(1) Here is a program designed to find solutions to car-buying problem:

```
PREDICATES
           can_buy(symbol, symbol)
           person(symbol)
           car(symbol)
           likes(symbol, symbol)
           for_sale(symbol)
CLAUSES
           can_buy(X,Y):-
                     person(X),
                     car(Y),
                     likes(X,Y),
                     for_sale(Y).
           person(mohamed).
           person(saher).
           person(samia).
           person(osama).
           car(lemon).
           car(hot_rod).
           likes(mohamed, hot_rod).
           likes(saher, pizza).
           likes(samia, tennis).
           likes(osama, tennis).
           for_sale(pizza).
           for_sale(lemon).
           for_sale(hot_rod).
```

write this program and satisfy the following external goals:

a) can_buy(Who, What).
.....
.....
.....

b) can_buy(saher, What).
.....
.....
.....

c) can_buy(mohamed, What).
.....
.....
.....

d) can_buy(Who, hot_rod).
.....
.....
.....

e) likes(mohamed, hot_rod).
.....
.....
.....

f) for_sale(hot_rod).
.....
.....
.....

(2) The following parents example demonstrates how the anonymous variable is used:

```
PREDICATES
   male(symbol)
   female(symbol)
   parent(symbol, symbol)
CLAUSES
   male(ayman).
   male(blal).
   female(samr).
   female(fatma).
   parent(ayman,blal).
   parent(samr,blal).
   parent(blal,fatma).
```

write this program and satisfy the following external goals:

(a) Goal: parent(Parent, _)
.....
.....
.....

(b) owns(_, shoes).
.....
.....
.....

(c) eats(_).
.....
.....
.....

(3) The following program will yield a type error when run.
Correct it:

```
DOMAINS
   product = integer
PREDICATES
   add_em_up(sum,sum,sum)
   multiply_em(product,product,product)
CLAUSES
   add_em_up(X,Y,Sum):-
             Sum=X+Y.
   multiply_em(X,Y,Product):-
             Product=X*Y.
```

(a) Given the goal :

```
add_em_up(32, 54, Sum).
```

the output is:

.....
.....
.....

(b) If you need to double the product of 31 and 17, so you write the goal:

.....
.....
.....
.....
.....
.....

(4) The following program defines isletter predicate, which when given the goals

```
isletter('%').
isletter('Q').
```

will return No and Yes, respectively.

```
PREDICATES
    isletter(char)
CLAUSES
    isletter(Ch):-
        'a' <= Ch,
        Ch <= 'z'.
    isletter(Ch):-
        'A' <= Ch,
        Ch <= 'Z'.
```

write this program and try each of the following goals:

a) isletter('x').

.....
.....

b) isletter('2').
.....
.....

c) isletter("hello").
.....
.....

d) isletter(a).
.....
.....
e) isletter(X).
.....
.....

(5) Consider the following program, which contains facts about the names and ages of some players in a racquet club:

```
DOMAINS
   child = symbol
   age   = integer
PREDICATES
   player(child, age)
CLAUSES
   player(esam,9).
   player(amir,10).
   player(fatma,9).
   player(sumia,9).
```

What the solutions for the following complex goal:

```
player(Person1, 9),
player(Person2, 9),
Person1 <> Person2.
```

.....
.....
.....

(6) Write a program to count integer numbers from 1 to 999.
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(7) The following uses repeat predicate to keep accepting characters and printing them until the user presses Enter:

```
PREDICATES
   repeat
   typewriter
CLAUSES
   repeat.
   repeat:-repeat.
   typewriter:- repeat,
   readchar(C),
   write(C), C='\r',!.
```

.....
.....
.....
.....
.....
.....
.....
.....
.....

(8) The following not tail recursion program to compute the factorial for integer number:

```
PREDICATES
   factorial(integer,real)
CLAUSES
   factorial(1,1):-!.
   factorial(X,FactX):-
       Y=X-1,
       factorial(Y,FactY),
       FactX = X*FactY.
```

Write out the output of the following external goals:

a) factorial(4,Fact).

.....
.....
.....

b) factorial(Fact, 3).

.....
.....
.....

c) factorial(4,24).

.....
.....
.....

(9) Correct the following tail recursive program to count numbers and run it:

```
PREDICATES
   cutcount2(real)
   cutcount3(real)
   check(real)
CLAUSES
   cutcount2(X):-
       X>=0,
       write(X), nl,
       NewX = X + 1,
       cutcount2(NewX).
   cutcount2(_):-
       write("X is negative.").
   cutcount3(X):-
       write(X), nl,
       NewX = X+1,
       check(NewX),
       !,
       cutcount3(NewX), nl.
   check(Z):-Z >= 0.
   check(Z):-Z
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(10) Complete the following tail recursive program to compute factorial:

```
PREDICATES
  factorial(integer,real)
  factorial(integer,real,integer,real)
CLAUSES
  factorial(N,FactN):-
       factorial(N,    ,1, ).
  factorial(N,FactN,   ,FactN):-!.
       factorial(N,Fact ,I,P):-
       NewI = I+   ,
       NewP = P  NewI,
  factorial(N, FactN, New  , NewP).□□
```

.....
.....
.....

(11) Write a tail recursive program that accepts a number as input and can end in either of two ways. It will start multiplying the number by itself over and over until it either reaches 81 or reaches a number greater than 100. If it reaches 81, it will print yes; if it exceeds 100, it will print no.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(12) Here's a program that scans a list of numbers and copies them, leaving out the negative numbers:

```
DOMAINS
   list = integer*
PREDICATES
   discard_negatives(list, list)
CLAUSES
   discard_negatives([], []).
   discard_negatives([H|T],ProcessedTail):-
        H < 0,
        !,
       discard_negatives(T,ProcessedTail).
   discard_negatives([H|T],[H|ProcessedTail]):-
        discard_negatives(T, ProcessedTail).
```

a) The solutions for the goal
  discard_negatives([-20, 5, -3, 68], X)

.....
.....
.....

b) Rewrite this program to scans a list of numbers and copies them, leaving out the non negative numbers:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(13) The following program uses findall predicate to print the average age of a group of people, correct and run it.

```
DOMAINS
   name,address = str
   age = int
   list = age
PREDICATES
   person(name, address, age)
   sumlist(list, age, integer)
CLAUSES
   sumlist([], 0  ).
   sumlist([H,T],Sum,N):-
      sumlist(T,S1,N1),
      Sum=H+S1, N=1+N1.
   person("Ahmed Taher", "Mansoura Univ. Bulding", 15).
   person("Osama Salh", "Apt. 22, 21st Street", 15).
   person("Sulaf Taher", "Zamlik, Quads st.", 17).
GOALS
   findall(Age,person(_, _, Age),L),
   sumlist(L,Sum,N),
   Ave = Sum/N,
   write("Average=", Ave),nl.
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....
.....
.....
.....

Write the output for this program:

.....
.....
.....
.....
.....
.....

(14) Write a predicate, oddlist, that takes two arguments. The first argument is a list of integers, while the second argument returns a list of all the odd numbers found in the first list.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(15) Write a predicate, real_average, that calculates the average value of all the elements in a list of reals.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(16) The following program uses random/1 to select three names from five at random. Correct and run it.

```
PREDICATES
   person(integer, symbol)
   rand_int_1_5(integer)
   rand_person(integer)
CLAUSES
   person(1,farid).
   person(2,tark).
   person(3,mariam).
   person(4,samy).
   person(5,samh).
   rand_int_1_5(X):-
      random(Y),
      X=Y*4+1.
   rand_person(0):-!.
   rand_person(Count):-
      rand_int_1_5(N),
      person(N,Name),
      write(Name),nl,
      NewCount=Count-1,
      rand_person(NewCount).
GOAL
   rand_person(3)
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(17) Use the trigonometric functions in Turbo Prolog to display a table of sine, cosine, and tangent values on the screen. The left column of the table should contain angle values in degrees, starting at 0 degrees and continuing to 360 degrees in steps of 15 degrees.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(18) The following program writes out the elements in a list with no more than five elements per line.

```
DOMAINS
   integerlist = integer*
PREDICATES
   writelist(integerlist)
   write5(integerlist,integer)
CLAUSES
   writelist(NL):- nl,
      write5(NL,0),nl.
   write5(TL,5):-!, nl,
      write5(TL, 0).
   write5([H|T],N):-!,
      write(H," "),
      N1=N+1, write5(T,N1).
   write5([],_).
After you compile the aove program try the following goal:

writelist([2,4,6,8,10,12,14,16,18,20,22]).

the output is:
```

.....
.....
.....
.....
.....
.....

(19) Write a Turbo Prolog program that, when given a list of addresses contained in the program as clauses of the form:

```
address("Samir Ramdan","14 Mohamed Fathi St", "Mansoura",
27240).
```

displays the addresses in a form suitable for mailing labels, such as:

```
Samir Ramdan
14 Mohamed Fathi St
Mansoura
27240
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(20) In the following program, frontchar is used to define a predicate that changes a string to a list of characters.

```
DOMAINS
   charlist = char*
PREDICATES
   string_chlist(string, charlist)
CLAUSES
   string_chlist("", []):-!.
   string_chlist(S, [H|T]):-
      frontchar(S,H,S1),
      string_chlist(S1,T).
```

Try the goal string_chlist("AHMED", Z)
This goal will return Z bound to:
.....
.....
.....
.....
.....

(21) The following program illustrates how you can use fronttoken to divide a sentence into a list of names.

```
DOMAINS
   namelist = name*
   name  = symbol
PREDICATES
   string_namelist(string, namelist)
CLAUSES
   string_namelist(S,[H|T]):-
      fronttoken(S,H,S1),!,
      string_namelist(S1,T).
      string_namelist(_,[]).
```

Try the goal:
`string_namelist("ahmed farid samir esam hamdy",X).`
X will be bound to:
.....
.....
.....

(22) Edit and run Turbo Prolog program to print the minimum and the maximun values of a given list of numbers. Also find the range where

Range=maximum value - minimum value.

.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(23) A positive integer P is prime if its only positive divisors are 1 and P, for example 1,2,3,5,7,...

Edit and run Turbo Prolog program to print the primes from 3 to N.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....