MINI PROLOG

PART A

data types:-

Term :- have 2 constructors Variable and Constant both take a String as a parameter <have different names allowed us to differentiate between variables and constants >

Prdct :-have Only one constructor Predicate that takes a string for the name of the predicate and a list of terms

Goals :- is of type type and not data because it is composed of builtin type list .Goals is composed of list of predicates

Rle :-have only one constructor Rule that takes a predicate

(head of rule) and list of predicates<Goals>
Solution:-have 2 constructors Yes that takes a list of terms as a parameter and No that have no parameters

PART B
inunify:- takes 2 lists of type term and outputs a list of list in the form of [[Variable”X”,Constant “”Name”], [Variable”Y” ,Constant”Name2”]]< It pairs the inputed lists>
check:-checks if the inputed two lists of terms are identical if they are constants and if they are variables they are evaluated to true without any checks same for variable &&

constant and constant&variable
unify:-takes a predicate and a list of predicates<data base> and checks if the input predicate unifies with any of the predicates in the database if yes it returns a list of terms which is the values it might have in order to unify with the database if the inputed predicate contains variables otherwise if the inputed list of variables in the predicate query consists of constants then the output would be an empty list if the fact<queried predicate> was found in the database
unifyWithHead :- outputs unifies out put in the form of solution

e.g.:-unifyWithHead
(Predicate"game"[Variable"A",Con
stant"2",Constant"3"])
[Predicate"game"[Constant"1",Con
stant"2",Constant"3"],Predicate"
game"[Constant"4",Constant"3",Co
nstant"5"],Predicate"game"[Const
ant"6",Constant"2",Constant"7"]]
Yes [Variable "A",Constant "1"]

PART C

search:- takes as an input 2 list and searches for the elements that are in the 1st list but not in the 2nd one<using contain as a helper> it returns a list with all differences

contains:- takes as an input 2 parameters a term and list of

terms searches for the term in the list return true if the element was found and false otherwise

getSimilarties:- takes a list of lists as an input and calls merge on the first 2 lists of the input list it removes duplicated elements in different lists from a list of lists of termes

searchin:- takes 2 lists of goals and checks if an element found in first list is found in the second list if yes then the next element in the 2nd list is accumulated to the output list else continue searching in the rest of the list the returned

list is a list of the updated name of the variables that the user have used in his/her query if any

unifyPredicate:- takes as an input a list of terms and a list of predicates representing database it matches the variables of the input with that of the data base and unifies with them to find the answer

applySolSubBody:-takes a list of valid solutions and list of Goals calls unifyPredicate and outputs a list of the goals that have to be achieved in order to get the valid list of solutions given in the query

```
applySolSubBody(Yes [Variable "X",Constant "slim",Variable
```

```
"X",Constant "salah"])[(Predicate "male" [Variable "X"]),
(Predicate "Parent" [Variable "X",Variable "Y"])]
[Predicate "male" [Constant "slim"],Predicate "male" [Constant
"salah"],Predicate "Parent" [Constant "slim",Variable
"Y"],Predicate "Parent" [Constant "salah",Variable "Y"]]
```

PART D

callUnifyPredicate:- takes a list of terms and a list of predicates and outputs a list of list of unified predicates <used to unify the list of goals>

isfact:-takes a list of terms and returns true if the list contains no variables

relevantRule:- takes as an input the query entered by the user and a list of a list of predicates representing data base

searchrelevantRule:- takes as an input a list of Predicates and the queried predicate and outputs a list of goals that are

relevant to the query
flatten:-flattens a list up till 2nd level with the help of flat
flat:- flattens a list of lists
searchFilterFact:- searches for the relevant facts that are needed in order to get a solution that evaluates to true
checkConstant:-returns true if the given list is composed of only constants
try:- takes the user`s query and the database and outputs a list of all possible answer that will evaluate to true
eg
try(Predicate"father"[Variable”X”,Variable”Y”])
[[[Predicate"male"[Constant"slim

"]]],
[[Predicate"Parent"[Constant"slim",Constant"sara"]]],
[[Predicate"father"[Variable”X”,Variable”Y”]],
[(Predicate"male"[Variable"X"]),Predicate"Parent"[Variable"X",Variable"Y"]]]]
[Variable "X",Constant "slim",Variable "Y",Constant "sara"]

final:-takes the same input as try and outputs the alternative variable used by the user

e.g.final(Predicate"father"[Variable"A",Variable"B"])
[[[Predicate"male"[Constant"slim"]]],
[[Predicate"male"[Constant"salah

```
"]]],
[[Predicate"Parent"[Constant"sal
ah",Constant"youmna"]]],
[[Predicate"Parent"[Constant"sli
m",Constant"sara"]]],
[[Predicate"father"[Variable"X",
Variable"Y"]],
[(Predicate"male"[Variable"X"]),
Predicate"Parent"[Variable"X",Va
riable"Y"]]]]
[Variable "A",Variable
"X",Variable "B",Variable "Y"]
```

get:-takes list of goals <goals that match the query entered by user> and list of predicates that resembles the database and calles unify on the whole list it outputs a list of valid solutions for the entire body of

rule removeEmptyList:-removes empty lists from a list of lists merge:- checked if the first list is a subset of the second list if so then it returns the second list otherwise it returns the differences replin:-matches the variables entered by the user to that in the database

```
replin 12 13[12,13,14,14,12] [13,13,14,14,13]
```

replacement:- recurs over the list of terms and calls replin on each term in order to change each X in the database to A entered by the user e.g.:

```
replacment[Variable "Y",Variable"B",Variable
```

```
"X",Variable "A"][Variable
"X",Constant "salah",Variable
"Y",Constant "youmna",Variable
"X",Constant "slim",Variable
"Y",Constant "sara"]
[Variable "A",Constant
"salah",Variable "B",Constant
"youmna",Variable "A",Constant
"slim",Variable "B",Constant
"sara"]
```

about:-

```
 about(Predicate"father"[Constant"slim",Variable"B"])
[[[Predicate"male"[Constant"slim"]]],
[[Predicate"male"[Constant"salah"]]],
[[Predicate"Parent"[Constant"salah",Constant"youmna"]]],
[[Predicate"Parent"[Constant"slim",Constant"sara"]]],
[[Predicate"father"[Variable"X",Variable"Y"]],
[(Predicate"male"[Variable"X"]),Predicate"Parent"[Variable"X",Va
riable"Y"]]]]
[Predicate "father" [Constant "slim",Variable "B"],Predicate
"male" [Constant "slim"],Predicate "Parent" [Constant
"slim",Variable "B"]]
```

allSolutions:- does the same job as applySolSubBody but on list eg.

```
allSolutions(Predicate"father"[V
```

```
ariable"A",Variable"B"])
[[[Predicate"male"[Constant"slim
"]]],
[[Predicate"male"[Constant"salah
"]]],
[[Predicate"Parent"[Constant"sal
ah",Constant"youmna"]]],
[[Predicate"Parent"[Constant"sli
m",Constant"sara"]]],
[[Predicate"father"[Variable"X",
Variable"Y"]],
[(Predicate"male"[Variable"X"]),
Predicate"Parent"[Variable"X",Va
riable"Y"]]]]
Yes [Variable "A",Constant
"salah",Variable "B",Constant
"youmna",Variable "A",Constant
"slim",Variable "B",Constant
"sara"]
```

```
   allSolutions(Predicate"father"[Constant"slim",Constant"sara"])
[[[Predicate"male"[Constant"slim"]]],[[Predicate"male"[Constant"salah"]]],
[[Predicate"Parent"[Constant"salah",Constant"youmna"]]],
[[Predicate"Parent"[Constant"slim",Constant"sara"]]],
[[Predicate"father"[Variable"X",Variable"Y"]],
[(Predicate"male"[Variable"X"]),Predicate"Parent"[Variable"X",Variable"Y"]]]]
Yes []
```