Artificial Intelligence

**Practical file**

**Output Screen**

Name : Abhishek Tiwari

University Roll no : 19020570003

College Roll no : 2019/1403

Course : B.Sc (H) Computer Science

Semester : 6th

**Practical List**

[Practical 1 2](#_bookmark0)

[Practical 2 3](#_bookmark1)

[Practical 3 4](#_bookmark2)

[Practical 4 5](#_bookmark3)

[Practical 5 6](#_bookmark4)

[Practical 6 7](#_bookmark5)

[Practical 7 8](#_bookmark6)

[Practical 8 9](#_bookmark7)

[Practical 9 10](#_bookmark8)

[Practical 10 11](#_bookmark9)

[Practical 11 12](#_bookmark10)

[Practical 12 13](#_bookmark11)

[Practical 13 14](#_bookmark12)

[Practical 14 15](#_bookmark13)

[Practical 15 16](#_bookmark14)

[Practical 16 17](#_bookmark15)

[Practical 17 18](#_bookmark16)

[Practical 18 19](#_bookmark17)

1. Write a prolog program to calculate the sum of two numbers.

# Sum(A,B,S):- S is A+B.

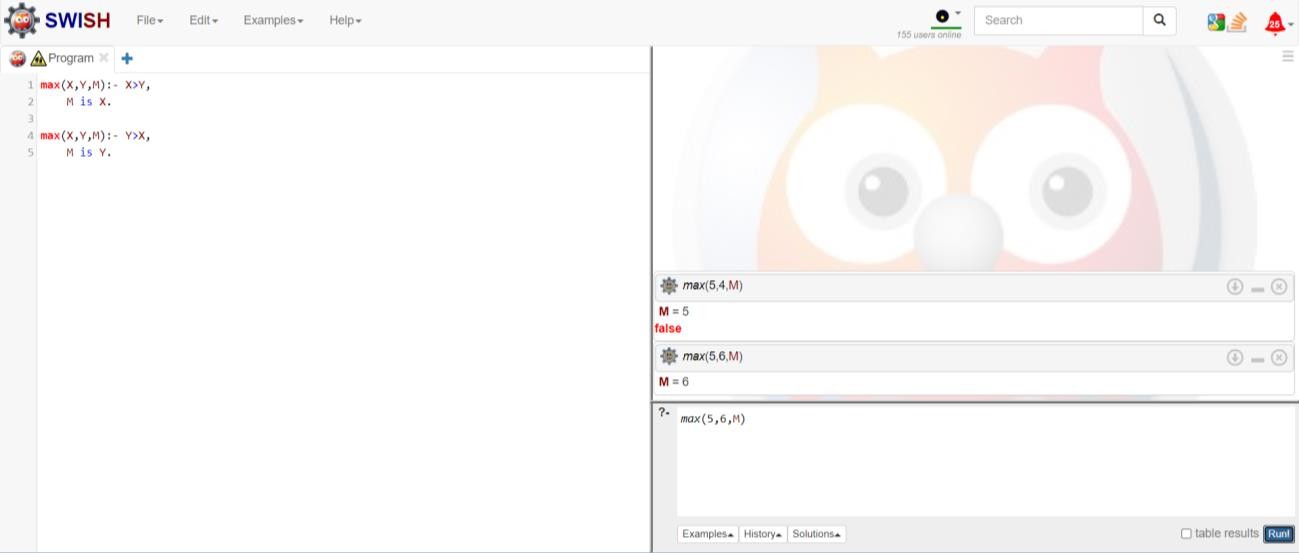
1. Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.

# max(X,Y,M):- X>Y,

**M is X.**

**max(X,Y,M):- Y>X,**

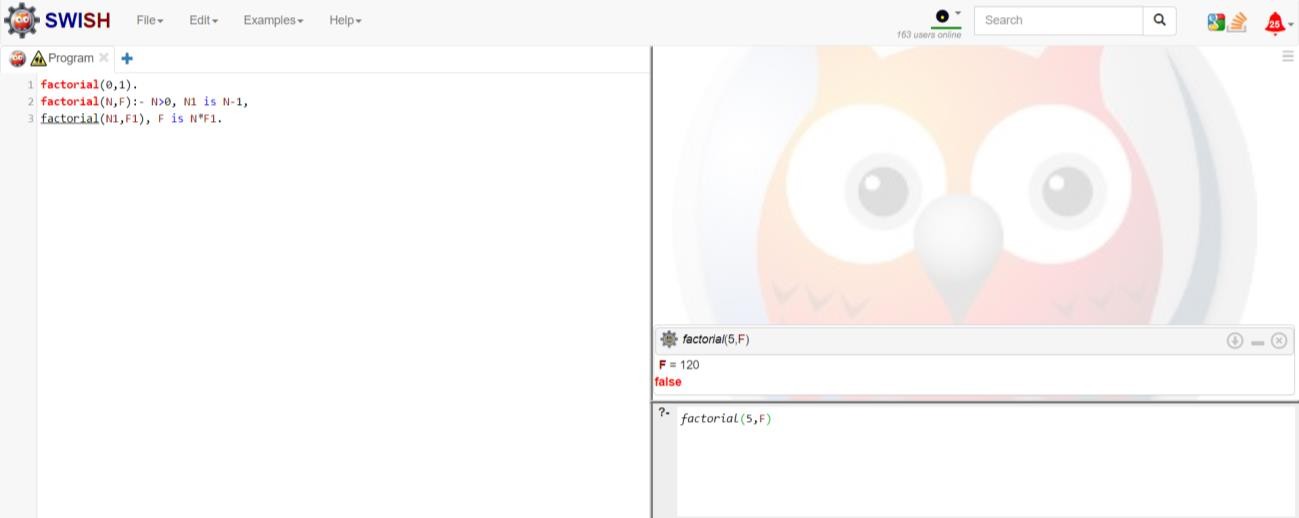
**M is Y.**



1. Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.

# factorial(0,1).

**factorial(N,F):- N>0, N1 is N-1, factorial(N1,F1), F is N\*F1.**



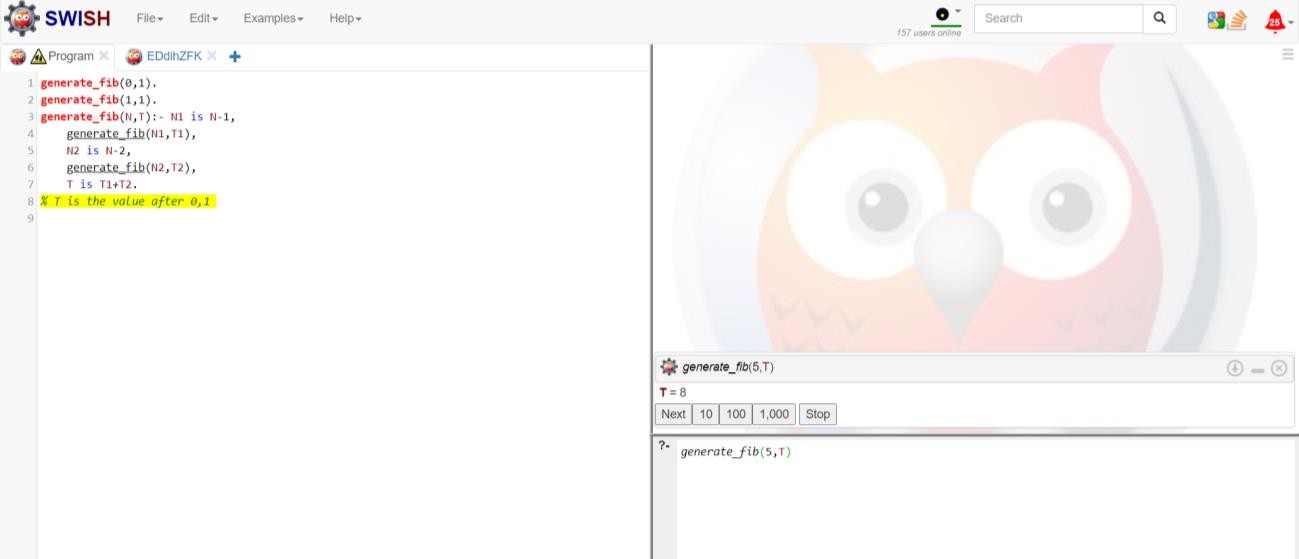
1. Write a program in PROLOG to implement generate\_fib(N,T) where T represents the Nth term of the fibonacci series.

# generate\_fib(0,1). generate\_fib(1,1). generate\_fib(N,T):- N1 is N-1,

**generate\_fib(N1,T1), N2 is N-2,**

**generate\_fib(N2,T2), T is T1+T2.**

**% T is the value after 0,1**



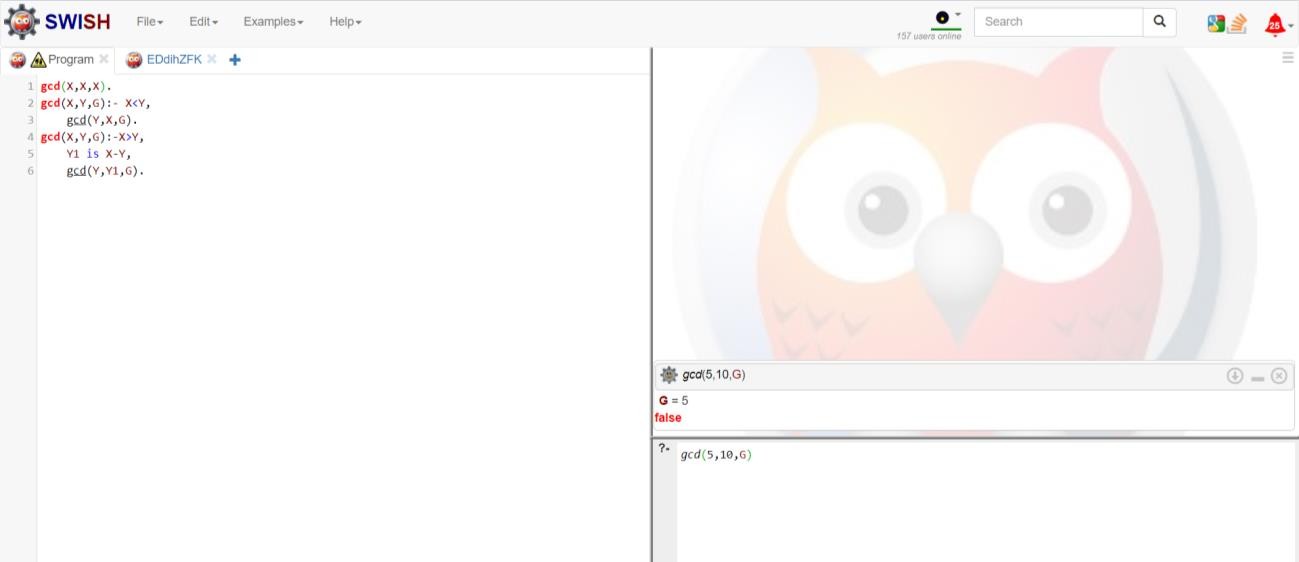
1. Write a Prolog program to implement GCD of two numbers.

# gcd(X,X,X).

**gcd(X,Y,G):- X<Y,**

**gcd(Y,X,G).**

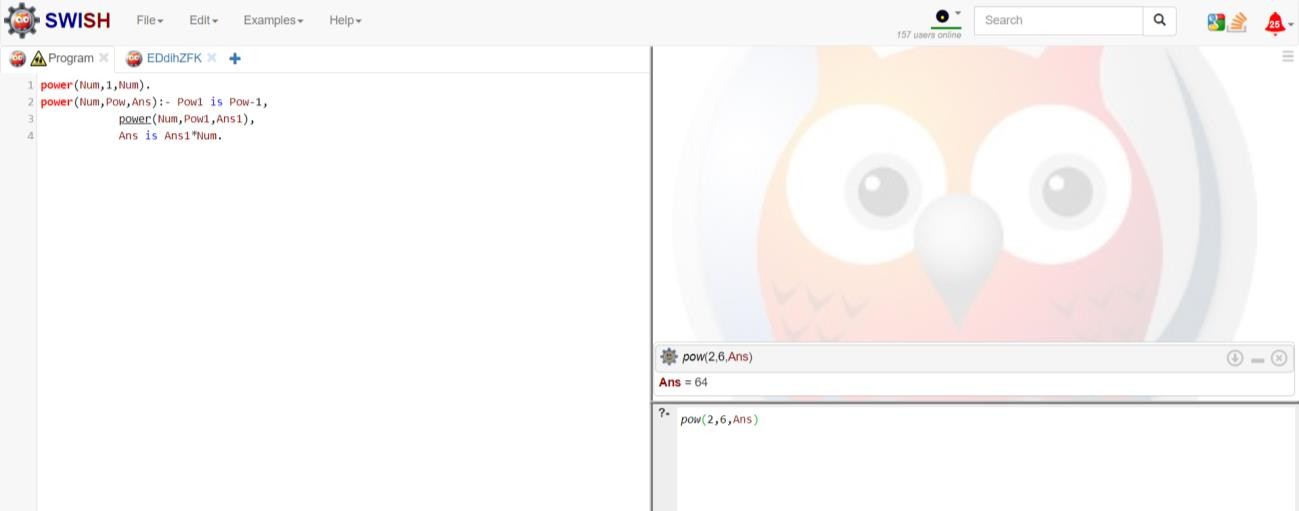
**gcd(X,Y,G):-X>Y, Y1 is X-Y, gcd(Y,Y1,G).**



1. Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.

# power(Num,1,Num). power(Num,Pow,Ans):- Pow1 is Pow-1,

**power(Num,Pow1,Ans1), Ans is Ans1\*Num.**

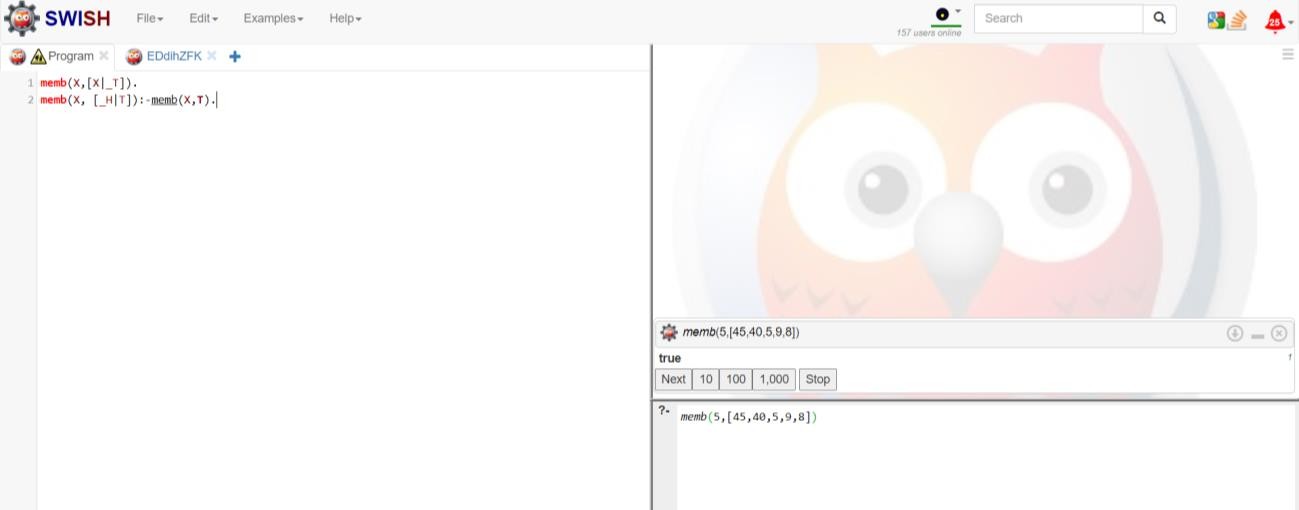


1. Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.

# multi(N1,N2,R):- R is N1\*N2.

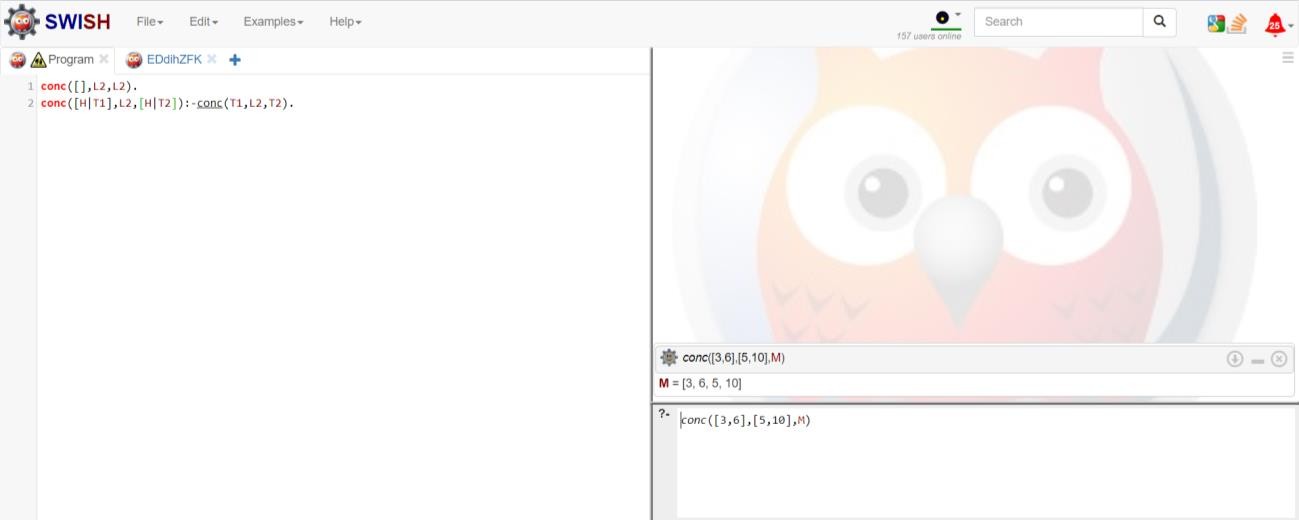
1. Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.

# memb(X,[X|\_T]).

**memb(X, [\_H|T]):-memb(X,T).**

1. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.

# conc([],L2,L2).

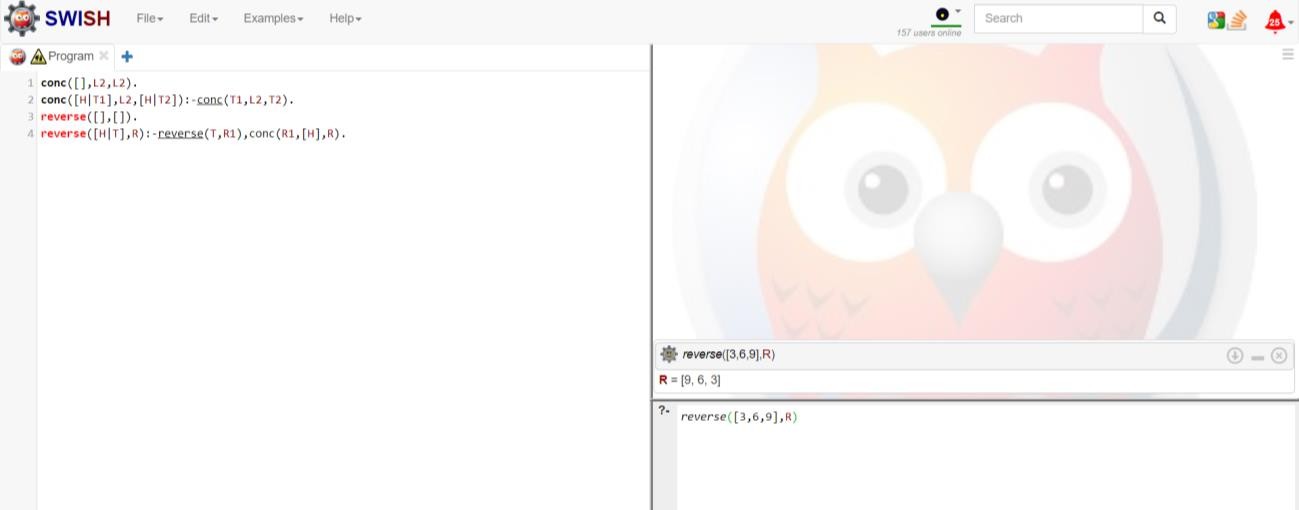
**conc([H|T1],L2,[H|T2]):-conc(T1,L2,T2).**

1. Write a Prolog program to implement reverse (L, R) where List L is original and List R is reversed list.

# conc([],L2,L2).

**conc([H|T1],L2,[H|T2]):-conc(T1,L2,T2).**

**reverse([],[]).**

**reverse([H|T],R):-reverse(T,R1),conc(R1,[H],R).**

1. Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.

# conc([],L2,L2).

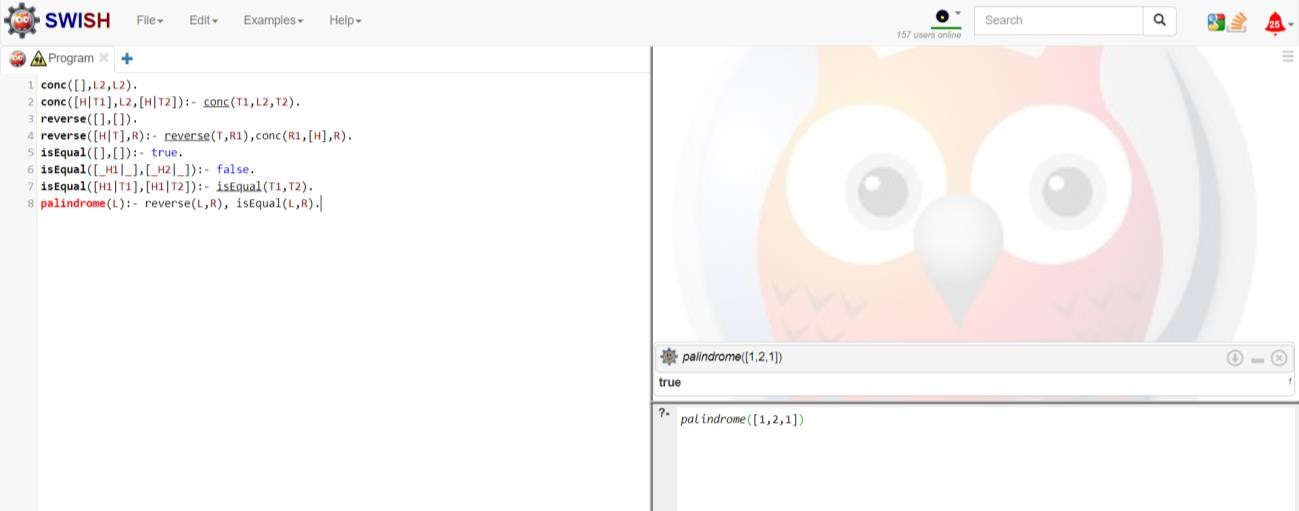
**conc([H|T1],L2,[H|T2]):- conc(T1,L2,T2).**

**reverse([],[]).**

**reverse([H|T],R):- reverse(T,R1),conc(R1,[H],R). isEqual([],[]):- true.**

**isEqual([\_H1|\_],[\_H2|\_]):- false.**

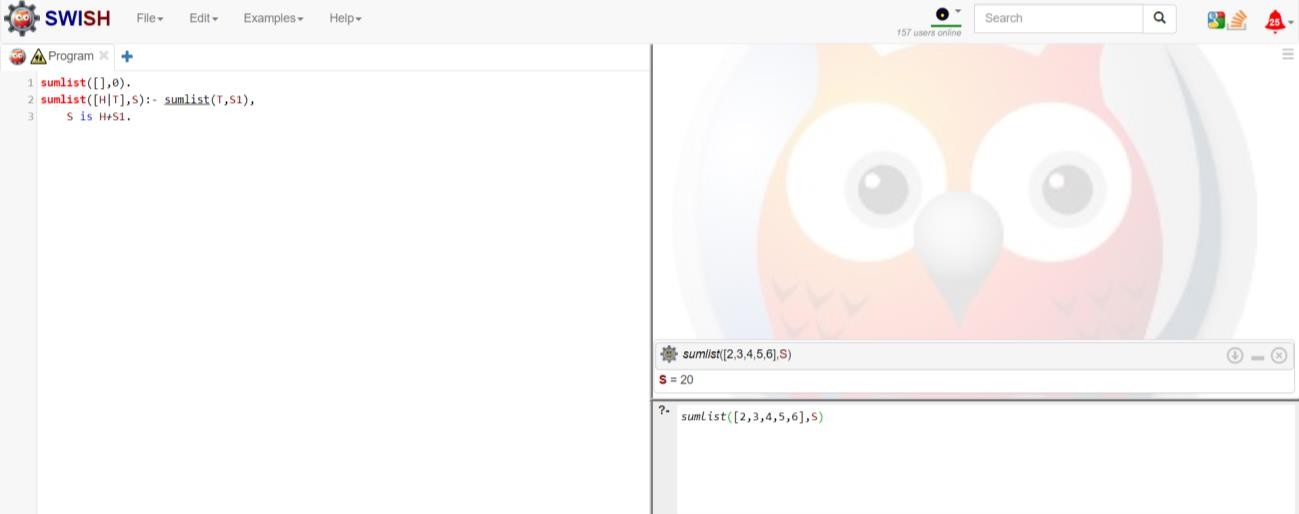
**isEqual([H1|T1],[H1|T2]):- isEqual(T1,T2). palindrome(L):- reverse(L,R), isEqual(L,R).**



1. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.

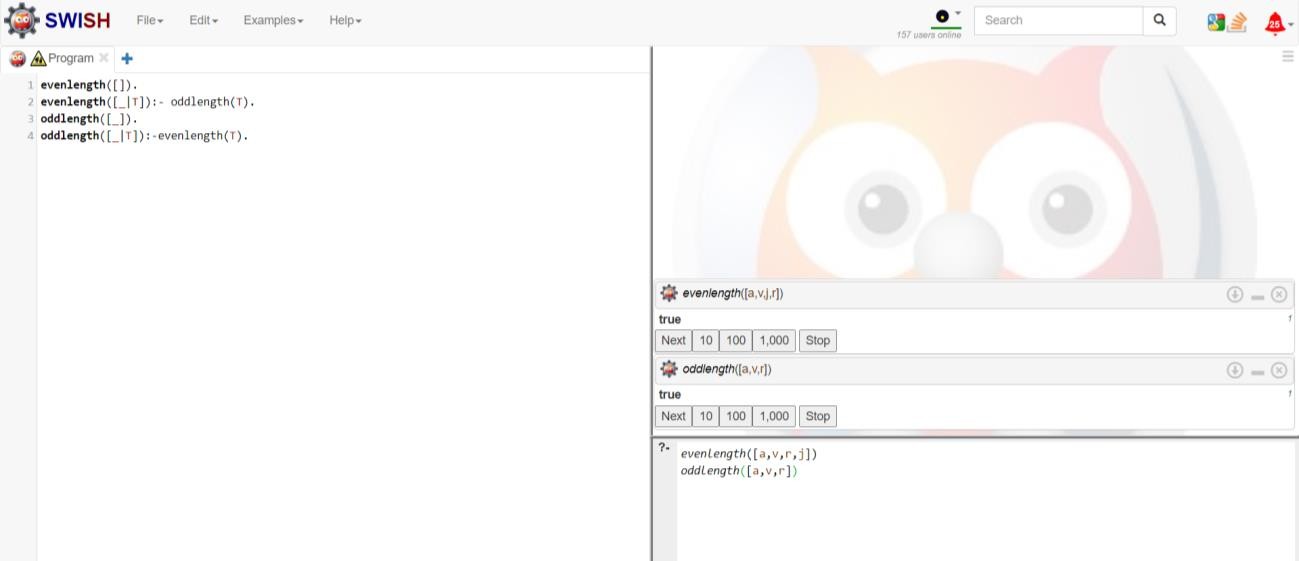
# sumlist([],0).

**sumlist([H|T],S):- sumlist(T,S1), S is H+S1.**



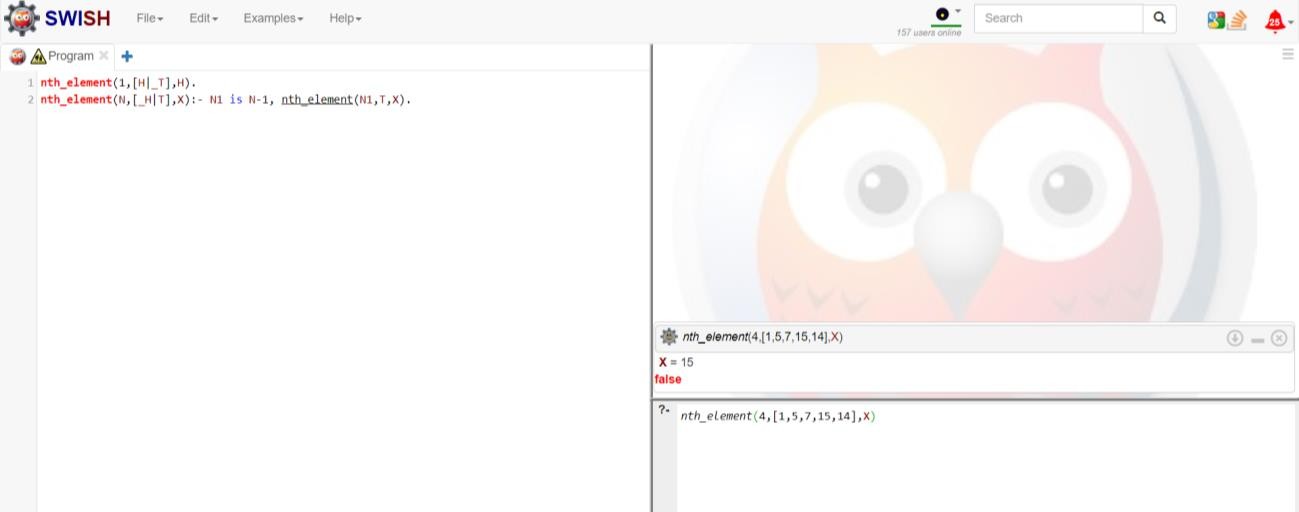
1. Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.

# evenlength([]). evenlength([\_|T]):- oddlength(T). oddlength([\_]).

**oddlength([\_|T]):-evenlength(T).**

1. Write a Prolog program to implement nth\_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.

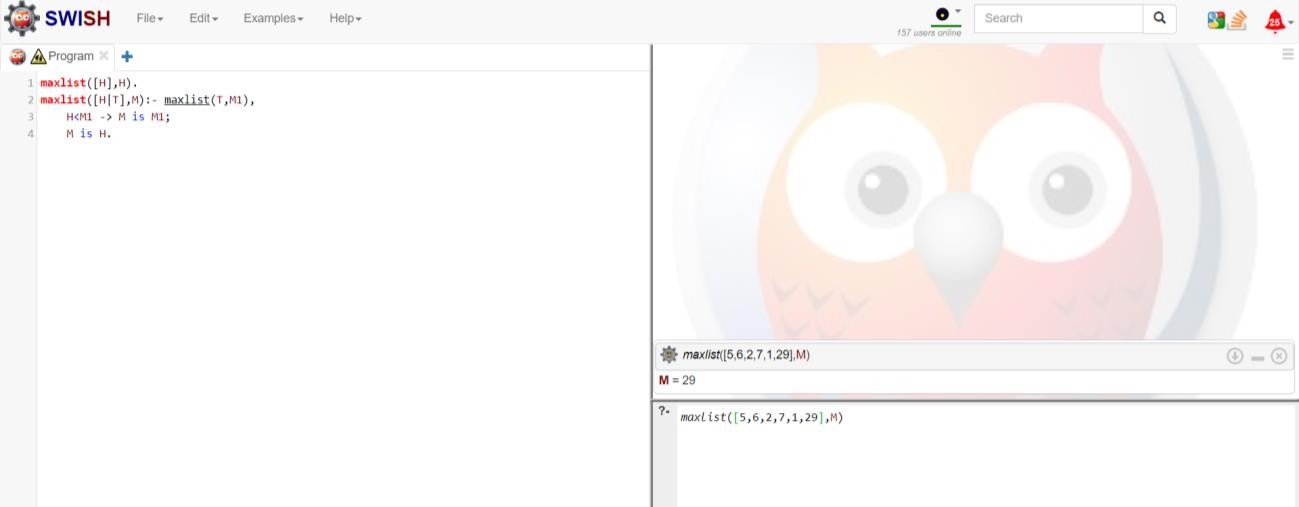
# nth\_element(1,[H|\_T],H). nth\_element(N,[\_H|T],X):- N1 is N-1,

**nth\_element(N1,T,X).**

1. Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.

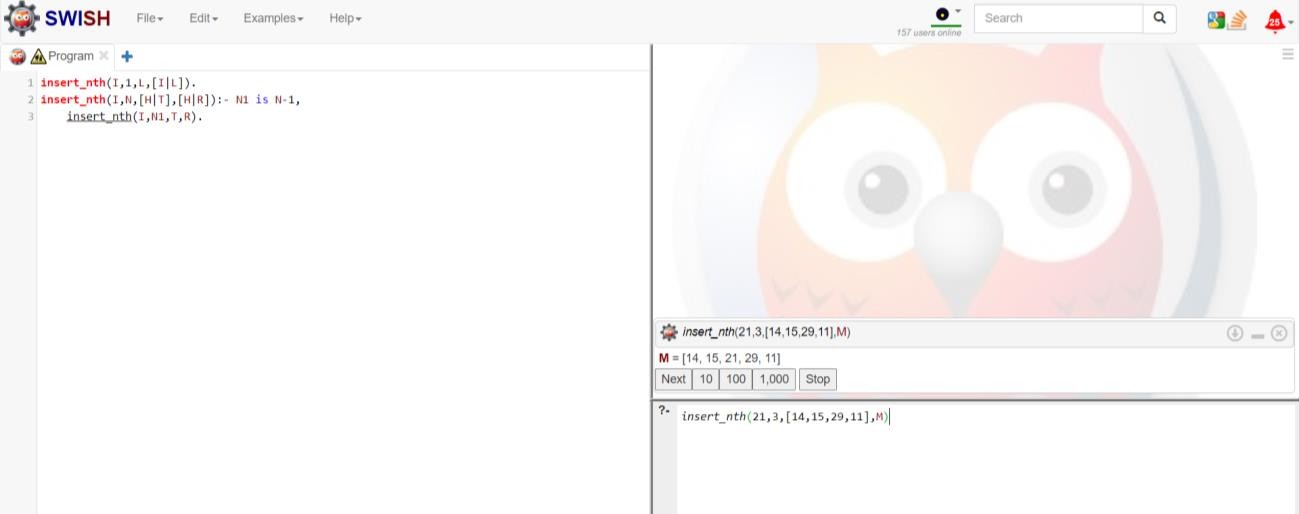
# maxlist([H],H).

**maxlist([H|T],M):- maxlist(T,M1), H<M1 -> M is M1;**

**M is H.**

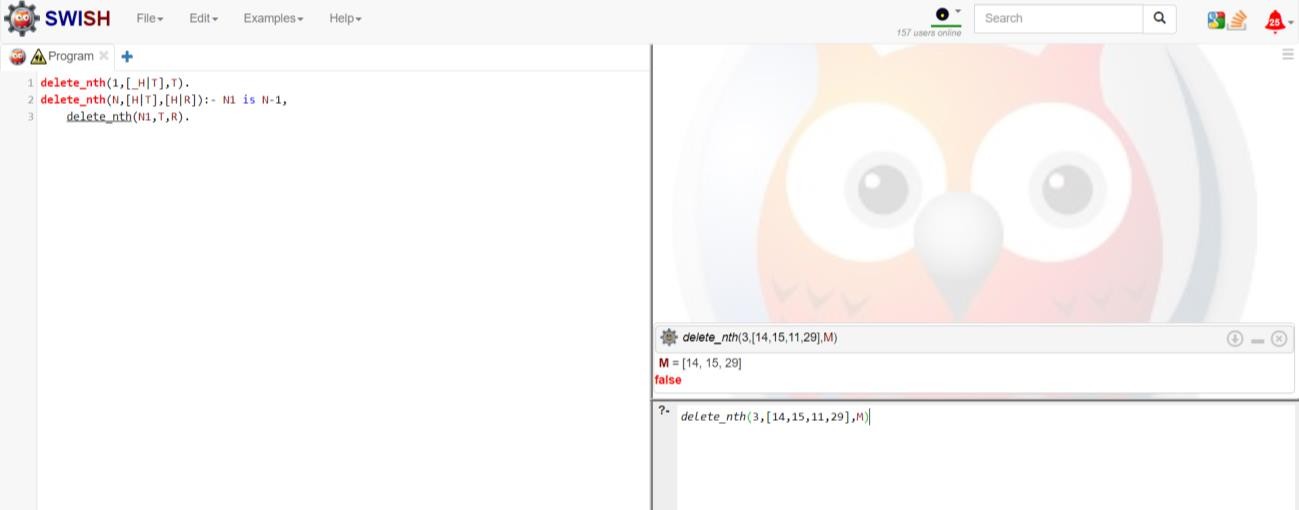
1. Write a prolog program to implement insert\_nth (I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.

# insert\_nth(I,1,L,[I|L]). insert\_nth(I,N,[H|T],[H|R]):- N1 is N-1,

**insert\_nth(I,N1,T,R).**

1. Write a Prolog program to implement delete\_nth (N, L, R) that removes the element on Nth position from a list L to generate a list R.

# delete\_nth(1,[\_H|T],T). delete\_nth(N,[H|T],[H|R]):- N1 is N-1,

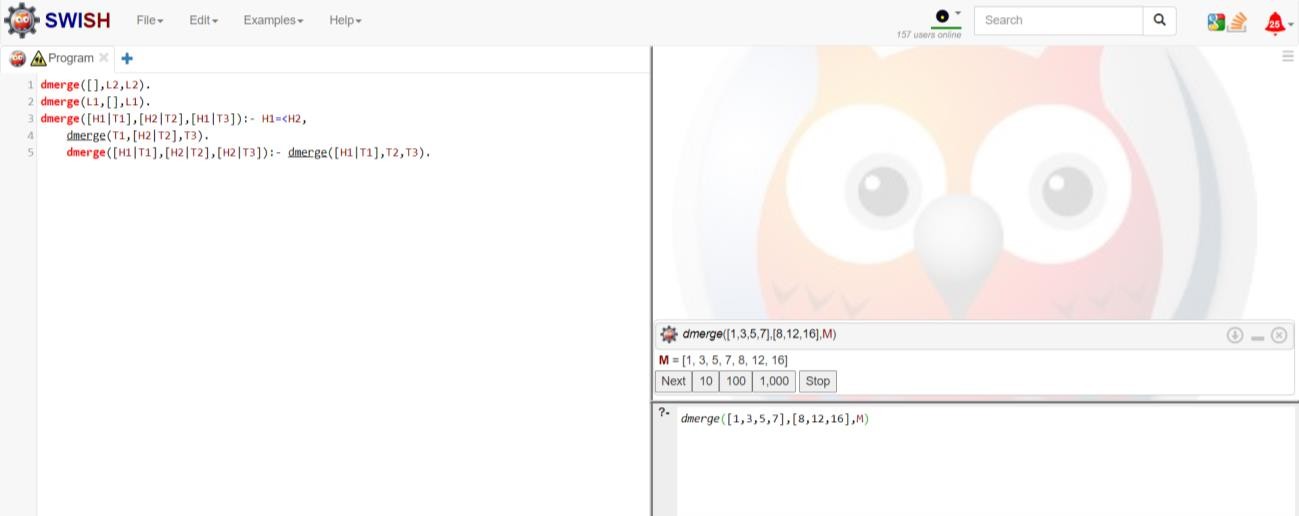
**delete\_nth(N1,T,R).**

1. Write a program in PROLOG to implement merge (L1, L2, L3) where L1 is first ordered list and L2 is second ordered list and L3 represents the merged list.

# dmerge([],L2,L2).

**dmerge(L1,[],L1). dmerge([H1|T1],[H2|T2],[H1|T3]):- H1=<H2,**

**dmerge(T1,[H2|T2],T3).**

**dmerge([H1|T1],[H2|T2],[H2|T3]):- dmerge([H1|T1],T2,T3).**