laptop.
|: tech.
Please enter a word and its category on separate lines:
|: phone.
|: tech.
Please enter a word and its category on separate lines:
|: pizza.
|: food.
Please enter a word and its category on separate lines:
|: pasta.
|: food.
Please enter a word and its category on separate lines:
|: fries.
|: food.
Please enter a word and its category on separate lines:
|: hound.
|: animals.
```

WINNING:

```
|: animals.
Please enter a word and its category on separate lines:
|: done.


Done building the words database...

The available categories are: [animals,food,tech]
Choose a category:
|: tech.
Choose a length:
|: 5.

Game started. You have 6 guesses

Enter a word composed of 5 letters
|: fries.
Correct letters are: [e]
Correct letters in correct positions are: []

-> Remaining guesses are 5
```

```
Enter a word composed of 5 letters
|: hound.
Correct letters are: [h,o,n]
Correct letters in correct positions are: [n]

-> Remaining guesses are 4

Enter a word composed of 5 letters
|: pasta.
Correct letters are: [p]
Correct letters in correct positions are: [p]

-> Remaining guesses are 3

Enter a word composed of 5 letters
|: phone.
You Won!
true .

2 ?-
```

LOSING

```
The available categories are: [animals,food,tech]
Choose a category:
|: food.
Choose a length:
|: 5.

Game started. You have 6 guesses

Enter a word composed of 5 letters
|: hound.
Correct letters are: []
Correct letters in correct positions are: []

-> Remaining guesses are 5

Enter a word composed of 5 letters
|: phone.
Correct letters are: [p]
Correct letters in correct positions are: [p]

-> Remaining guesses are 4
```

```
Enter a word composed of 5 letters
|: fries.
Correct letters are: [i]
Correct letters in correct positions are: []

-> Remaining guesses are 3

Enter a word composed of 5 letters
|: pasta.
Correct letters are: [p,a]
Correct letters in correct positions are: [p,a]

-> Remaining guesses are 2

Enter a word composed of 5 letters
|: pasti.

Invalid Word
->Remaining Guesses 2
```

```
->Remaining Guesses 2
Enter a word composed of 5 letters
|: phone.
Correct letters are: [p]
Correct letters in correct positions are: [p]

-> Remaining guesses are 1

Enter a word composed of 5 letters
|: laptop.

Word is not composed of 5 letters. Try again.
->Remaining guesses are 1

Enter a word composed of 5 letters
|: pasto.

Invalid Word
->Remaining Guesses 1
Enter a word composed of 5 letters
|: pasta.
You Lost
true
```

1) main-> -using write to print the required statements, and calls the build_kb predicate, then calls the play predicate.

2) build_kb -> using write to print the required statements, first will read the input if it is done , we go back to main, if not then we will use assert to add the incoming word with it it's corresponding category and then recursive call build_kb again.

3) is_category(A)-> is true if there is word in my KD with A as a category.

4) categories(L)-> uses setof to return a list of the categories without duplicates, I implemented this technique as I came across it in the interwebs, basically what the A^ does it that it says that I am not interested in the first argument of the predicate word, only interested in the second argument which is C, thus it is true if L is a L of distinct C's.

5)available_length(L,Cat)-> is true if there is a word in the KB with length L and category Cat, check all the words in a category, and see if any word has length L.

6)pick_word(W,X,C)-> picks a word with length X from category C, using my implemented available_length, and selected any word that has a length X, but available length must satisfied first, and the cut to ensure only one word is the result for further uses.

7)correct_letters()-> checks if the head of the first list is a member in the second in list, if yes then we will use predefined delete to remove all instances of that element(to eliminate duplicates), and using recursion we will get our list which is a list of the intersection of the both lists.

8)correct_positions-> we used an accumulator that accumulates when both heads are equal and using recursion, we will get our resulting list when both lists are empty (other base cases are negligible since we will only compare words of equal lengths).

9) play-> writes required statements and presents the available categories and calls initializegame1.

10)initializegame1-> chooses a category, checks if it is in the KB, if not prompts the user to enter a valid one(recursively), then calls initilizegame2 which takes an argument of the chosen category.

11)initializegame2-> prompts the user to choose a length, check if there is a word with the length in the category, if not prompts the user to enter another length(recursively), and then get the number of guesses which is length + 1, then using pick_word, we will get a word with the specified length, in the specified category, and then calls play2 which takes as arguments the length, the word from pick_word, the no. of guesses and the category.

12)play2 -> prompts to user to enter a word of the specified length, if the user's input equates the word that play2 takes as an argument the user wins. Otherwise, we will check if it is in the correct length, if it is then we will see if the word exists in the KB, then check if the current no of guesses is greater than one , if any of these are not satisfied the appropriate message will be prompted ,if they are, then we will use the predefined predicate string_chars to convert the words into a list of chars in order to user our predicates correct letters and correct positions then we print the lists from both our predicates, then we decrement the no. of guesses and recursively call play2 again with the updated no. of guesses, if the no. of guesses reaches 0 the player loses.