

**GUJARAT** **TECHNOLOGICAL** **UNIVERSITY**

**Government** **Engineering** **College,** **Bhavnagar**

Subject: **Artificial Intelligence**

**B.E.** **C.E.** **Semester 7th** (**Computer** **Branch**)

**Submitted** **By:**

**Name:** **Vankani** **Arjun**

**Enrollment:** **180210107060**

**Prof.** **Ashish Nimavat** (Faculty Guide) **Prof.** **KARSHAN** **KANDORIYA**

**Prof.** **Kirit Rathod** (Lab Guide)

(Head of the Department)

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR No.** | **Algorithm**  **(Lab work)** | **Page No** |
| **1** | **Write a PROLOG program that list four addresses in a label form, each address should list a name, one-line address, city, state &ZIP code.** | **03** |
| **2** | **WAP to Create Database for Hobbies of Different Person.** | **05** |
| **3** | **Write a PROLOG program for diagnosis the childhood diseases.** | **06** |
| **4** | **Write a PROLOG program for Family Relationship.** | **09** |
| **5** | **A)  Give an opportunity to user to re-enter the password ‘n’ no. Of times, on entering wrong password. B) Give an opportunity to user to re-enter the password three (03) times, on entering wrong password.** | **13** |
| **6** | **Write a PROLOG program to implement Tower of Hanoi Problem.** | **15** |
| **7** | **Write a PROLOG program to calculate the roots of quadratic equation Consider all possibilities real, equal, imaginary.** | **16** |
| **8** | **Write a PROLOG program to solve Water-Jug Problem.** | **19** |
| **9** | **Implement Breadth first search and breadth first search algorithms in choice of your language** | **22** |
| **10** | **Implement Depth first search and breadth first search algorithms in choice of your language.** | **24** |

**Artificial Intelligence**

* **Practical-1: Write a PROLOG program that list four addresses in a label form, each address should list a name, one-line address, city, state &ZIP code.**

**Code in Prolog:**

**Domains**

**Name,Soci,Mycity,Sta,Code = String**

**Predicates**

**getaddress(Name,Soci,Mycity,Sta,Code).**

**Clauses**

**soc(arjun,gayatrinagar).**

**soc(vankani,victoria).**

**soc(arjunvankani,ringroad).**

**city(arjun,bhavnagar).**

**city(vankani,broda).**

**city(arjunvankani,ahm).**

**state(arjun,gujrat).**

**state(vankani,gujrat).**

**state(arjunvankani,gujrat).**

**zip(arjun,364001).**

**zip(vankani,314001).**

**zip(arjunvankani,382002).**

**getaddress(Name,Soci,Mycity,Sta,Code):-**

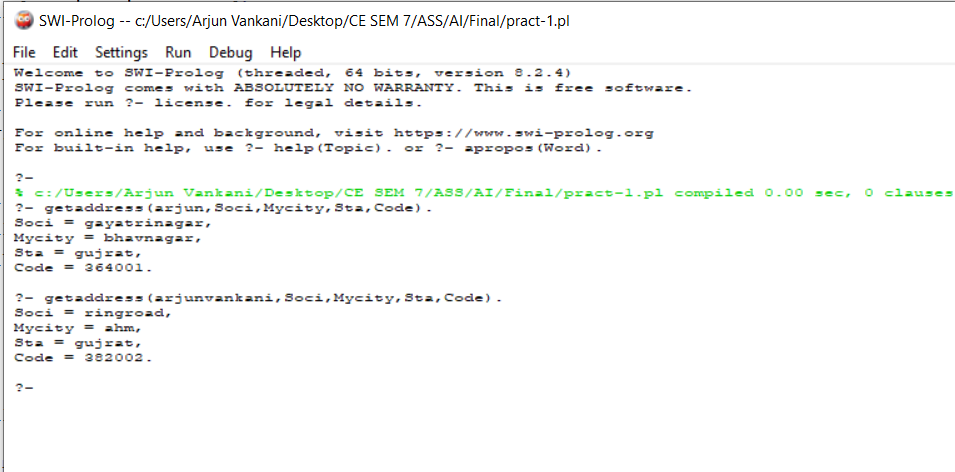
**soc(Name,Soci),**

**city(Name,Mycity),**

**state(Name,Sta),**

**zip(Name,Code).**

**Output:**



* **Practical-2: WAP to Create Database for Hobbies of Different Person.**

**Code in Prolog:**

**Domains**

**Name,hobbies = True or False**

**Predicates**

**likes(arjun,chess).**

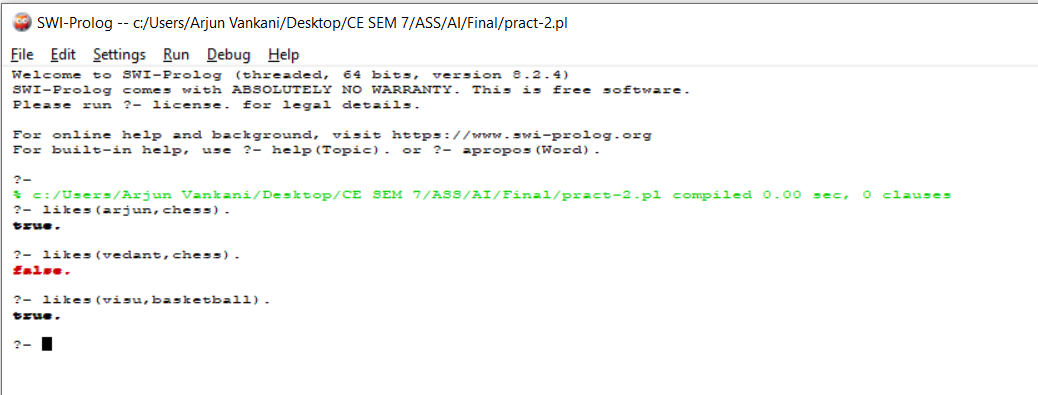
**Clauses**

**likes(arjun,chess).**

**likes(vedant,volleyball).**

**likes(visu,basketball).**

**Output:**

****

* **Practical-3: Write a PROLOG program for diagnosis the childhood diseases.**

**Code in Prolog:**

**Domains**

**Patient,Disease = String**

**Predicates**

**hypothesis(patient,Disease).**

**Clauses**

**symptom(arjun,fever).**

**symptom(arjun,headache).**

**symptom(arjun,runnynose).**

**symptom(arjun,rash).**

**hypothesis(patient,measles):-**

**symptom(Patient,fever),**

**symptom(Patient,cough),**

**symptom(Patient,conjunctive),**

**symptom(Patient,runnynose),**

**symptom(Patient,rash).**

**hypothesis(Patient,germanmeasles):-**

**symptom(Patient,fever),**

**symptom(Patient,headache),**

**symptom(Patient,runnynose),**

**symptom(Patient,rash).**

**hypothesis(Patient,flu):-**

**symptom(Patient,fever),**

**symptom(Patient,headache),**

**symptom(Patient,bodyache),**

**symptom(Patient,chills),**

**symptom(Patient,sorethrought),**

**symptom(Patient,cough),**

**symptom(Patient,conjunctive),**

**symptom(Patient,conjunctive),**

**symptom(Patient,runnynose).**

**hypothesis(Patient,commoncold):-**

**symptom(Patient,headache),**

**symptom(Patient,runnynose),**

**symptom(Patient,snuzing),**

**symptom(Patient,chills),**

**symptom(Patient,sorethrought).**

**hypothesis(Patient,mumps):-**

**symptom(Patient,fever),**

**symptom(Patient,swallenglands).**

**hypothesis(Patient,chikenpox):-**

**symptom(Patient,fever),**

**symptom(Patient,rash),**

**symptom(Patient,bodyache).**

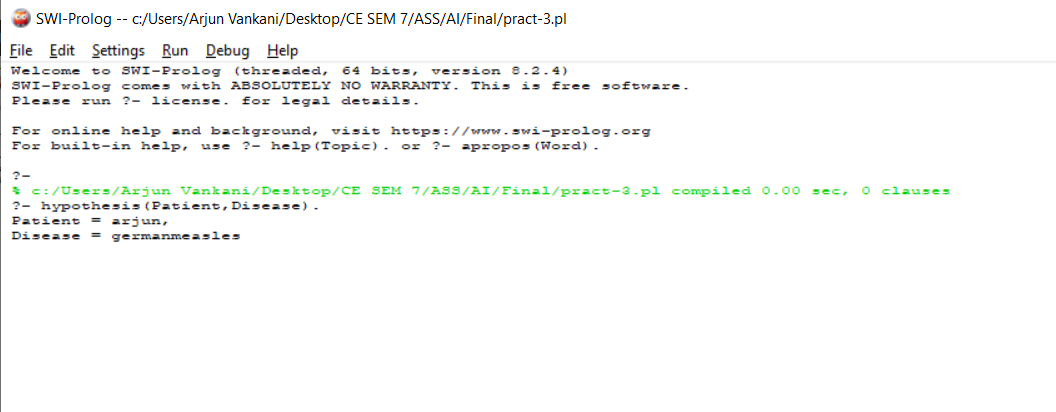
**hypothesis(Patient,whooping-cough):-**

**symptom(Patient,runnynose),**

**symptom(Patient,snuzing),**

**symptom(Patient,cough).**

**Output:**

****

*   **Practical-4: Write a PROLOG program for Family Relationship.**

Ø

**Code in Prolog:**

**Domains**

**Person = symbol**

**Predicates**

male(person)

female(person)

parent(person,person)

father(person,person)

mother(person,person)

sister(person,person)

brother(person,person)

son(person,person)

daughter(person,person)

aunt(person,person)

uncle(person,person)

child(person,person)

wife\_of(person,person)

husband\_of(person,person)

grand\_father(person,person)

grand\_mother(person,person)

cousin(person,person)

nephew(person,person)

**Clauses**

**male(arjun).**

**male(bakulbhai).**

**male(natubhai).**

**female(bharvi).**

**female(arunaben).**

**female(kundanben).**

**child(arjun,bakulbhai).**

**child(arjun,arunaben).**

**child(bharvi,bakulbhai).**

**child(bharvi,arunaben).**

**child(bakulbhai,natubhai).**

**child(bakulbhai,kundanben).**

**brother(X,Y):-**

**male(X),**

**child(X,Z),**

**child(Y,Z),**

**X\=Y.**

**sister(X,Y):-**

**female(X),**

**child(X,Z),**

**child(Y,Z),**

**X\=Y.**

**father(X,Y):-**

**male(X),**

**child(Y,X).**

**mother(X,Y):-**

**female(X),**

**child(Y,X).**

**grandfather(X,Y):-**

**male(X),**

**child(Y,Z),**

**child(Z,X).**

**grandmother(X,Y):-**

**female(X),**

**child(Y,Z),**

**child(Z,X).**

**ancestor(X,Y):-**

**male(X),**

**child(Y,Z),**

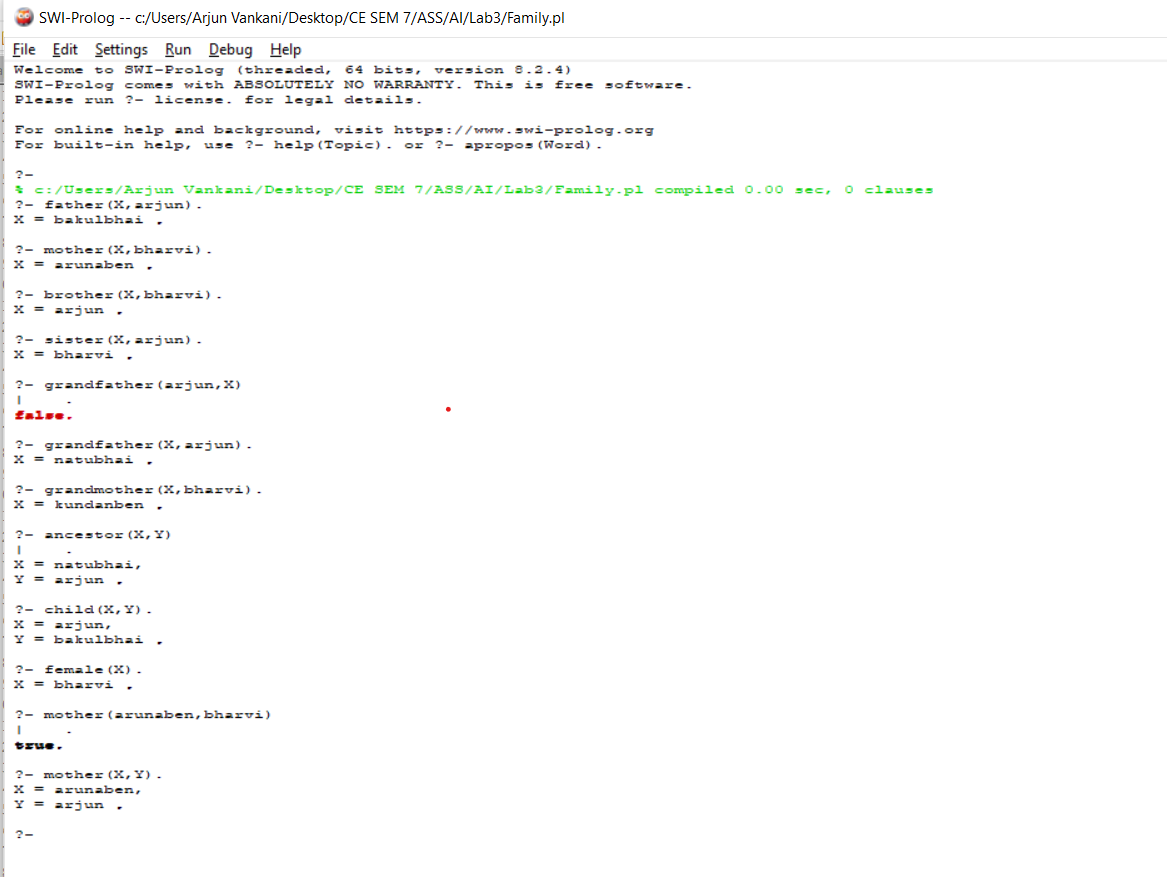
**child(Z,X);**

**female(X),**

**child(Y,Z),**

**child(Z,X).**

**Output:**



* **Practical-5:**  **A) Give an opportunity to user to re-enter the password ‘n’ no. Of times, on entering wrong password.  
  B) Give an opportunity to user to re-enter the password three (03) times, on entering wrong password.**

**Code in Prolog:**

**Domains**

**Name, password = symbol**

**Predicates**

**getinput,**

**logon,**

**user(name,password)**

**Clauses**

**logon :- getinput,**

**write('You are logged in.'),nl.**

**logon :- repeat,**

**write('Sorry, you are not permitted.'),nl,**

**write('Try again.'),nl,**

**getinput,**

**write('You are now logged in.').**

**getinput :- write('Login Windows'),nl,**

**write('Enter your username : '),**

**read(Name),nl,**

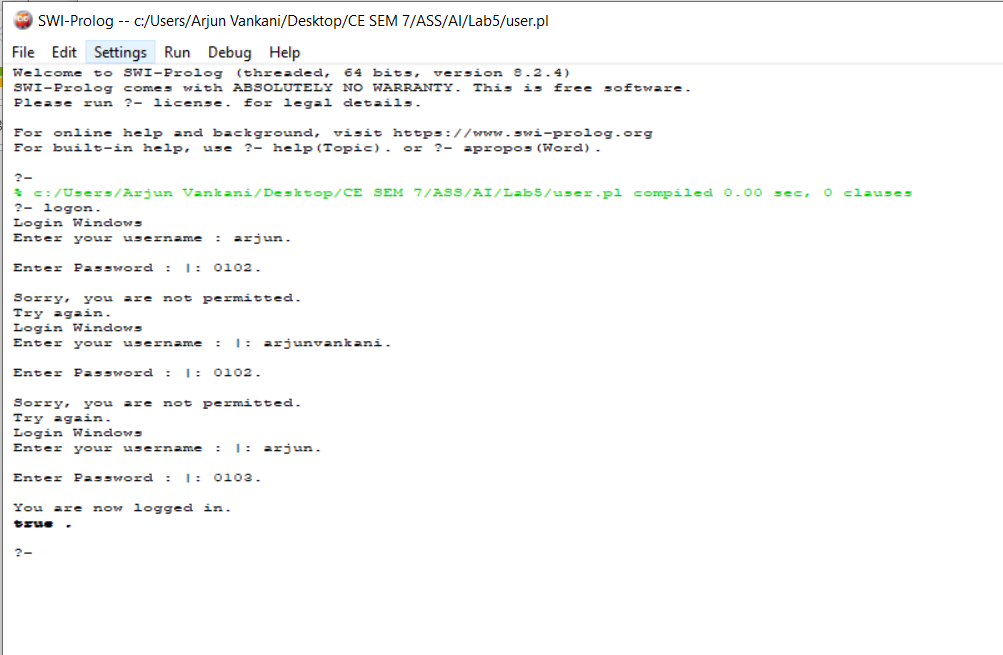
**write('Enter Password : '),**

**read(Password),nl,**

**user(Name, Password).**

**user(arjun,0103).**

**Output:**



*   **Practical-6: Write a PROLOG program to implement Tower of Hanoi Problem.**

Ø

**Code in Prolog:**

**Domains**

**POLE = symbol**

**Predicates**

**Move (INTEGER, POLE, POLE, POLE)**

**Clauses**

**move(1,X,Y,\_):-write('Move disk from '),write(X),write(' to'),write(Y),nl.**

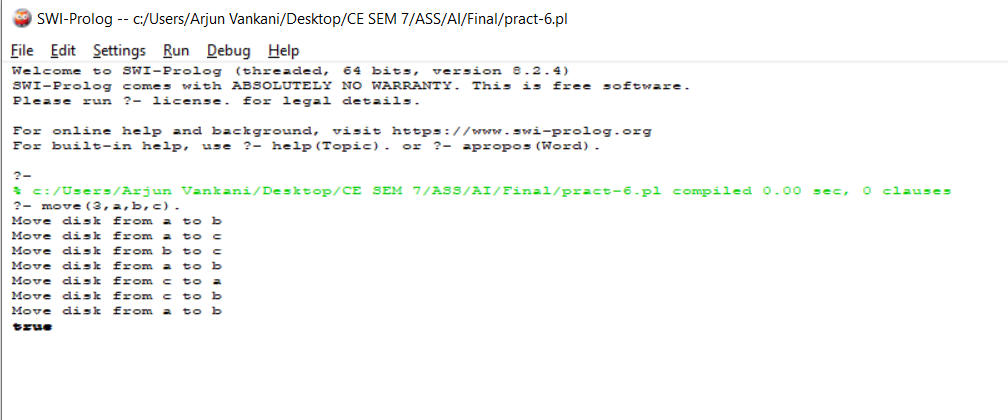
**move(N,X,Y,Z):-N>1,M is N-1,**

**move(M,X,Z,Y),**

**move(1,X,Y,\_),**

**move(M,Z,Y,X).**

**Output:**



* **Practical-7: Write a PROLOG program to calculate the roots of quadratic equation Consider all possibilities real, equal, imaginary**

Ø

**Code in Prolog:**

**Domains**

**Predicates**

root(real,real,real,real)

run

**root(real,real,real,real).**

**Run.**

**Clauses**

**run:-**

**write("Enter the value of A :" ),**

**read(A),**

**write("Enter the value of B :" ),**

**read(B),**

**write("Enter the value of C :" ),**

**read(C),**

**D = (B\*B)-(4\*A\*C),**

**root(A,B,C,D).**

**root(A,B,C,D):-**

**A=0.0,**

**write("Only one root exists."),**

**ANS = (-C/B),**

**write(ANS);**

**D>=0,**

**ANS = (-B - sqrt(D)) / (2\*A),**

**ANS1 = (-B + sqrt(D)) / (2\*A),**

**write("First root is : "),**

**write(ANS),nl,**

**write("Second root is : "),**

**write(ANS1);**

**REAL= (-B) / (2\*A),**

**IMG = sqrt(-D) / (2\*A),**

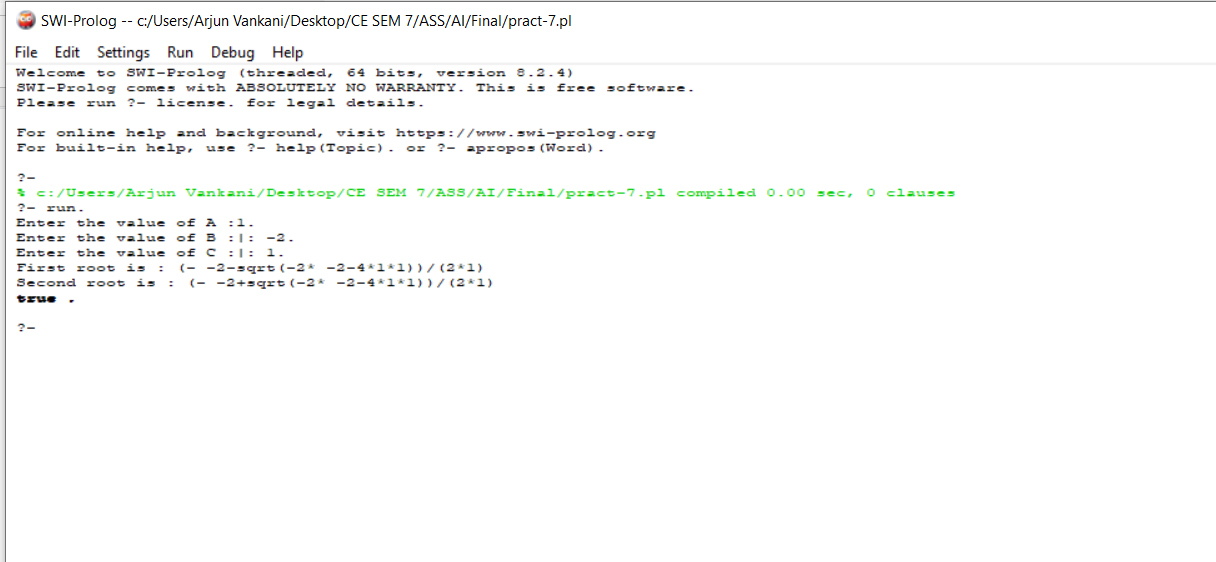
**write("Real root is : "),**

**write(REAL),nl,**

**write("Imaginary root is : "),**

**write(IMG).**

**Output:**



***   Practical-8: Write a PROLOG program to solve Water-Jug Problem.**

Ø

**Code in Prolog:**

**Domains**

**Predicates**

**jug(INTRGER, INTRGER)**

**Clauses**

**jug(2, \_).**

**jug(0,2):-**

**write('(0,2)'),nl,**

**write('(2,0)'),nl.**

**jug(4,0) :-**

**write('(4,0)'),nl,**

**jug(0,0).**

**jug(4,3) :-**

**write('(4,3)'),nl,**

**jug(0,0).**

**jug(3,0) :-**

**write('(3,0)'),nl,**

**jug(3,3).**

**jug(X,0) :-**

**write('('),write(X),write(',0'),nl,**

**jug(0,3).**

**jug(0,3) :-**

**write('(0,3)'),nl,**

**jug(3,0).**

**jug(0,X) :-**

**write('(0,'),write(X),write(')'),nl,**

**jug(0,0).**

**jug(3,3) :-**

**write('(3,3)'),nl,**

**jug(4,2).**

**jug(4,2) :-**

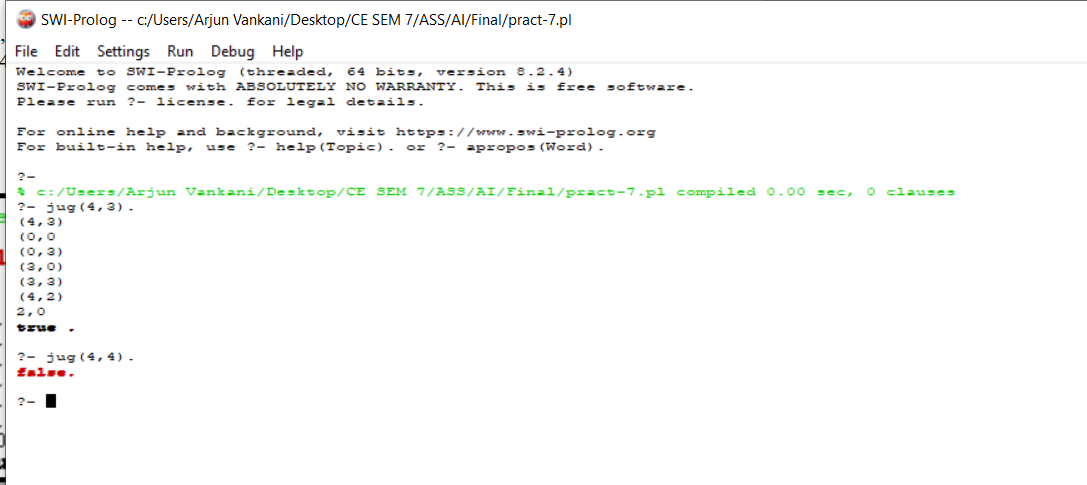
**write('(4,2)'),nl, write('2,0'), nl,**

**jug(2,0).**

**jug(X, Y) :-**

**X>4,fail,Y>3,fail.**

**Output:**



*   **Practical-9:  Implement Breadth first search algorithms in choice of your language.** 

Ø

**Code in Prolog:**

**Domains**

**Predicates**

**Clauses**

**s(a, b).**

**s(a, c).**

**s(b, g).**

**s(b, f).**

**s(c, r).**

**s(c, e).**

**goal(f).**

**solve( Start, Solution) :-**

**breadthfirst( [ [Start] ], Solution).**

**breadthfirst( [ [Node | Path] |\_], [Node | Path] ) :-**

**goal( Node).**

**breadthfirst( [ [N | Path] | Paths], Solution) :-**

**bagof([M,N|Path],**

**( s( N, M), \+ member( M, [N | Path] ) ), NewPaths),**

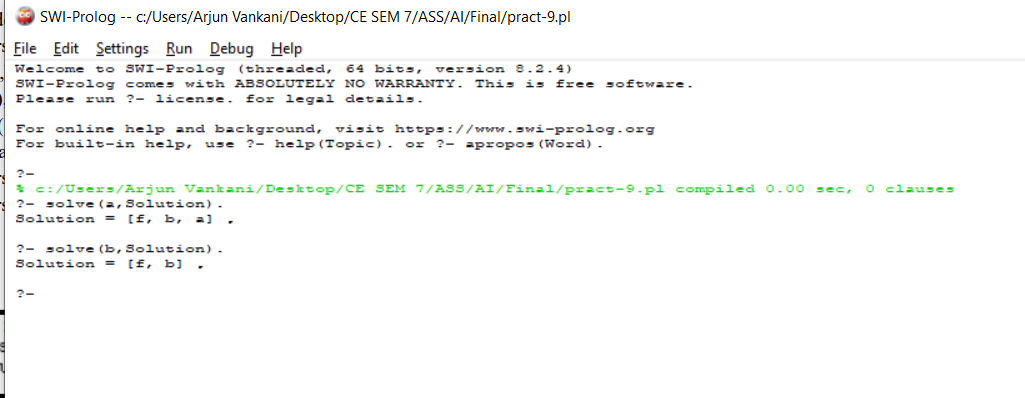
**%conc( Paths, NewPaths, Pathsl), !,**

**append(Paths, NewPaths, Pathsl), !,**

**breadthfirst( Pathsl, Solution);**

**breadthfirst( Paths, Solution).**

**Output:**



* **Practical-10:  Implement Depth first search and breadth first search algorithms in choice of your language.**

Ø

**Code in Prolog:**

**Domains**

**Predicates**

**Clauses**

**connected(1,7,1).**

**connected(1,8,1).**

**connected(1,3,1).**

**connected(7,4,1).**

**connected(7,20,1).**

**connected(7,17,1).**

**connected(8,6,1).**

**connected(3,9,1).**

**connected(3,12,1).**

**connected(9,19,1).**

**connected(4,42,1).**

**connected(20,28,1).**

**connected(17,10,1).**

**connected2(X,Y,D) :- connected(X,Y,D).**

**connected2(X,Y,D) :- connected(Y,X,D).**

**next\_node(Current, Next, Path) :-**

**connected2(Current, Next, \_),**

**not(member(Next, Path)).**

**depth\_first(Goal, Goal, \_, [Goal]).**

**depth\_first(Start, Goal, Visited, [Start|Path]) :-**

**next\_node(Start, Next\_node, Visited),**

**write(Visited), nl,**

**depth\_first(Next\_node, Goal, [Next\_node|Visited], Path).**

**Output:**

