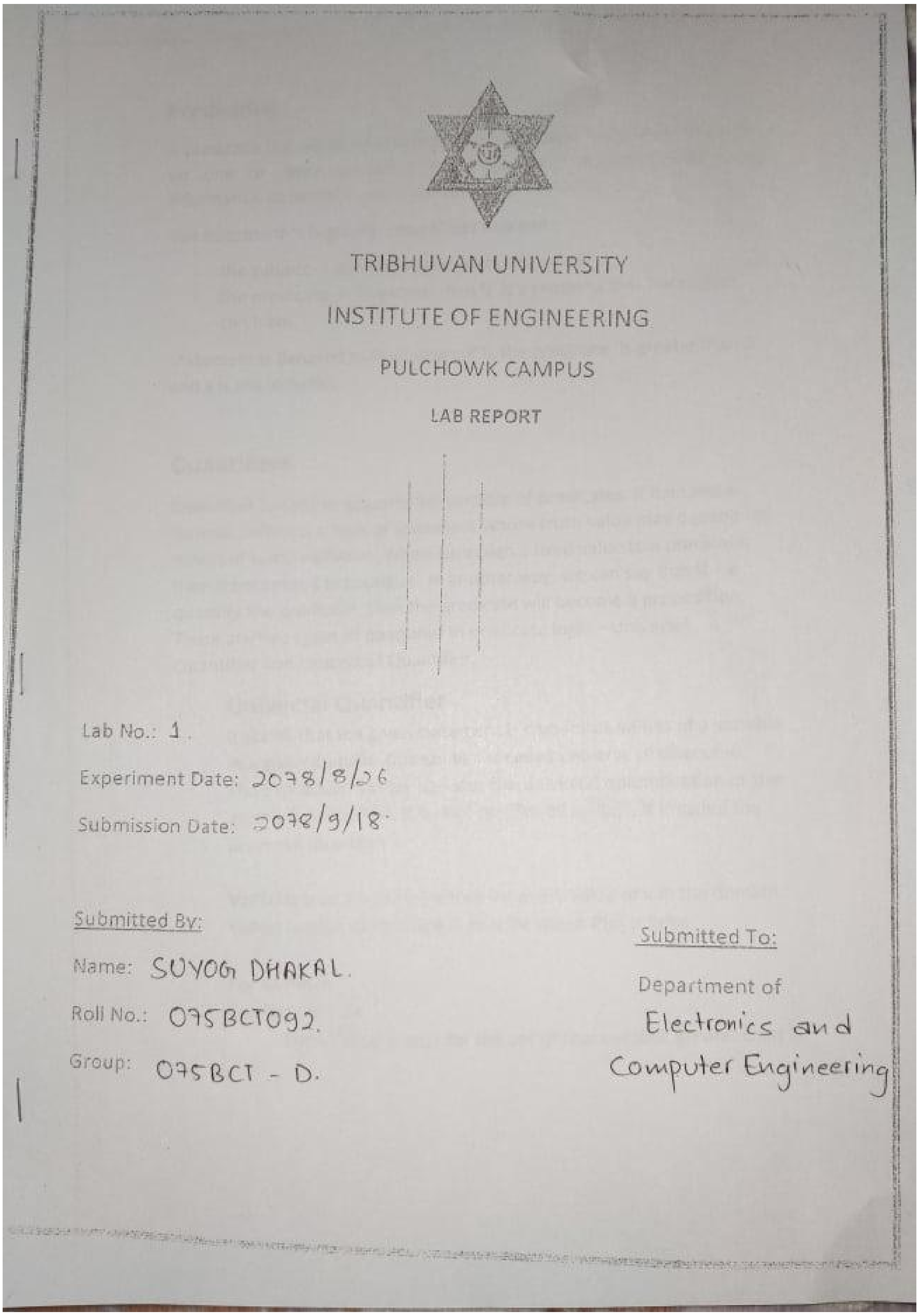


TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

LAB REPORT

Lab No.: 1.

Experiment Date: 2078/8/26

Submission Date: 2078/9/18

Submitted By:

Name: SUYOG DHAKAL.

Roll No.: 075BCT092.

Group: 075BCT - D.

Submitted To:

Department of
Electronics and
Computer Engineering

Prolog

Prolog is a very important tool in programming artificial intelligence applications and in the development of expert systems. Prolog as the name itself suggests, is the short form of LOGical PROgramming. It is a logical and declarative programming language. Prolog can be viewed as a tool to solve problems in the field of artificial intelligence or it can be very well used a general programming language.

Data types in prolog

- Atoms and numbers
- Variables
- Structures

Atoms and numbers

Atoms can be constructed in three different ways

1. strings of letters, digits, and the underscore character '_' starting with a lower case letter.

for example:

man, ram, comp_students, pc_ct_059.

2. strings of special characters

for example:

<------->

:::::::::::

Care should be taken not to use the character combination that may have some built in meaning.

3. strings of characters enclosed in quotes

for example

'Ram'

'Bird'

Numbers used in prolog are integers and real numbers.

Variables

Variables are strings of letters digits and underscore that start with an underscore or an upper-case letter. The scope of a variable is one clause only. So the same variable used in different clauses mean different thing.

For example:

X, Ram, _weight etc.

Note here that Ram is a variable unlike the earlier use ‘Ram’ where it was a constant, an atom.

An underscore ‘_’ also known as anonymous variable is used in clauses when a variable need not be inferred to more than once.

Structures

Structures are objects that have different components. The components can be atoms or yet some other structures. A functor is used to construct a structure as follows.

family(father, mother, children)

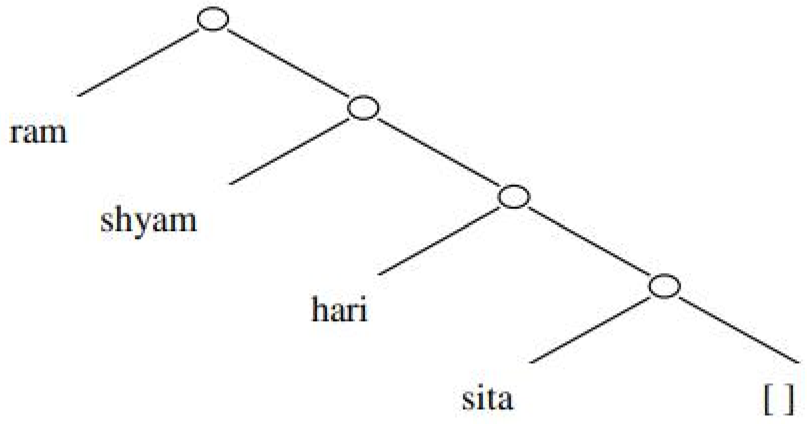
Here family is a structure that has father, mother and the children as its elements. The father and mother may be atoms while the children may be yet another structure or a list of atoms. List is a special built in structure in prolog.

A list is a built in structure in prolog. It can be thought of as a sequence of elements ordered linearly however it is internally represented as a binary tree.

For example:

[ram,shyam,hari,sita]

The representation is as follows



The list as such can be broken down into two parts, the HEAD and the TAIL. The head is the first element of the list and the tail is the remaining list. The above list can be broken down as

[H| T]

Where H= ram and T= [shyam, hari,sita]

List is one of the most useful structure in prolog.

Example of program in Prologs

```
% program that finds or checks the larger number

PREDICATES
bigger(integer,integer,integer)

CLAUSES
bigger(X,Y,Z):-
        X>Y,Z=X.
bigger(X,Y,Z):-
        X<Y,Z=Y.

GOAL
bigger(10,70,X).
```

Predicate Logic – Definition

A predicate is an expression of one or more variables defined on some specific domain. A predicate with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable.

The following are some examples of predicates −

- Let E(x, y) denote "x = y"
- Let X(a, b, c) denote "a + b + c = 0"
- Let M(x, y) denote "x is married to y"

Well Formed Formula

Well Formed Formula (wff) is a predicate holding any of the following −

1.All propositional constants and propositional variables are wffs.

2.If x is a variable and Y is a wff, $\forall xY \forall xY$ and $\exists xY \exists xY$ are also wff.

3.Truth value and false values are wffs.

4.Each atomic formula is a wff.

5.All connectives connecting wffs are wffs.

Quantifiers

The variable of predicates is quantified by quantifiers. There are two types of quantifier in predicate logic − Universal Quantifier and Existential Quantifier.

Universal Quantifier

Universal quantifier states that the statements within its scope are true for every value of the specific variable. It is denoted by the symbol $\forall \forall$.

$\forall xP(x) \forall xP(x)$ is read as for every value of x, P(x) is true.

Example − "Man is mortal" can be transformed into the propositional form $\forall xP(x) \forall xP(x)$ where P(x) is the predicate which denotes x is mortal and the universe of discourse is all men.

Existential Quantifier

Existential quantifier states that the statements within its scope are true for some values of the specific variable. It is denoted by the symbol $\exists$ $\exists$.

$\exists xP(x) \exists xP(x)$ is read as for some values of x, P(x) is true.

Example − "Some people are dishonest" can be transformed into the propositional form $\exists xP(x) \exists xP(x)$ where P(x) is the predicate which denotes x is dishonest and the universe of discourse is some people.

Nested Quantifiers

If we use a quantifier that appears within the scope of another quantifier, it is called nested quantifier.

Example

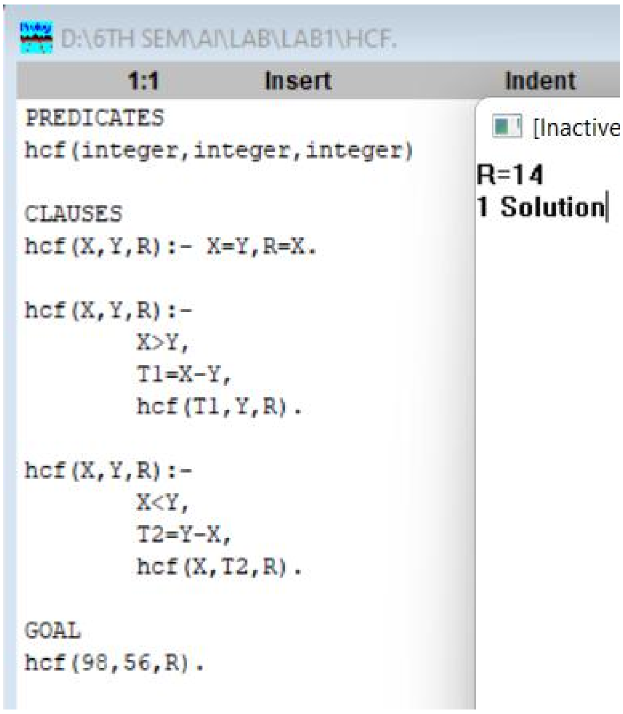
1. $\forall a \exists bP(x,y) \forall a \exists bP(x,y)$ where $P(a,b)P(a,b)$ denotes $a+b=0a+b=0$.

2. $\forall a \forall b \forall cP(a,b,c) \forall a \forall b \forall cP(a,b,c)$ where $P(a,b)P(a,b)$ denotes $a+(b+c)=(a+b)+ca+(b+c)=(a+b)+c$.

Note − $\forall a \exists bP(x,y) \neq \exists a \forall bP(x,y)$

Programs:

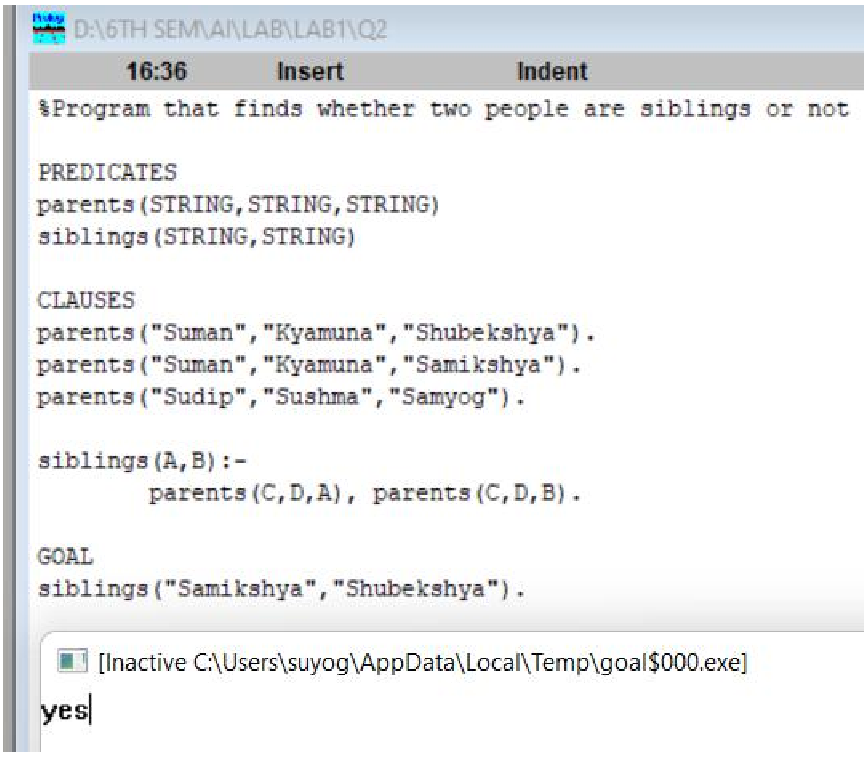
1. Write a program to find the HCF of two numbers.



```
PREDICATES
hcf(integer,integer,integer)

CLAUSES
hcf(X,Y,R):- X=Y,R=X.

hcf(X,Y,R):-
        X>Y,
        T1=X-Y,
        hcf(T1,Y,R).

hcf(X,Y,R):-
        X<Y,
        T2=Y-X,
        hcf(X,T2,R).

GOAL
hcf(98,56,R).
```

```
R=14
1 Solution
```

2.

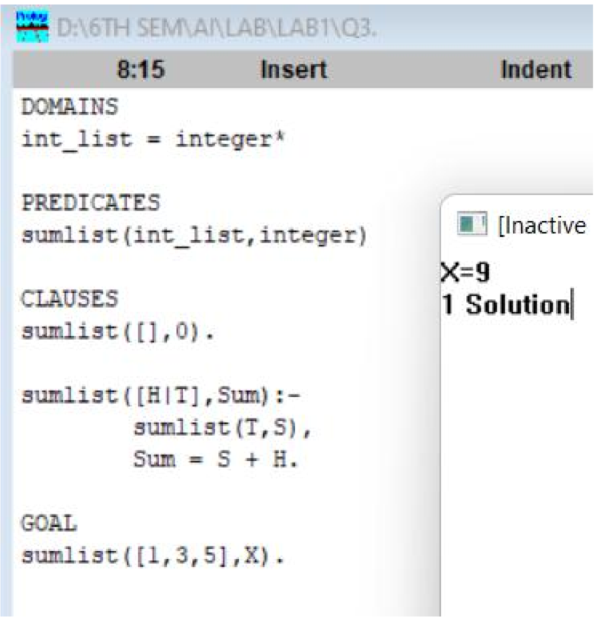


D:\6TH SEM\AI\LAB\LAB1\Q2
16:36
Insert
Indent
[Inactive C:\Users\suyog\AppData\Local\Temp\goal$000.exe]

```
%Program that finds whether two people are siblings or not

PREDICATES
parents(STRING,STRING,STRING)
siblings(STRING,STRING)

CLAUSES
parents("Suman","Kyamuna","Shubekshya").
parents("Suman","Kyamuna","Samikshya").
parents("Sudip","Sushma","Samyog").

siblings(A,B):-
        parents(C,D,A), parents(C,D,B).

GOAL
siblings("Samikshya","Shubekshya").
```

```
yes
```

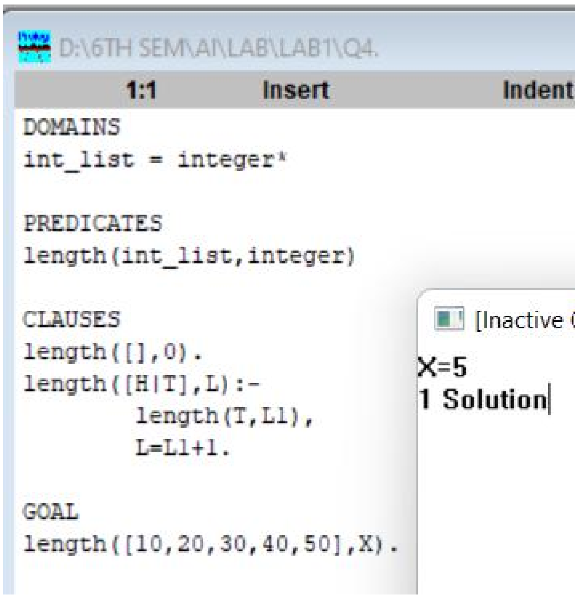
3.Write a program to add content of an integer list and display it.



```
DOMAINS
int_list = integer*

PREDICATES
sumlist(int_list,integer)

CLAUSES
sumlist([],0).

sumlist([H|T],Sum):-
        sumlist(T,S),
        Sum = S + H.

GOAL
sumlist([1,3,5],X).
```

```
X=9
1 Solution
```

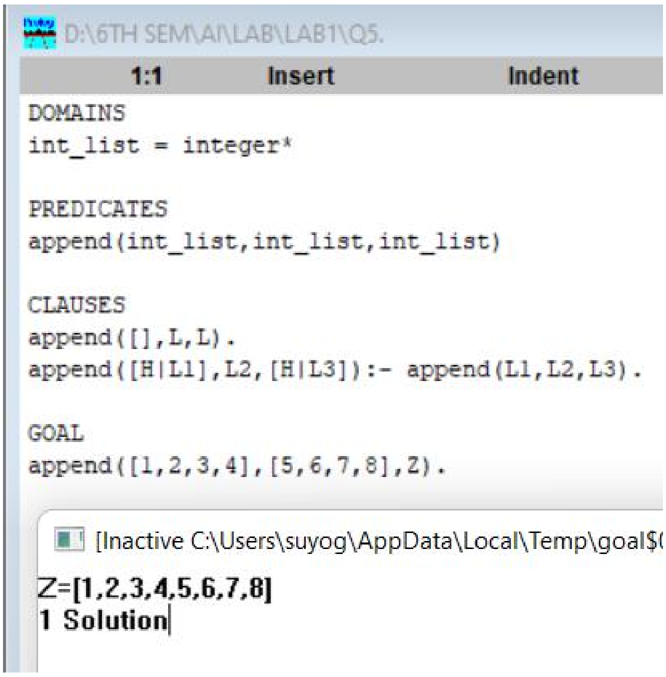
4. Write a program to find length of a list.



```
DOMAINS
int_list = integer*

PREDICATES
length(int_list,integer)

CLAUSES
length([],0).
length([H|T],L):-
        length(T,L1),
        L=L1+1.

GOAL
length([10,20,30,40,50],X).
```

```
X=5
1 Solution
```

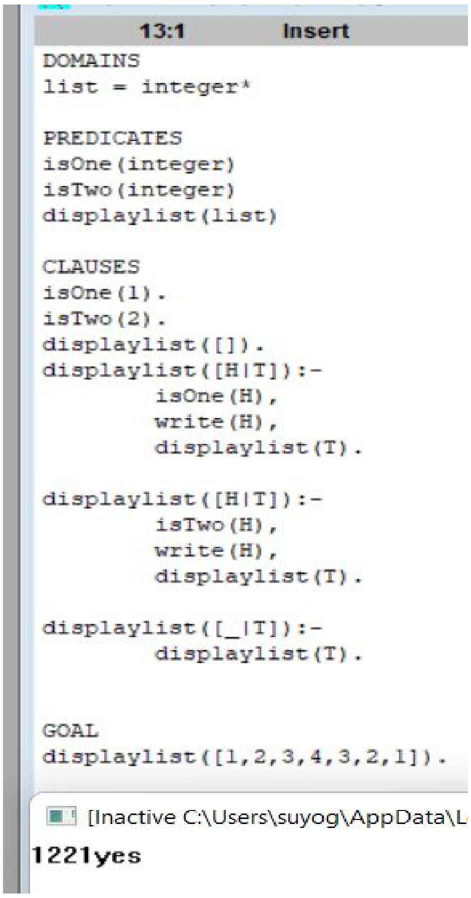
5. Write a program to append two list.


D:\6TH SEM\AI\LAB\LAB1\Q5.
1:1
Insert
Indent
[Inactive C:\Users\suyog\AppData\Local\Temp\goal$(

```
DOMAINS
int_list = integer*

PREDICATES
append(int_list,int_list,int_list)

CLAUSES
append([],L,L).
append([H|L1],L2,[H|L3]):- append(L1,L2,L3).

GOAL
append([1,2,3,4],[5,6,7,8],Z).
```

```
Z=[1,2,3,4,5,6,7,8]
1 Solution
```

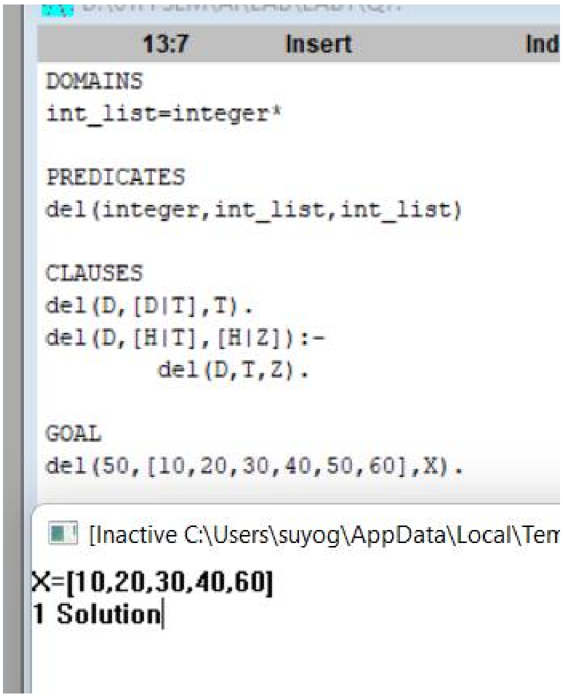
6. Write a program which takes lists of integer and display 1s and 2s only.



```
DOMAINS
list = integer*

PREDICATES
isOne(integer)
isTwo(integer)
displaylist(list)

CLAUSES
isOne(1).
isTwo(2).
displaylist([]).
displaylist([H|T]):-
        isOne(H),
        write(H),
        displaylist(T).

displaylist([H|T]):-
        isTwo(H),
        write(H),
        displaylist(T).

displaylist([_|T]):-
        displaylist(T).


GOAL
displaylist([1,2,3,4,3,2,1]).
```

```
1221yes
```

7. Write a program to delete an element from a list.



```
DOMAINS
int_list=integer*

PREDICATES
del(integer,int_list,int_list)

CLAUSES
del(D,[D|T],T).
del(D,[H|T],[H|Z]):-
        del(D,T,Z).

GOAL
del(50,[10,20,30,40,50,60],X).
```

```
X=[10,20,30,40,60]
1 Solution
```

Discussion

In this lab we learned about the basics of Prolog. We learned about domains, predicates, clauses and many more syntax. After reading the content provided on labsheet and visualizing the given examples on visual prolog, we solved the assignment.

Conclusion

Hence, we learned about prolog and wrote the program of prolog on visual prolog software.