**Task Name:** Write a hello Sandip program in Prolog.

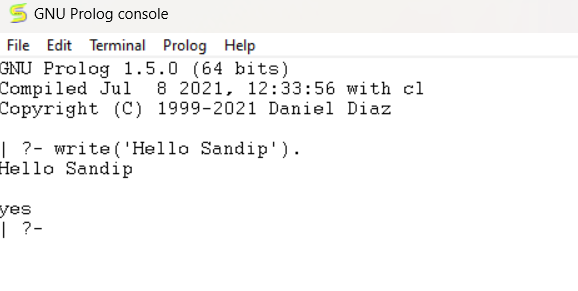
**Objective:**

The objective of this task is to write a basic Prolog program that prints "**'Hello Sandip”** to the console.

**Code:**

**write('Hello Sandip').**

**Output:**

****

**Task Name**: Represent the following facts in Prolog.

a. Tom is a cat

b. Kunal loves to eat Pasta

c. Hair is black

d. Nawaz loves to play games

e. Pratyusha is lazy.

f. Lili dances.

g. Tom is searching for food.

h. Jack loves to play cricket.

i. Bili loves to play cricket.

j. Ryan is free.

**Objective:**

The objective of this task is to familiarize you with defining and representing basic facts in Prolog.

**Code:**

cat(tom).

loves(kunal, pasta).

loves(nawaz, games).

loves(jack, cricket).

loves(bill, cricket).

hair(black).

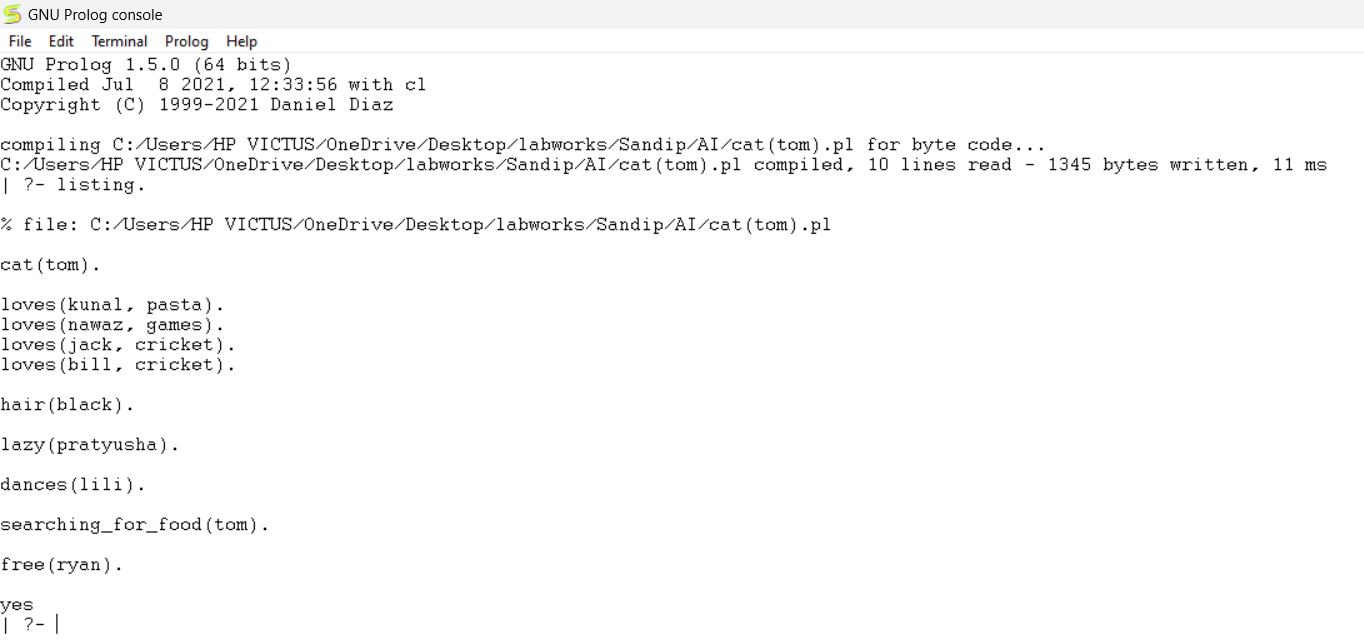
lazy(pratyusha).

dances(lili).

searching\_for\_food(tom).

free(ryan).

**Output:**



**Task Name:** Represent the following relations in Prolog along with the facts from Q. no. 2.

**Objective:**

The objective of this task is to use the previously defined facts to establish relationships between different entities and write queries to retrieve information based on these relationships.

**Code:**

% Facts

cat(tom).

hair(tom, black).

lazy(pratyusha).

dances(lili).

searching\_for\_food(tom).

free(ryan).

loves(kunal, pasta).

loves(nawaz, games).

loves(jack, cricket).

loves(bill, cricket).

school\_closed.

% Relations

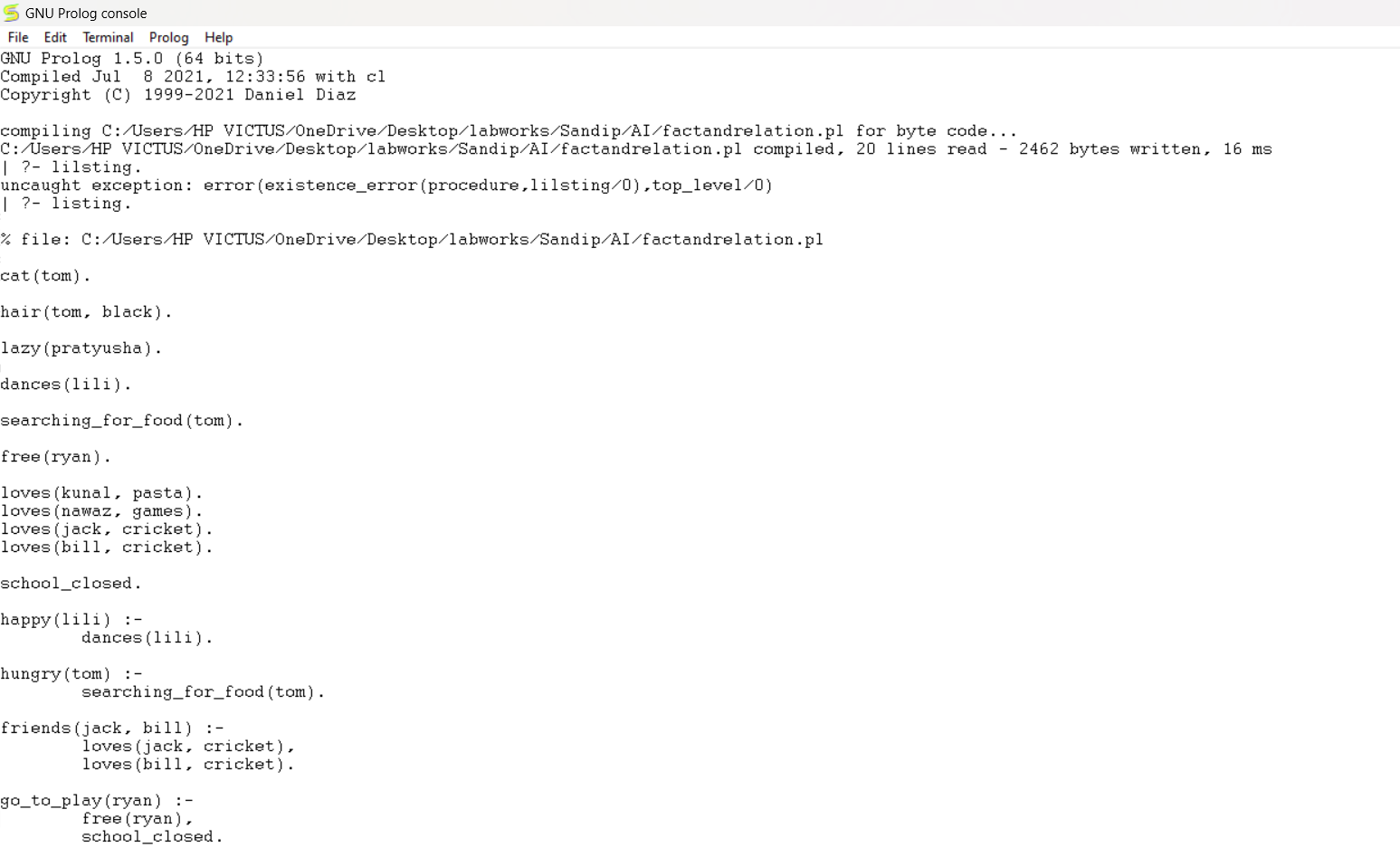
happy(lili) :- dances(lili).

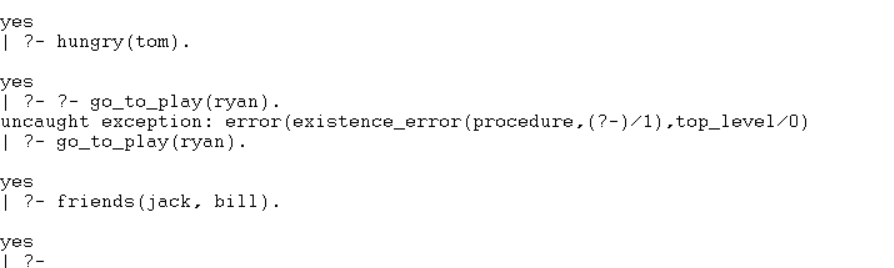
hungry(tom) :- searching\_for\_food(tom).

friends(jack, bill) :- loves(jack, cricket), loves(bill, cricket).

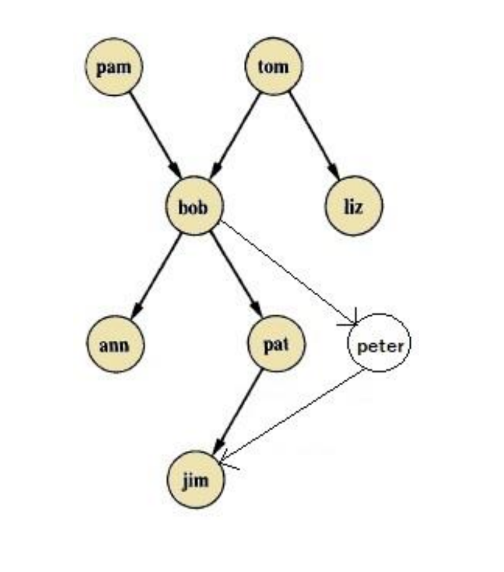
go\_to\_play(ryan) :- free(ryan), school\_closed.

**Output:**

****



**Task Name:** Implement the knowledge base of the family tree and write relationship rules.



**Objective:**

Implement a Prolog knowledge base to represent the given family tree and write rules to determine relationships such as mother, sister, father, grandfather, grandmother, grandparent, uncle, wife, and husband.

**Code:**

mother(pam, bob).

mother(pal, jim).

wife(pam, tom).

wife(pal, peter).

sister(ann, pal).

grandfather(tom, ann).

grandfather(tom, pal).

grandfather(tom, peter).

grandfather(bob, jim).

grandmother(pam, ann).

grandmother(pam, pal).

grandmother(pam, peter).

father(tom, bob).

father(tom, liz).

father(bob, ann).

father(bob, pal).

father(bob, peter).

father(peter, jim).

husband(tom, pam).

husband(peter, pal).

uncle(liz, ann).

uncle(liz, pal).

uncle(liz, peter).

female(pam).

female(ann).

female(pat).

male(tom).

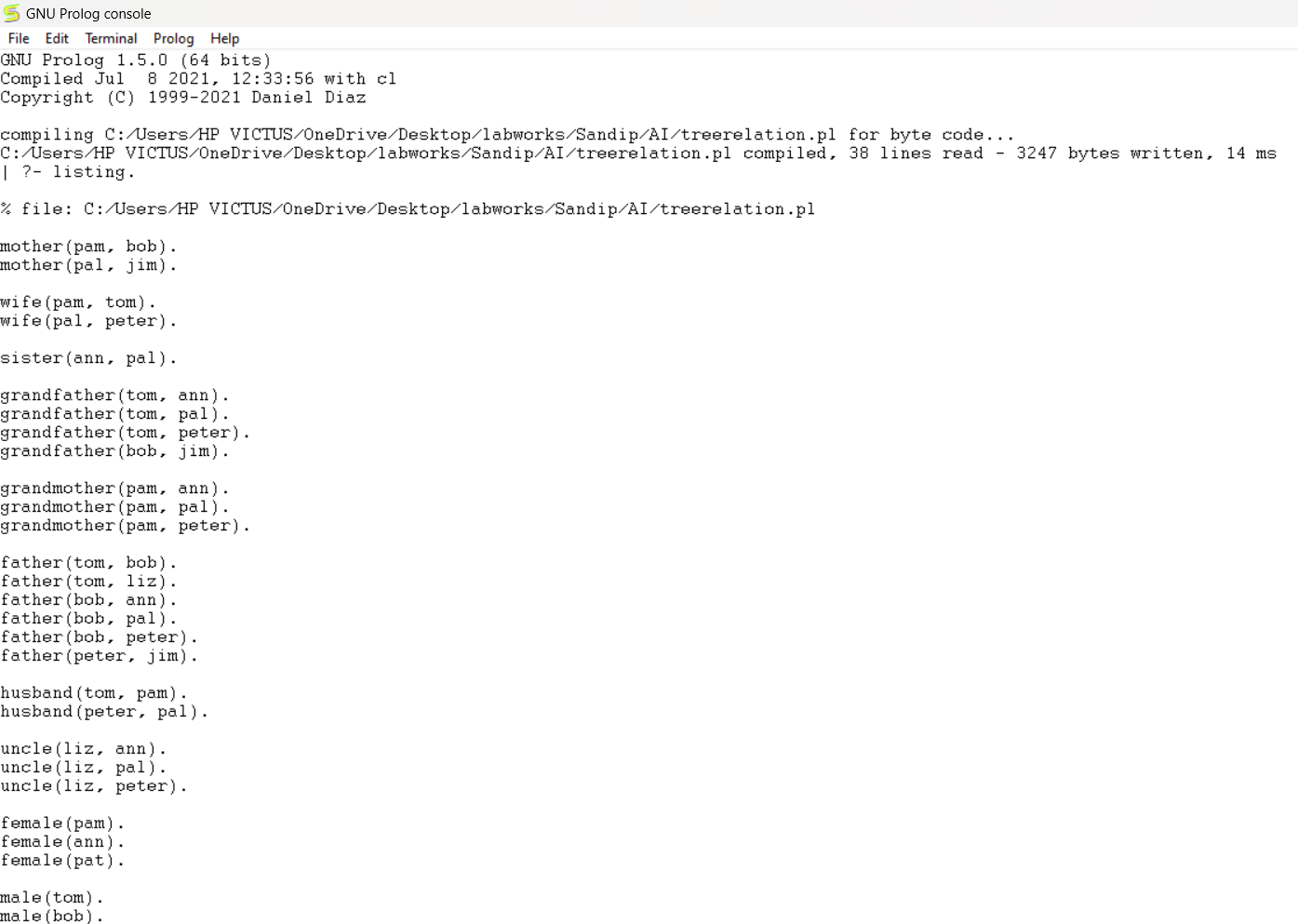
male(bob).

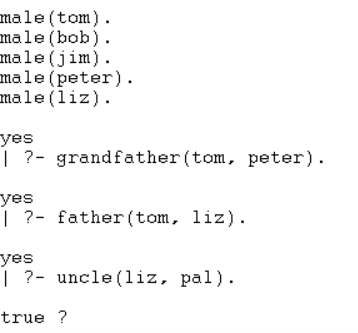
male(jim).

male(peter).

male(liz).

**Output:**

****

****

**Conclusion:**In this Prolog implementation, we've successfully represented a variety of facts and relationships using predicates to define individuals' attributes, likes, behaviors, and familial connections. From simple assertions like "Tom is a cat" to more complex relationships such as determining who is a mother, sister, grandfather, or friend based on defined rules, Prolog's logical programming paradigm proves effective for expressing and querying structured knowledge. Each predicate and rule serves to build a coherent knowledge base that can be queried to answer specific questions about the relationships and attributes defined within the system, showcasing the versatility and power of Prolog in knowledge representation and reasoning tasks.