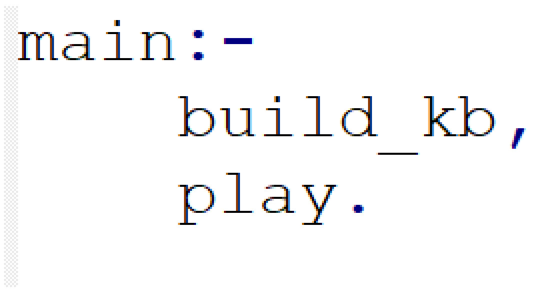
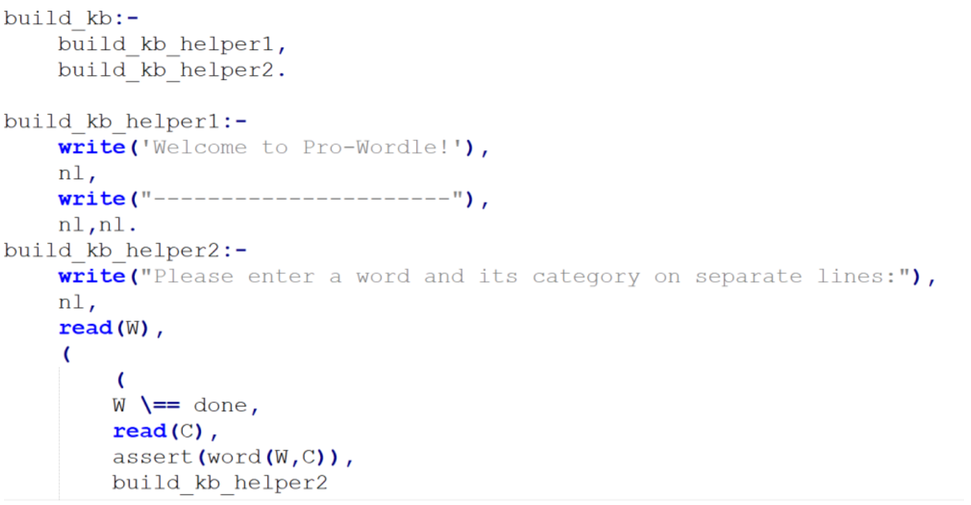
Team 44

- 52-1008 Aly Raafat AbdelFattah Aly T-06
- 52-3899 Karim Mohamed Gamaleldin Yehia Gamaleldin T-07
- 52-1805 Bassel Mohamed Farouk T-21
- 52-3434 Ahmed Labib Mohamed Maged Amer T-21
- 52-2292 Marwan Amgad Mohamed Abdelfattah T-09



```
main:-
	build_kb,
	play.
```

main/0 prompts the user to first build the knowledge base, then play the pro-wordle game.

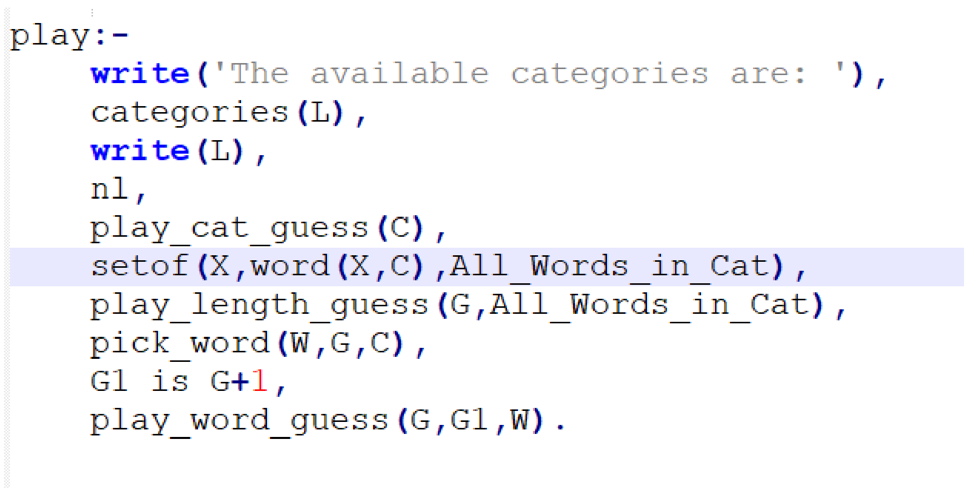


```
build_kb:-
    build_kb_helper1,
    build_kb_helper2.

build_kb_helper1:-
    write('Welcome to Pro-Wordle!'),
    nl,
    write("----------------------"),
    nl,nl.
build_kb_helper2:-
    write("Please enter a word and its category on separate lines:"),
    nl,
    read(W),
    (
        (
        W \== done,
        read(C),
        assert(word(W,C)),
        build_kb_helper2
```

build_kb/0 uses build_kb_helper1 and build_kb_helper2.

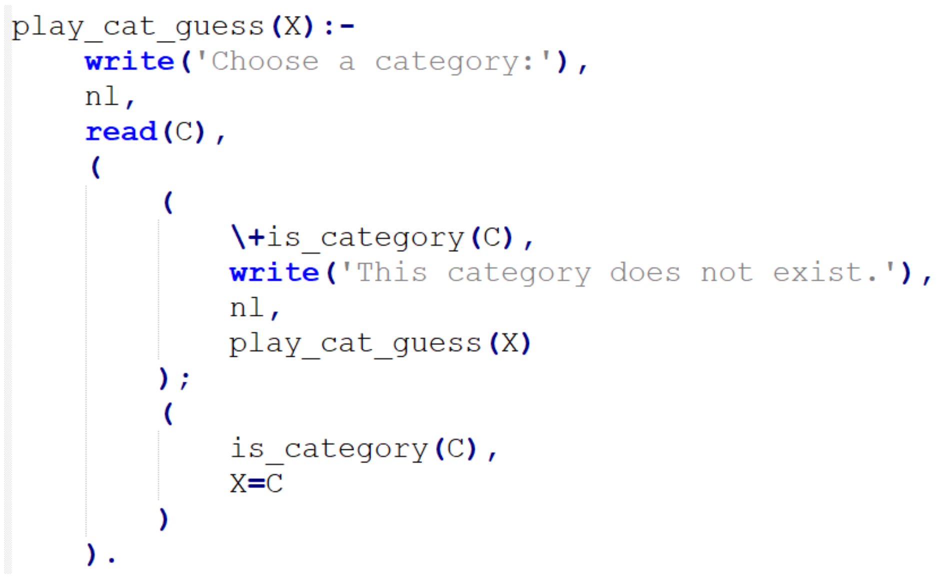
build_kb_helper1/0 writes the first two lines in the game.

build_kb_helper2/0 prompts the user to enter recursively a word and then its category until the user enters done.



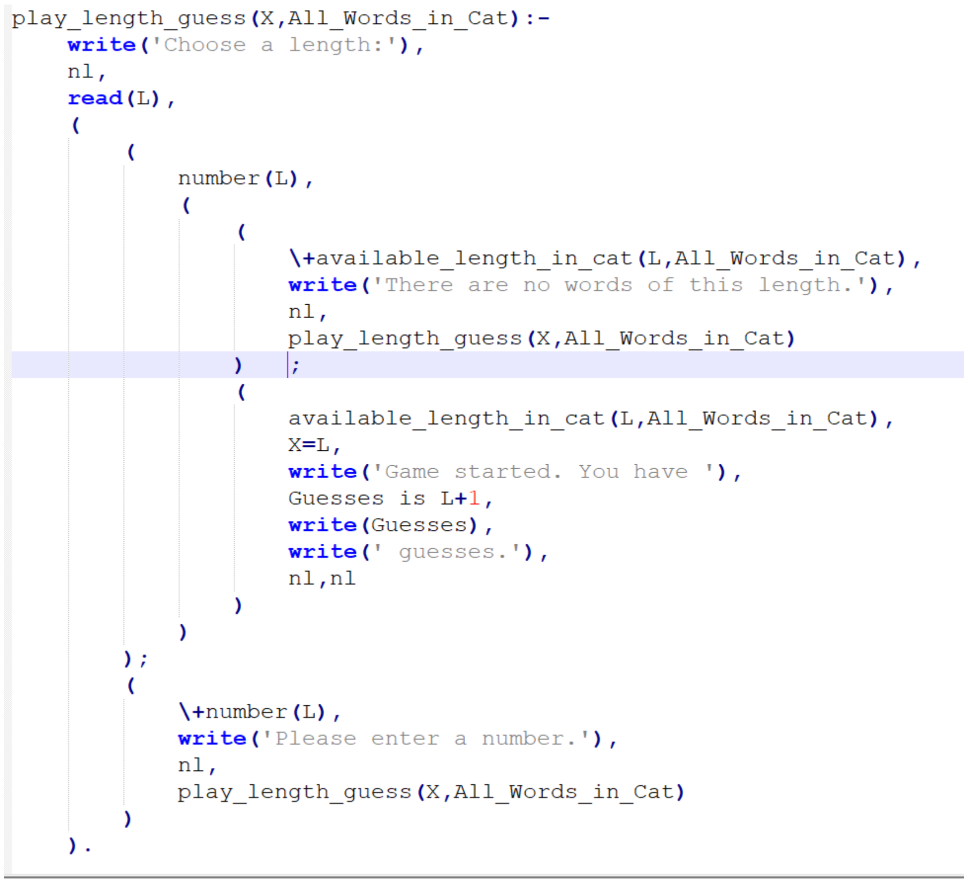
```
play:-
    write('The available categories are: '),
    categories(L),
    write(L),
    nl,
    play_cat_guess(C),
    setof(X,word(X,C),All_Words_in_Cat),
    play_length_guess(G,All_Words_in_Cat),
    pick_word(W,G,C),
    G1 is G+1,
    play_word_guess(G,G1,W).
```

play/0 that allows the user to guess the categories, then allows the user to enter a length and checks if this length is available and allows the user to guess the word.



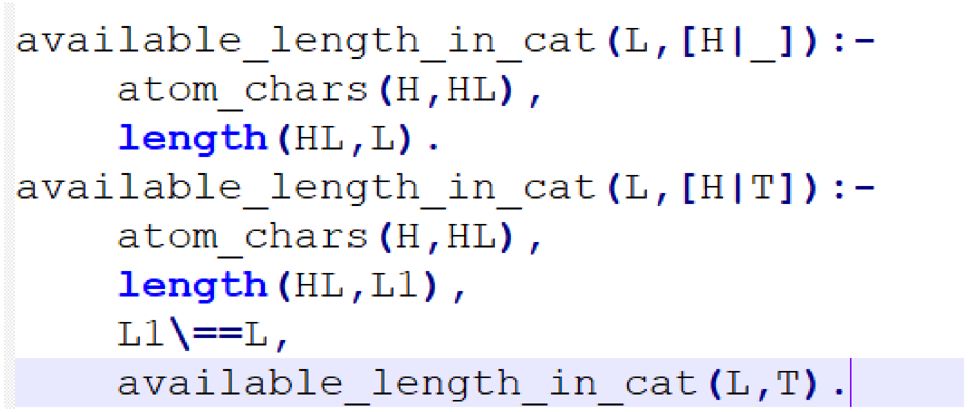
```
play_cat_guess(X):-
    write('Choose a category:'),
    nl,
    read(C),
    (
        (
            \+is_category(C),
            write('This category does not exist.'),
            nl,
            play_cat_guess(X)
        );
        (
            is_category(C),
            X=C
        )
    ).
```

play_cat_guess/1 which prompts the user to enter a category recursively and checks if this category is valid, it stops when the user enters a valid category.



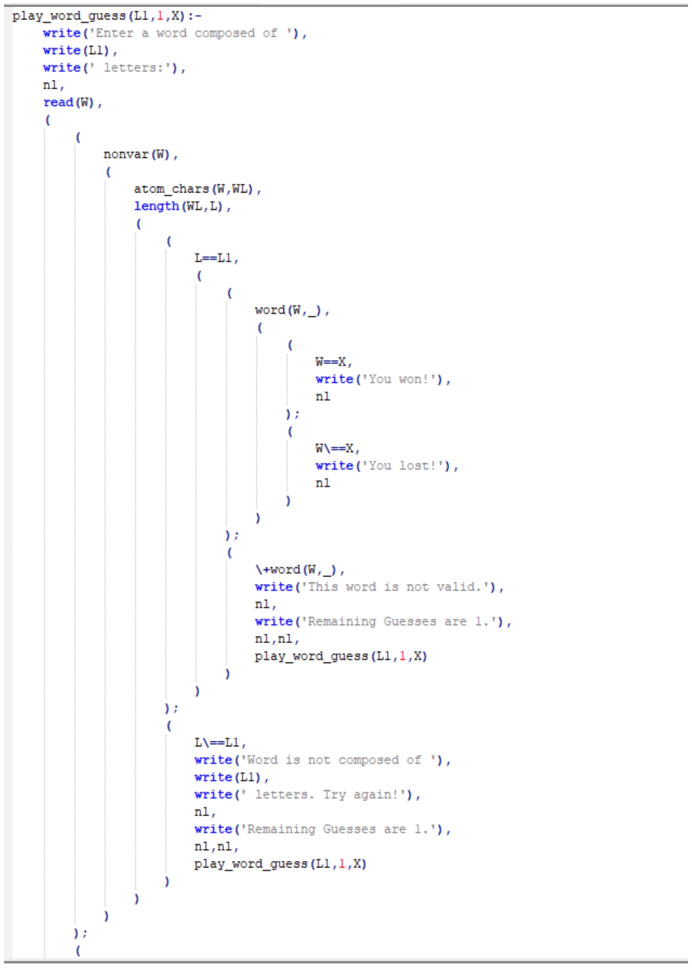
```prolog
play_length_guess(X,All_Words_in_Cat):-
    write('Choose a length:'),
    nl,
    read(L),
    (
        (
            number(L),
            (
                (
                    \+available_length_in_cat(L,All_Words_in_Cat),
                    write('There are no words of this length.'),
                    nl,
                    play_length_guess(X,All_Words_in_Cat)
                ) ;
                (
                    available_length_in_cat(L,All_Words_in_Cat),
                    X=L,
                    write('Game started. You have '),
                    Guesses is L+1,
                    write(Guesses),
                    write(' guesses.'),
                    nl,nl
                )
            )
        );
        (
            \+number(L),
            write('Please enter a number.'),
            nl,
            play_length_guess(X,All_Words_in_Cat)
        )
    ).
```

play_length_guess/2 which prompts the user to enter a length. If there are words available with this length then All_Words_in_Cat is a list of all the words in this category with this length.

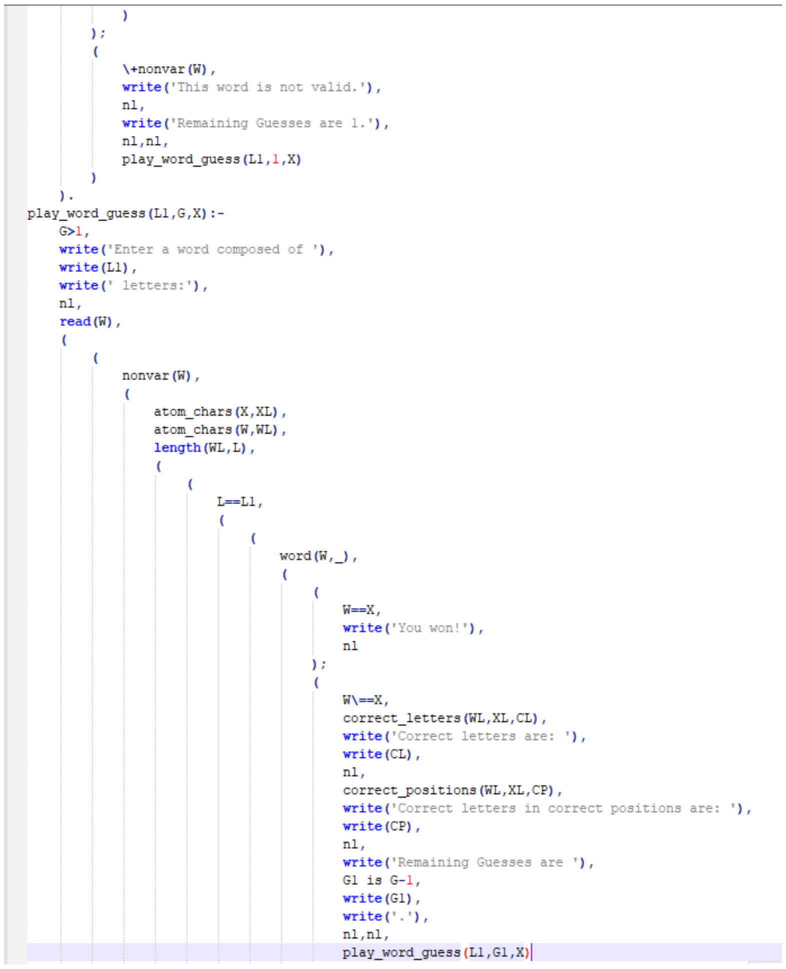


```prolog
available_length_in_cat(L,[H|_]):-
    atom_chars(H,HL),
    length(HL,L).
available_length_in_cat(L,[H|T]):-
    atom_chars(H,HL),
    length(HL,L1),
    L1\==L,
    available_length_in_cat(L,T).
```

available_length_in_cat/2 succeeds if H is a word which has the same number of characters as L.



```
play_word_guess(L1,1,X):-
    write('Enter a word composed of '),
    write(L1),
    write(' letters:'),
    nl,
    read(W),
    (
        (
            nonvar(W),
            (
                atom_chars(W,WL),
                length(WL,L),
                (
                    (
                        L==L1,
                        (
                            (
                                word(W,_),
                                (
                                    (
                                        W==X,
                                        write('You won!'),
                                        nl
                                    );
                                    (
                                        W\==X,
                                        write('You lost!'),
                                        nl
                                    )
                                )
                            );
                            (
                                \+word(W,_),
                                write('This word is not valid.'),
                                nl,
                                write('Remaining Guesses are 1.'),
                                nl,nl,
                                play_word_guess(L1,1,X)
                            )
                        )
                    );
                    (
                        L\==L1,
                        write('Word is not composed of '),
                        write(L1),
                        write(' letters. Try again!'),
                        nl,
                        write('Remaining Guesses are 1.'),
                        nl,nl,
                        play_word_guess(L1,1,X)
                    )
                )
            )
        );
        (
```



```
            )
        );
        (
            \+nonvar(W),
            write('This word is not valid.'),
            nl,
            write('Remaining Guesses are 1.'),
            nl,nl,
            play_word_guess(L1,1,X)
        )
    ).
play_word_guess(L1,G,X):-
    G>1,
    write('Enter a word composed of '),
    write(L1),
    write(' letters:'),
    nl,
    read(W),
    (
        (
            nonvar(W),
            (
                atom_chars(X,XL),
                atom_chars(W,WL),
                length(WL,L),
                (
                    (
                        L==L1,
                        (
                            (
                                word(W,_),
                                (
                                    (
                                        W==X,
                                        write('You won!'),
                                        nl
                                    );
                                    (
                                        W\==X,
                                        correct_letters(WL,XL,CL),
                                        write('Correct letters are: '),
                                        write(CL),
                                        nl,
                                        correct_positions(WL,XL,CP),
                                        write('Correct letters in correct positions are: '),
                                        write(CP),
                                        nl,
                                        write('Remaining Guesses are '),
                                        G1 is G-1,
                                        write(G1),
                                        write('.'),
                                        nl,nl,
                                        play_word_guess(L1,G1,X)
```

play_word_guess(L1,G,X) where L1 is the length entered by the user. G is the number of guesses remaining. X is the first word in the knowledge base in the category desired by the user.

play_word_guess/3 prompts the user to guess a word "W" until his guesses run out. Then, the predicate checks if the word the user entered is not a variable, has the length L1 and is in the knowledge base. The predicate compares X to W. Then, the predicate determines common letters by using the correct_letters/3 predicate. Lastly, the predicate correct_positions/3 checks if these correct letters are in the correct positions.

```
is_category(C):-
    word(_,C).
```

is_category/1 : The second argument in the fact ‘word’ is always a category, so if the argument C in is_category happens to be the second argument in word, then C is a category.

```
categories(L):-
    setof(C,is_category(C),L).
```

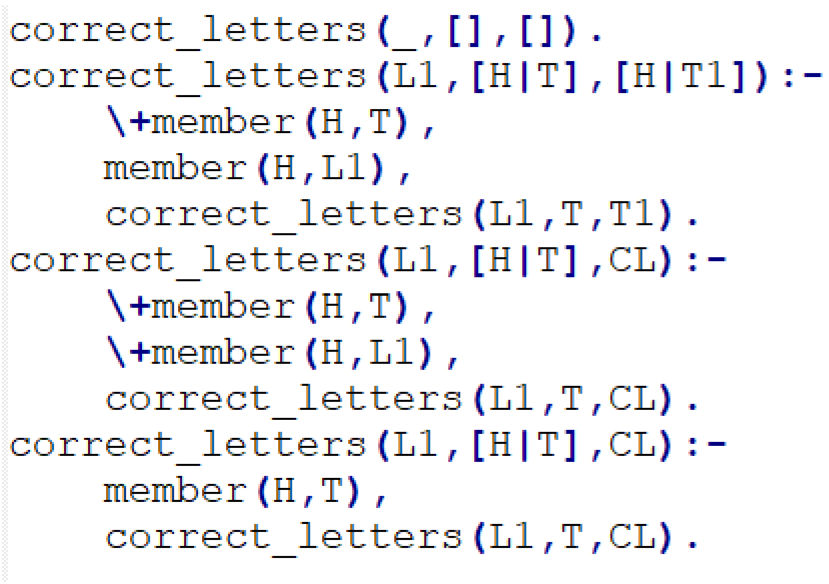
Categories/1: this predicate succeeds if L is also the third argument of the setof predefined predicate. We used setof to get a list of all available categories in the KB by setting the first argument as the variable C and the second argument as the condition that C must be a category by using the is_category predicate.

```
available_length(L):-
    word(X,_),
    atom_chars(X,X1),
    length(X1,L),!.
```

available_length/1 to check if L is available, we use the fact word(X,C) where X represents the word itself then we use the predefined predicate atom_chars(X,X1) where X1 is a list of the characters of the word X. For example, atom_chars(abc,X1). Outputs X1 = [a,b,c]. Then, we check if the length of this list is equal to the length L. finally, we used the cut to prevent backtracking as we only wish to compare the length L with the first word in the KB.

```
pick_word(W,L,C):-
    word(X,C),
    atom_chars(X,X1),
    length(X1,L),
    W = X.
```

pick_word/3: we used the fact word to generate X such that this X is of category C and the number of its characters is equal to length L and this was done by splitting it characters into a list by the predefined atom_chars predicate and comparing its length to L by the predefined predicate length. Finally, matching W to X.



```
correct_letters(_,[],[]).
correct_letters(L1,[H|T],[H|T1]):-
    \+member(H,T),
    member(H,L1),
    correct_letters(L1,T,T1).
correct_letters(L1,[H|T],CL):-
    \+member(H,T),
    \+member(H,L1),
    correct_letters(L1,T,CL).
correct_letters(L1,[H|T],CL):-
    member(H,T),
    correct_letters(L1,T,CL).
```

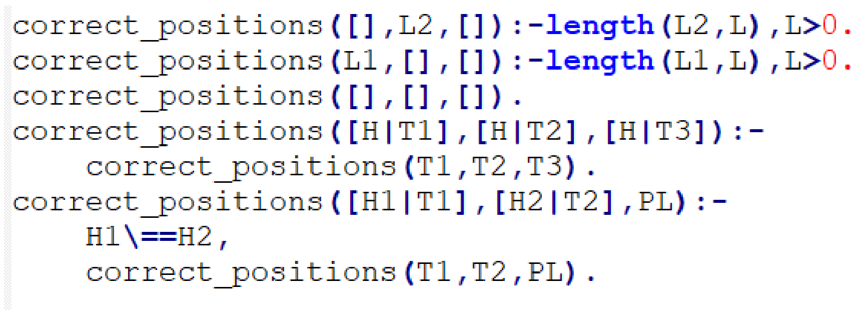
correct_letters/3: has a base case where the third argument (output) is an empty list if the second argument is an empty list.

We loop through the second list which is the second argument and we fix the first argument.

Case 1 (second correct_letters): if the head of the second list is not repeated in the its tail and at the same time is a member in the first argument, the head of the output which is the third argument is the head of the second argument then we repeat the same process by taking the tail of second argument as the second argument.

Case 2 (third correct_letters): we check that the head is not a member of its tail (not repeated) and at the same time the head is not a member of the first list (first argument). if both conditions are true then we will repeat the same process with taking the tail of the second list instead of the second list without changing the output (third argument).

Case 3: if the head of the second list is a member of its tail we will repeat the same process without changing the output but taking the tail of the second list as the second argument,



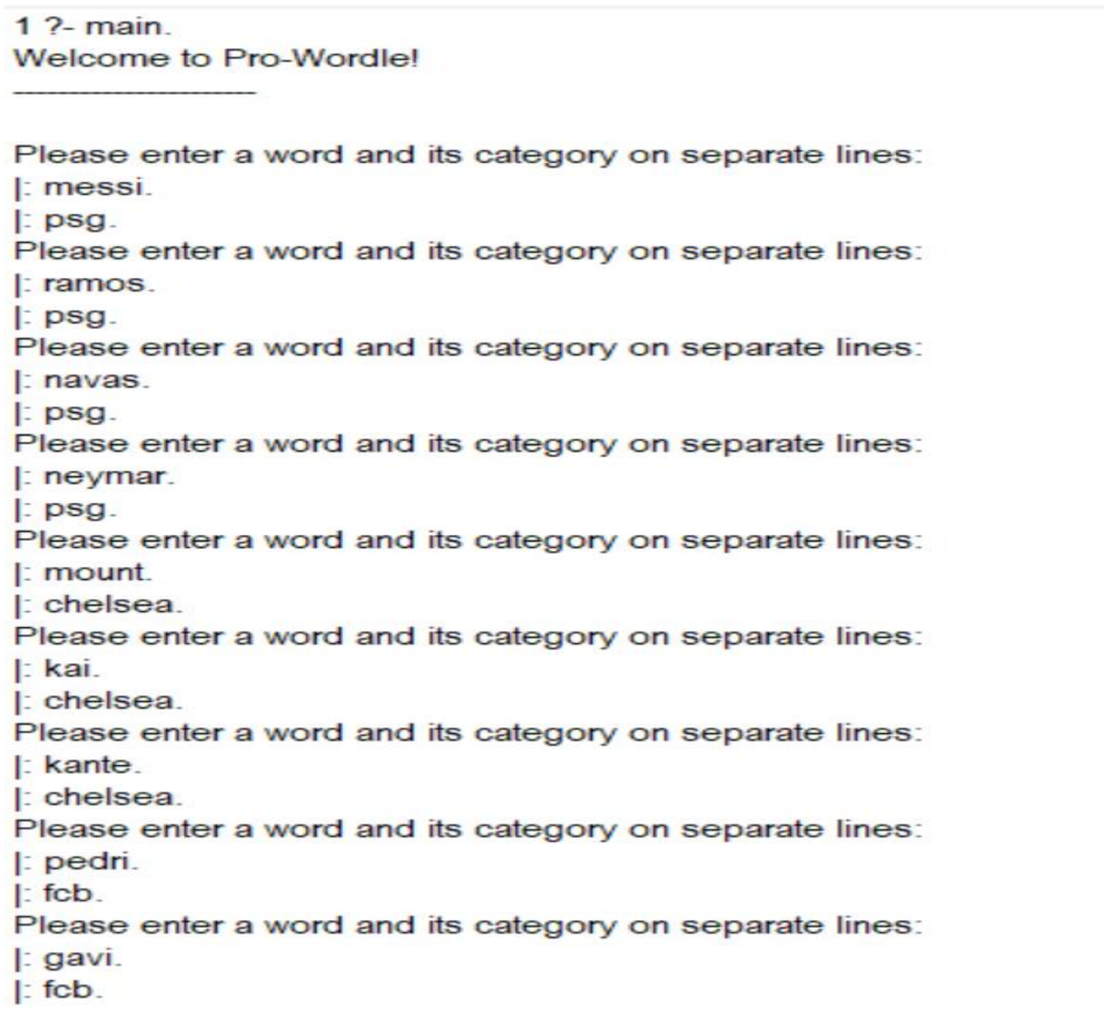
```
correct_positions([],L2,[]):-length(L2,L),L>0.
correct_positions(L1,[],[]):-length(L1,L),L>0.
correct_positions([],[],[]).
correct_positions([H|T1],[H|T2],[H|T3]):-
    correct_positions(T1,T2,T3).
correct_positions([H1|T1],[H2|T2],PL):-
    H1\==H2,
    correct_positions(T1,T2,PL).
```

Correct_position/3 we loop through both lists the first argument and the second argument. We check if the head of both arguments are the same. If it is, the head of the output list which is the third argument is the H of any of them. If not, the continue looping through both lists by taking their tails as the arguments for the recursion.

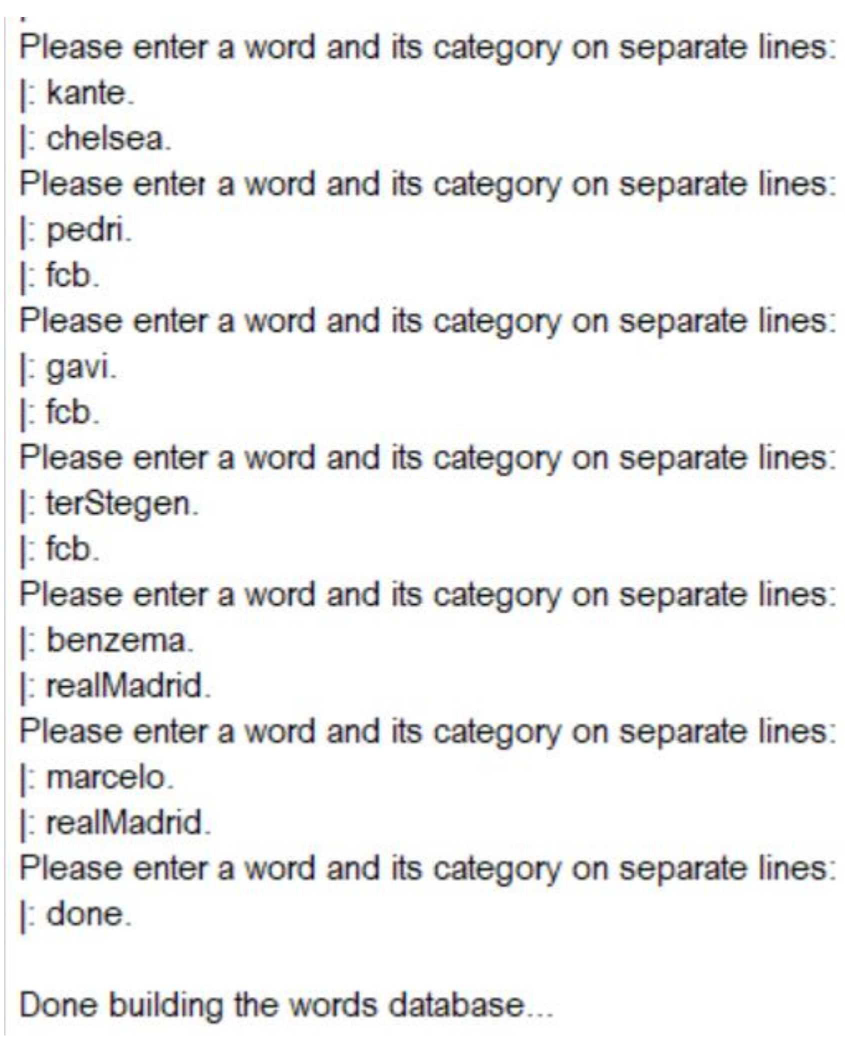
The recursion stops when any of the lists become empty then the 3rd list (3rd argument) becomes empty list.

1st Sample run (where the user wins):

1) Building the knowledge base:



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

Please enter a word and its category on separate lines:
|: messi.
|: psg.
Please enter a word and its category on separate lines:
|: ramos.
|: psg.
Please enter a word and its category on separate lines:
|: navas.
|: psg.
Please enter a word and its category on separate lines:
|: neymar.
|: psg.
Please enter a word and its category on separate lines:
|: mount.
|: chelsea.
Please enter a word and its category on separate lines:
|: kai.
|: chelsea.
Please enter a word and its category on separate lines:
|: kante.
|: chelsea.
Please enter a word and its category on separate lines:
|: pedri.
|: fcb.
Please enter a word and its category on separate lines:
|: gavi.
|: fcb.
```



```
.
Please enter a word and its category on separate lines:
|: kante.
|: chelsea.
Please enter a word and its category on separate lines:
|: pedri.
|: fcb.
Please enter a word and its category on separate lines:
|: gavi.
|: fcb.
Please enter a word and its category on separate lines:
|: terStegen.
|: fcb.
Please enter a word and its category on separate lines:
|: benzema.
|: realMadrid.
Please enter a word and its category on separate lines:
|: marcelo.
|: realMadrid.
Please enter a word and its category on separate lines:
|: done.

Done building the words database...
```

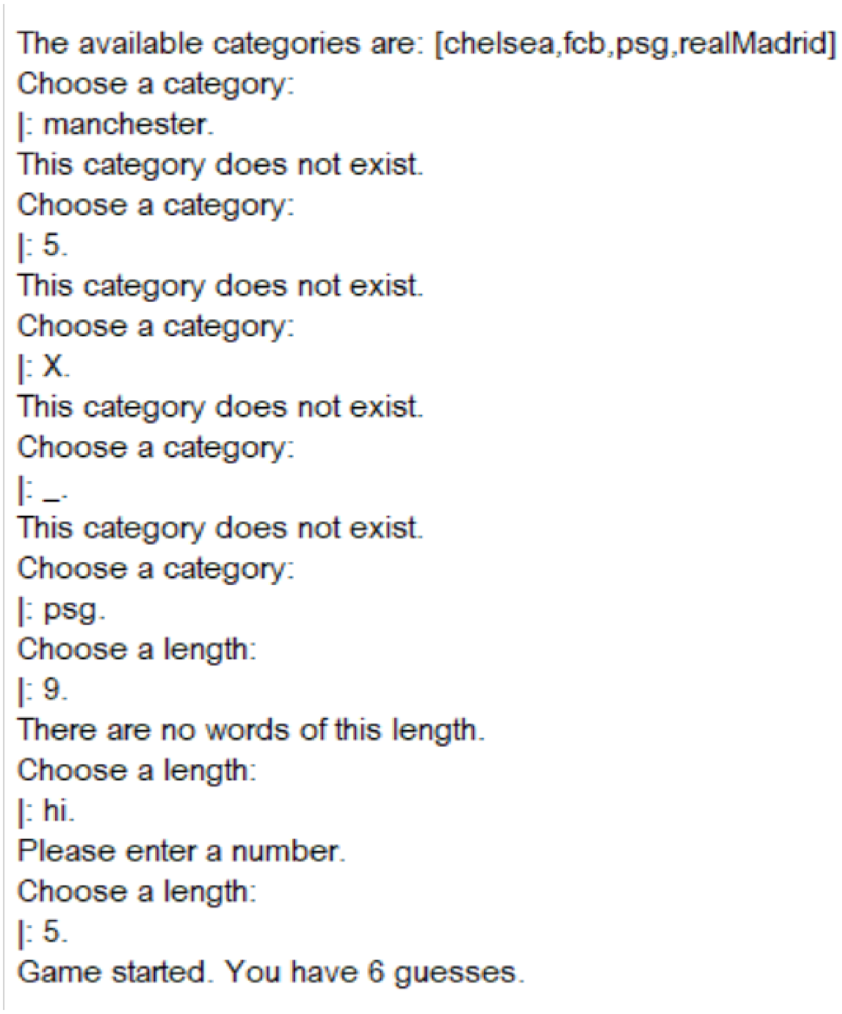
```
word(messi, psg).

word(ramos, psg).

word(navas, psg).

word(neymar, psg).

word(mount, chelsea).

word(kai,chelsea).

word(kante, chelsea).

word(pedri, fcb).

word(gavi,fcb).
```

word(terStegen, fcb).

word(benzema, realMadrid).

word(marcelo, realMadrid).

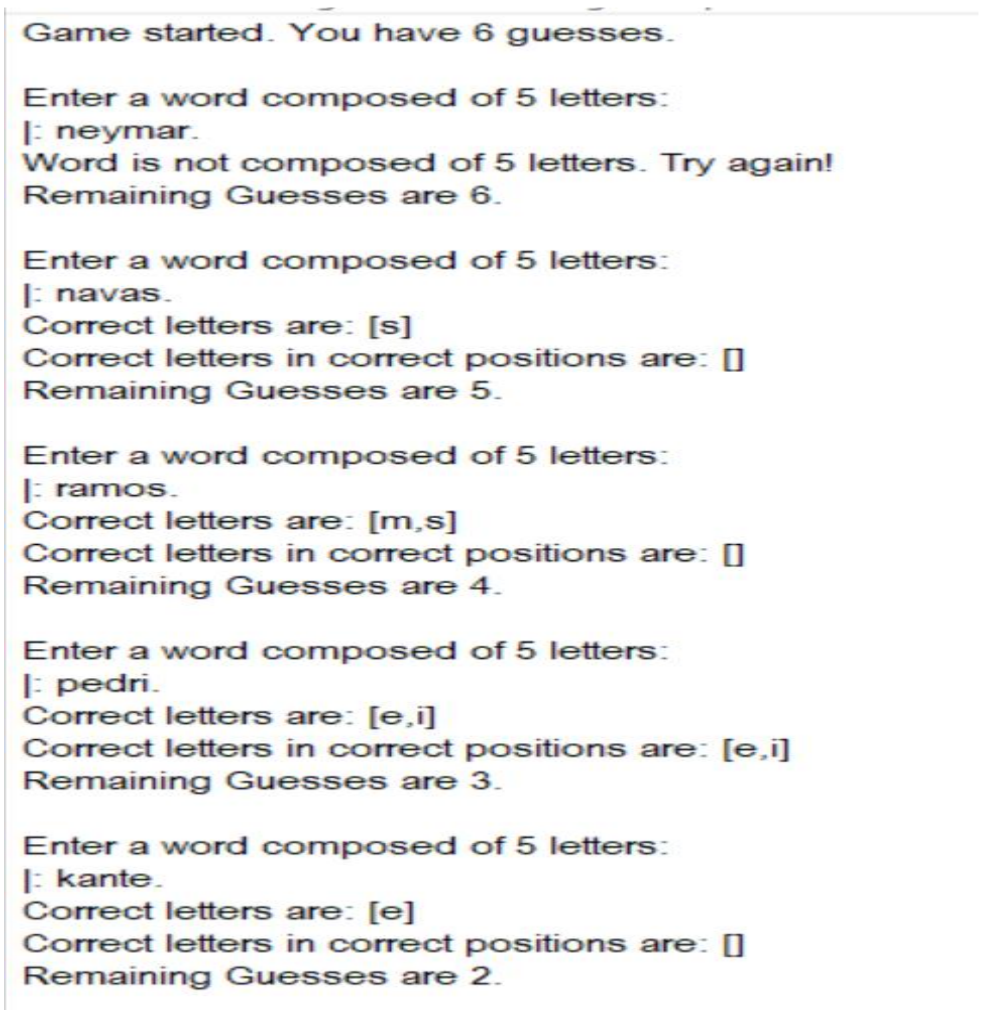
2) Choosing a category and choosing a length:



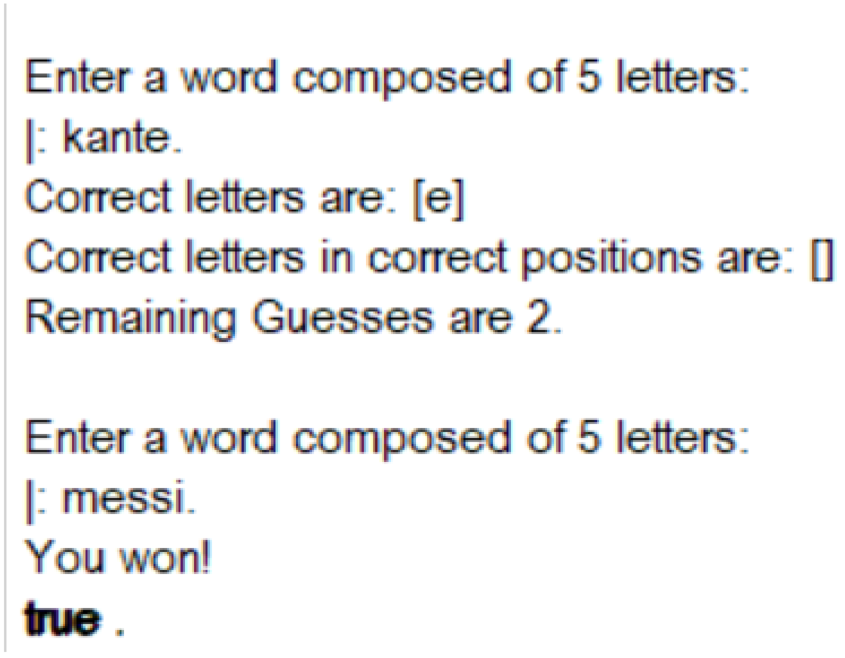
```
The available categories are: [chelsea,fcb,psg,realMadrid]
Choose a category:
|: manchester.
This category does not exist.
Choose a category:
|: 5.
This category does not exist.
Choose a category:
|: X.
This category does not exist.
Choose a category:
|: _.
This category does not exist.
Choose a category:
|: psg.
Choose a length:
|: 9.
There are no words of this length.
Choose a length:
|: hi.
Please enter a number.
Choose a length:
|: 5.
Game started. You have 6 guesses.
```

3) The user guessing the word:

The word picked by the game in order for the user to win is messi.



```
Game started. You have 6 guesses.

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

Enter a word composed of 5 letters:
|: navas.
Correct letters are: [s]
Correct letters in correct positions are: []
Remaining Guesses are 5.

Enter a word composed of 5 letters:
|: ramos.
Correct letters are: [m,s]
Correct letters in correct positions are: []
Remaining Guesses are 4.

Enter a word composed of 5 letters:
|: pedri.
Correct letters are: [e,i]
Correct letters in correct positions are: [e,i]
Remaining Guesses are 3.

Enter a word composed of 5 letters:
|: kante.
Correct letters are: [e]
Correct letters in correct positions are: []
Remaining Guesses are 2.
```

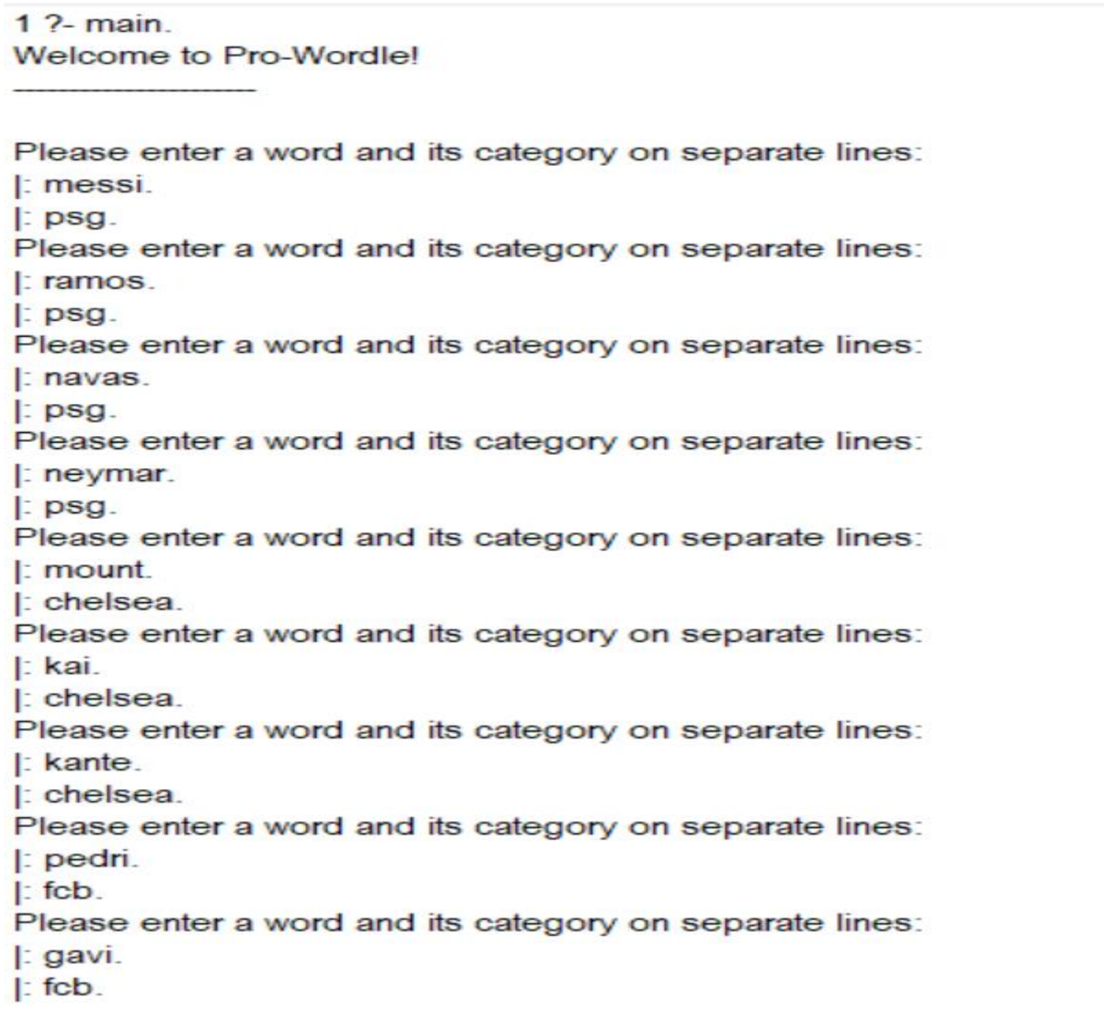


```
Enter a word composed of 5 letters:
|: kante.
Correct letters are: [e]
Correct letters in correct positions are: []
Remaining Guesses are 2.
```

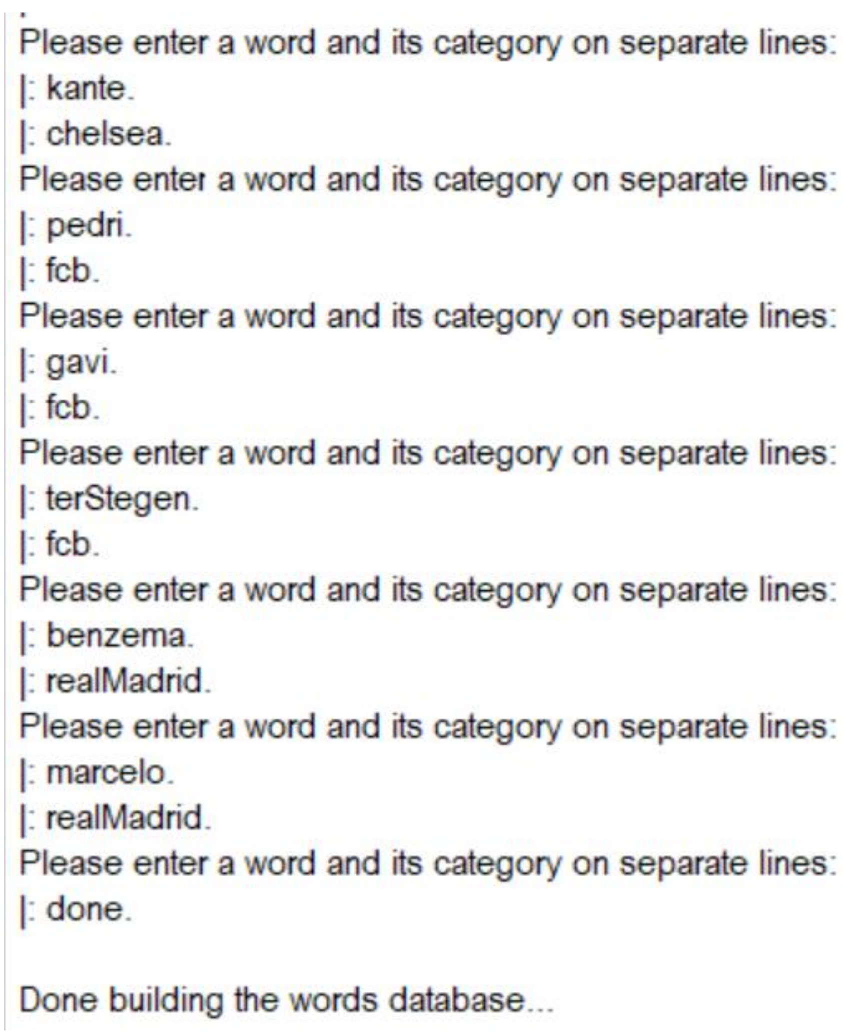
```
Enter a word composed of 5 letters:
|: messi.
You won!
true .
```

2nd Sample run (where the user loses):

1) Building the knowledge base:



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

Please enter a word and its category on separate lines:
|: messi.
|: psg.
Please enter a word and its category on separate lines:
|: ramos.
|: psg.
Please enter a word and its category on separate lines:
|: navas.
|: psg.
Please enter a word and its category on separate lines:
|: neymar.
|: psg.
Please enter a word and its category on separate lines:
|: mount.
|: chelsea.
Please enter a word and its category on separate lines:
|: kai.
|: chelsea.
Please enter a word and its category on separate lines:
|: kante.
|: chelsea.
Please enter a word and its category on separate lines:
|: pedri.
|: fcb.
Please enter a word and its category on separate lines:
|: gavi.
|: fcb.
```



```
.
Please enter a word and its category on separate lines:
|: kante.
|: chelsea.
Please enter a word and its category on separate lines:
|: pedri.
|: fcb.
Please enter a word and its category on separate lines:
|: gavi.
|: fcb.
Please enter a word and its category on separate lines:
|: terStegen.
|: fcb.
Please enter a word and its category on separate lines:
|: benzema.
|: realMadrid.
Please enter a word and its category on separate lines:
|: marcelo.
|: realMadrid.
Please enter a word and its category on separate lines:
|: done.

Done building the words database...
```

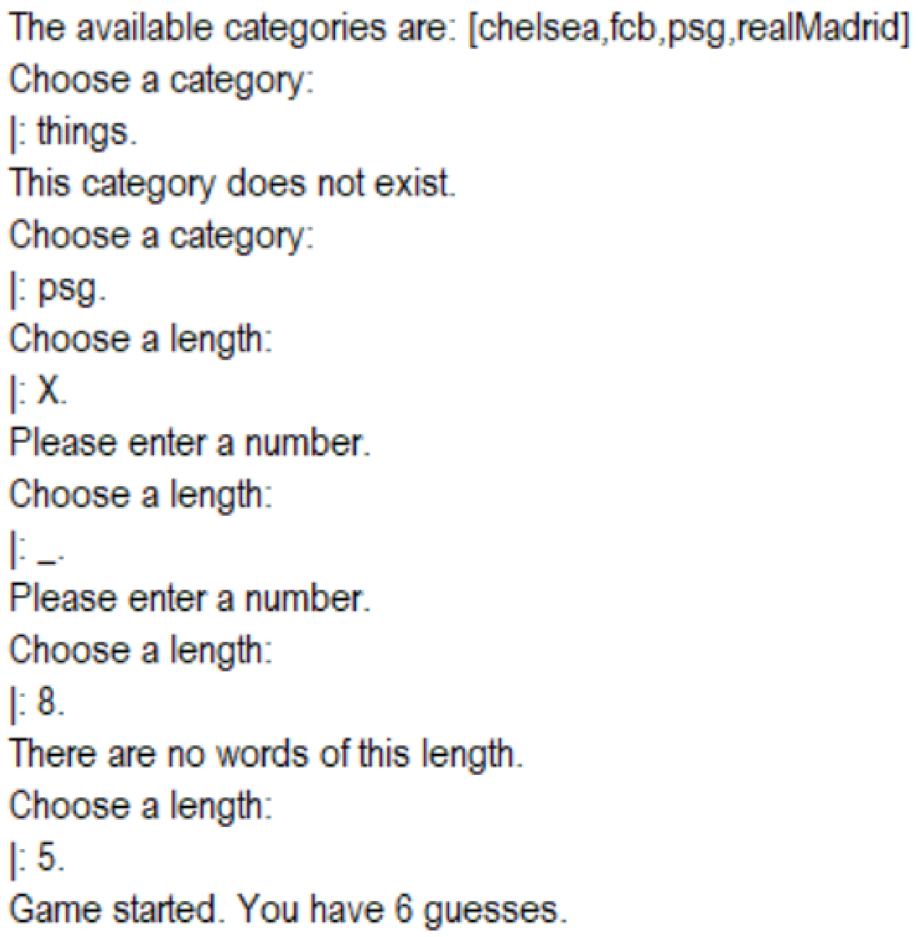
```
word(messi, psg).

word(ramos, psg).

word(navas, psg).

word(neymar, psg).

word(mount, chelsea).

word(kai,chelsea).

word(kante, chelsea).

word(pedri, fcb).

word(gavi,fcb).
```

word(terStegen, fcb).

word(benzema, realMadrid).

word(marcelo, realMadrid).

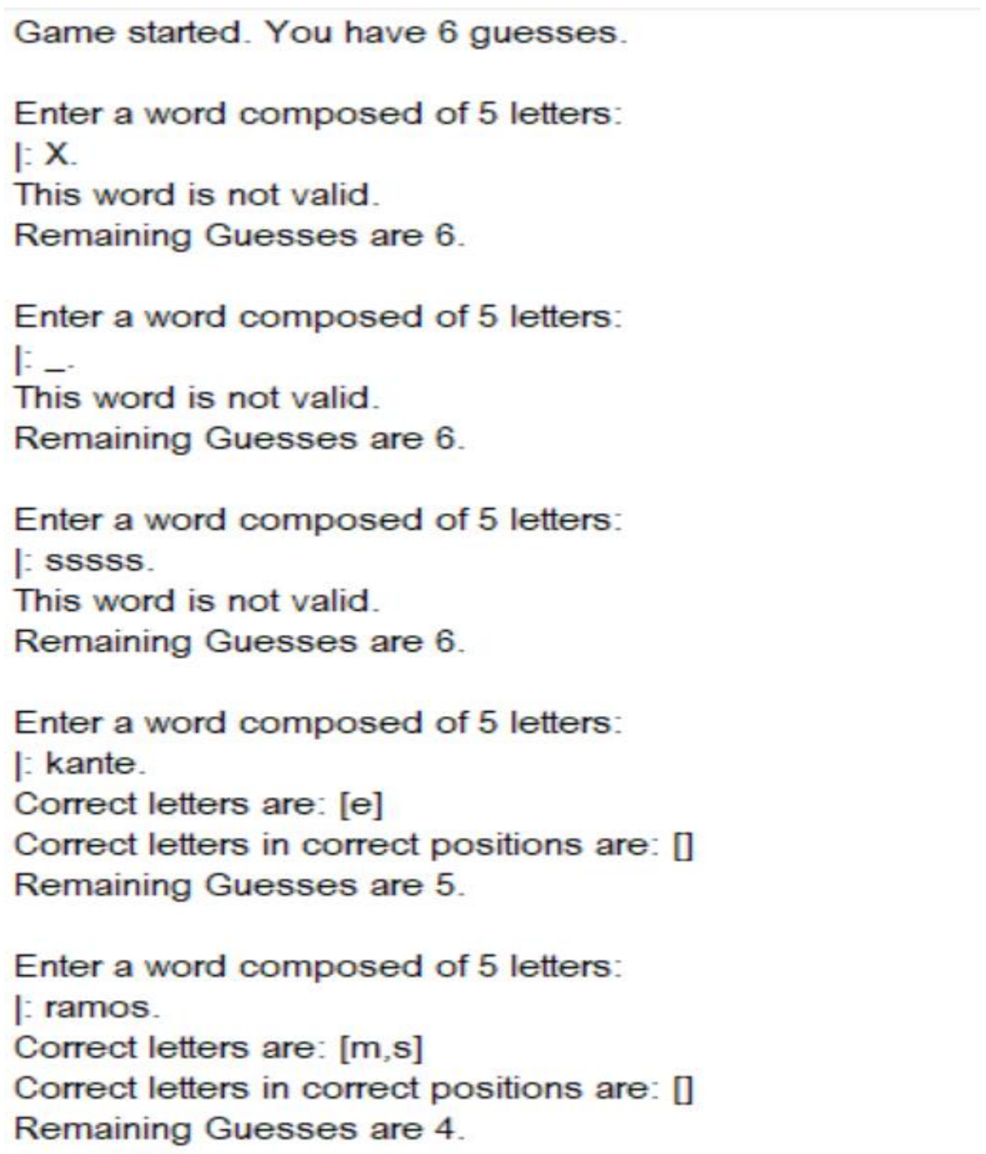
2) Choosing a category and choosing a length:



```
The available categories are: [chelsea,fcb,psg,realMadrid]
Choose a category:
|: things.
This category does not exist.
Choose a category:
|: psg.
Choose a length:
|: X.
Please enter a number.
Choose a length:
|: _.
Please enter a number.
Choose a length:
|: 8.
There are no words of this length.
Choose a length:
|: 5.
Game started. You have 6 guesses.
```

3) The user guessing the word:

The word picked by the game in order for the user to win is messi.



```
Game started. You have 6 guesses.

Enter a word composed of 5 letters:
|: X.
This word is not valid.
Remaining Guesses are 6.

Enter a word composed of 5 letters:
|: _.
This word is not valid.
Remaining Guesses are 6.

Enter a word composed of 5 letters:
|: sssss.
This word is not valid.
Remaining Guesses are 6.

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

Enter a word composed of 5 letters:
|: ramos.
Correct letters are: [m,s]
Correct letters in correct positions are: []
Remaining Guesses are 4.
```

```
Enter a word composed of 5 letters:
|: ramos.
Correct letters are: [m,s]
Correct letters in correct positions are: []
Remaining Guesses are 4.

Enter a word composed of 5 letters:
|: navas.
Correct letters are: [s]
Correct letters in correct positions are: []
Remaining Guesses are 3.

Enter a word composed of 5 letters:
|: marcelo.
Word is not composed of 5 letters. Try again!
Remaining Guesses are 3.

Enter a word composed of 5 letters:
|: pedri.
Correct letters are: [e,i]
Correct letters in correct positions are: [e,i]
Remaining Guesses are 2.

Enter a word composed of 5 letters:
|: mount.
Correct letters are: [m]
Correct letters in correct positions are: [m]
Remaining Guesses are 1.
```

```
Enter a word composed of 5 letters:
|: mount.
Correct letters are: [m]
Correct letters in correct positions are: [m]
Remaining Guesses are 1.

Enter a word composed of 5 letters:
|: neymar.
Word is not composed of 5 letters. Try again!
Remaining Guesses are 1.

Enter a word composed of 5 letters:
|: Y.
This word is not valid.
Remaining Guesses are 1.

Enter a word composed of 5 letters:
|: _.
This word is not valid.
Remaining Guesses are 1.

Enter a word composed of 5 letters:
|: fffff.
This word is not valid.
Remaining Guesses are 1.

Enter a word composed of 5 letters:
|: ramos.
You lost!
true .
```