

Artificial Intelligence Practical Record File

SUPERVISOR: Prof. Vibha Gaur, Ms. Amrita, Mr. Mehtab Alam

SUBMITTED BY

Amartya Sinha - AC-1207



2023

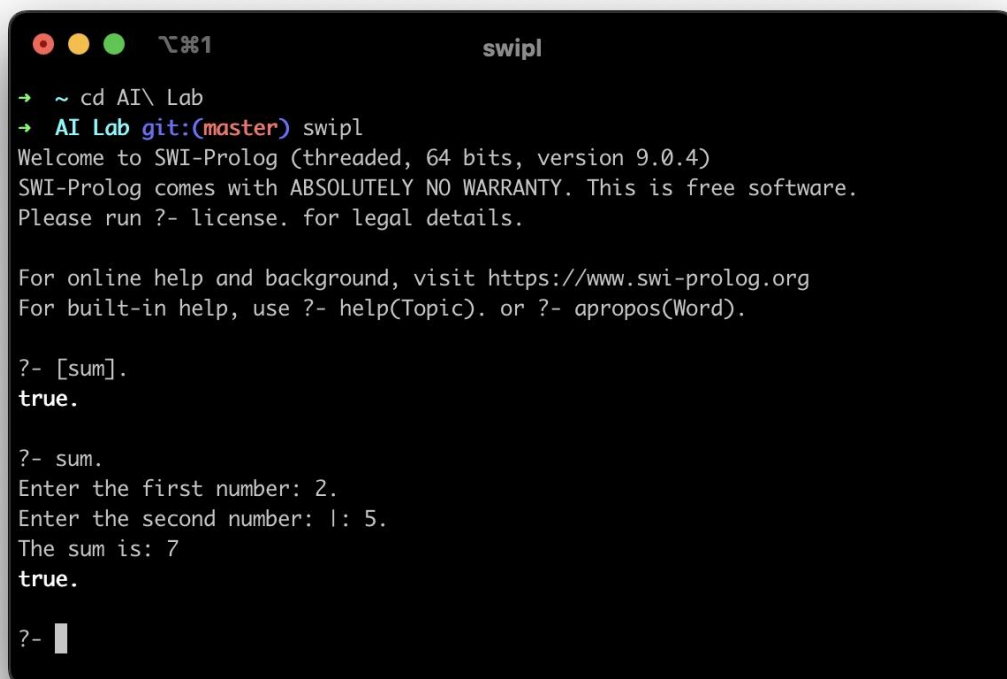
Department of Computer Science
ACHARYA NARENDRA DEV COLLEGE

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

Code:

```
sum :-  
    write('Enter the first number: '),  
    read(X),  
    write('Enter the second number: '),  
    read(Y),  
    Z is X + Y,  
    write('The sum is: '),  
    write(Z).
```

Output:



```
swipl  
→ ~ cd AI\ Lab  
→ AI Lab git:(master) swipl  
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit https://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?- [sum].  
true.  
  
?- sum.  
Enter the first number: 2.  
Enter the second number: 1: 5.  
The sum is: 7  
true.  
  
?-
```

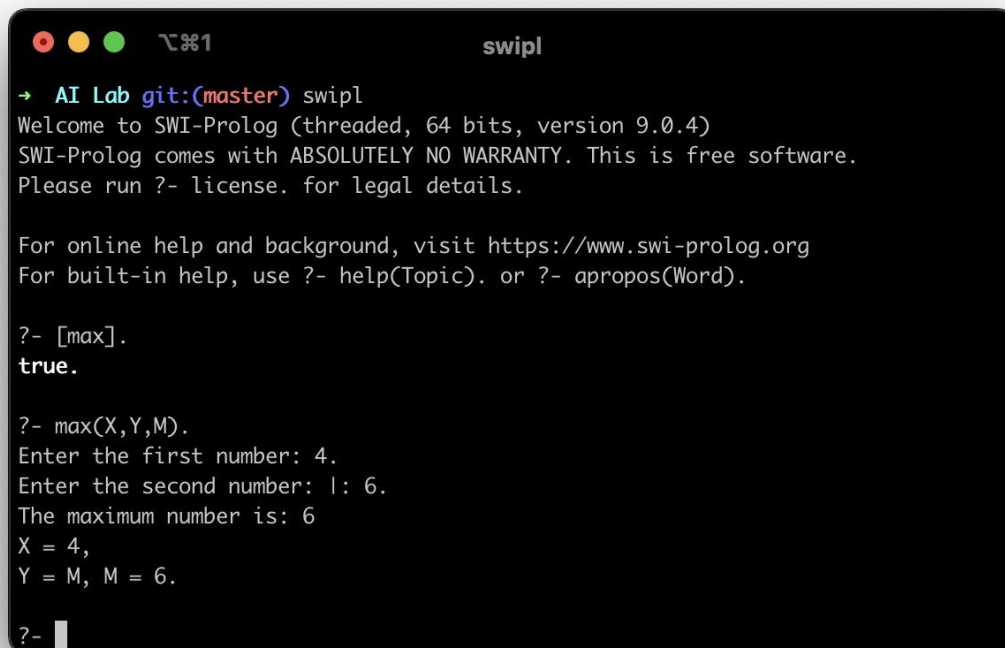
2. Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.

Code:

```
max(X, Y, M) :-  
    write('Enter the first number: '),  
    read(X),  
    write('Enter the second number: '),  
    read(Y),
```

```
(X >= Y -> M = X; M = Y),
write('The maximum number is: '),
write(M).
```

Output:



```
AI Lab git:(master) swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [max].
true.

?- max(X,Y,M).
Enter the first number: 4.
Enter the second number: 1: 6.
The maximum number is: 6
X = 4,
Y = M, M = 6.

?- 
```

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

Code:

```
:- initialization(main). % Call main predicate on program start

main :-
    write('Enter the number: '), % Prompt for input
    read(N), % Read the N from user input

    factorial(N,F), % Call factorial predicate to find the factorial of
N
    write('Factorial: '), write(F), nl, % Display the result
    halt. % Terminate the program

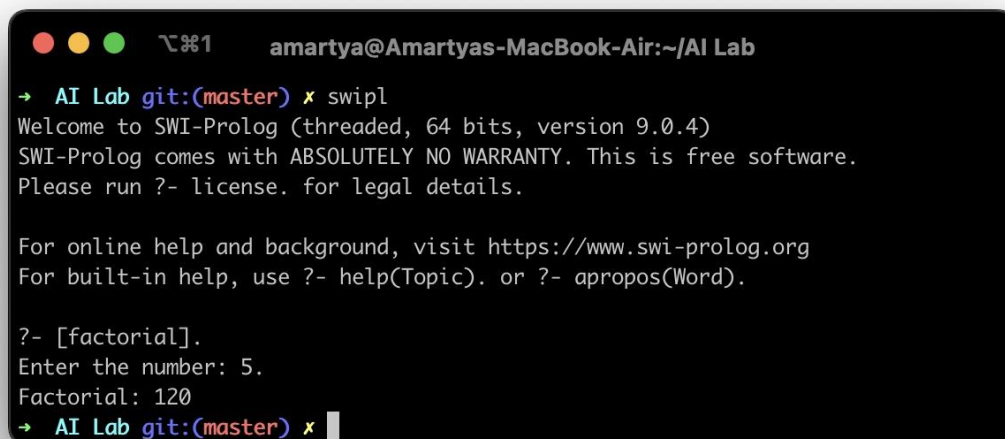
factorial(0,1).
factorial(N,F):-
```

```

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

```

Output:



```

amartya@Amartyas-MacBook-Air:~/AI Lab
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [factorial].
Enter the number: 5.
Factorial: 120
→ AI Lab git:(master) x

```

4. Write a program in PROLOG to implement generate_fib(N,T) where T represents the nth term of the fibonacci series.

Code:

```

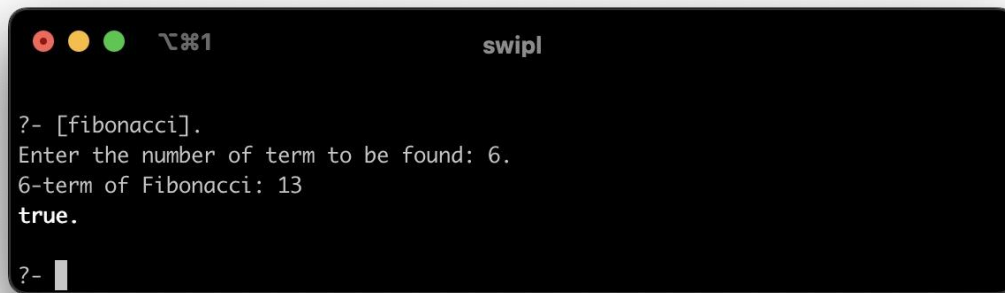
:- initialization(main). % Call main predicate on program start

main :-
    write('Enter the number of term to be found: '), % Prompt for input
    read(N), % Read the N from user input
    generate_fib(N,T),
    write(N), write('-term of Fibonacci: '), write(T), nl. % Display the
result

generate_fib(0,1).
generate_fib(1,1).
generate_fib(N,T) :-
    N1 is N-1,
    N2 is N-2,
    generate_fib(N1, T1),
    generate_fib(N2, T2),
    T is T1 + T2.

```

Output:



```
?- [fibonacci].
Enter the number of term to be found: 6.
6-term of Fibonacci: 13
true.
?- 
```

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

Code:

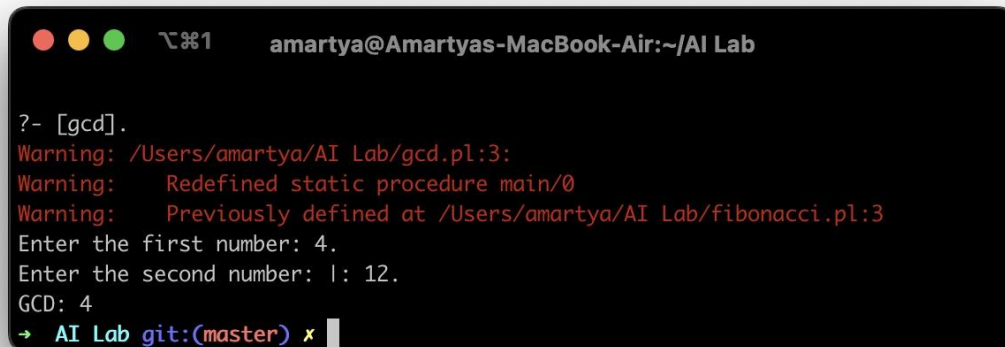
```
:- initialization(main). % Call main predicate on program start

main :-
    write('Enter the first number: '), % Prompt for input
    read(X), % Read the X from user input
    write('Enter the second number: '), % Prompt for input
    read(Y), % Read the Y from user input
    gcd(X, Y, Result), % Call gcd predicate to find GCD of two numbers
    write('GCD: '), write(Result), nl, % Display the result
    halt. % Terminate the program

gcd(X,Y,Result):- (
    X=0 -> (
        Result is Y
    );
    Y=0 -> (
        Result is X
    );
    X=Y -> (
        Result is X
    );
    X>Y -> (
        Y1 is X-Y,
        gcd(Y1,Y,Result)
    );
    X<Y->(
        Y1 is Y-X,
        gcd(X,Y1,Result)
    )
);
```

).
)

Output:



```
amartya@Amartyas-MacBook-Air:~/AI Lab
?- [gcd].
Warning: /Users/amartya/AI Lab/gcd.pl:3:
Warning:   Redefined static procedure main/0
Warning:   Previously defined at /Users/amartya/AI Lab/fibonacci.pl:3
Enter the first number: 4.
Enter the second number: 1: 12.
GCD: 4
→ AI Lab git:(master) x
```

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

Code:

```
power :-
    write('Enter the number: '),
    read(Num),
    write('Enter the power: '),
    read(Pow),
    power(Num, Pow, Ans),
    write(Num), write(' raised to the power of '), write(Pow),
    write(' is '), write(Ans), nl.

power(_, 0, 1).
power(Num, Pow, Ans) :-
    Pow > 0,
    NewPow is Pow - 1,
    power(Num, NewPow, NewAns),
    Ans is Num * NewAns.
```

Output:



```
AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [power].
true.

?- power.
Enter the number: 5.
Enter the power: 1: 2.
5 raised to the power of 2 is 25
true .

?- 
```

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

Code:

```
:- initialization(main). % Call main predicate on program start

main :-
    write('Enter the first number: '), % Prompt for input
    read(N1), % Read the N1 from user input
    write('Enter the second number: '), % Prompt for input
    read(N2), % Read the N2 from user input
    multi(N1, N2, R), % Call multi predicate to multiply N1 and N2
    write('Product: '), write(R), nl, % Display the result
    halt. % Terminate the program

multi(0, _, 0).
multi(_, 0, 0).
multi(N1, N2, R) :-
    R is N1 * N2.
```

Output:

```
amartya@Amartyas-MacBook-Air:~/AI Lab
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [multiply].
Enter the first number: 4.
Enter the second number: 1: 5.
Product: 20
→ AI Lab git:(master) x
```

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

Code:

```
memb :-
    write('Enter a list: '),
    read(L),
    write('Enter an element: '),
    read(X),
    memb(X, L),
    write(X), write(' is a member of '), write(L), nl.

memb(X, [X|_]).
memb(X, [_|T]) :- memb(X, T).
```

Output:


```
swipl

→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [member].
true.

?- memb.
Enter a list: [2,3,4,1,6,5,9]
l: ].
Enter an element: l: 4.
4 is a member of [2,3,4,1,6,5,9]
true .

?-
```

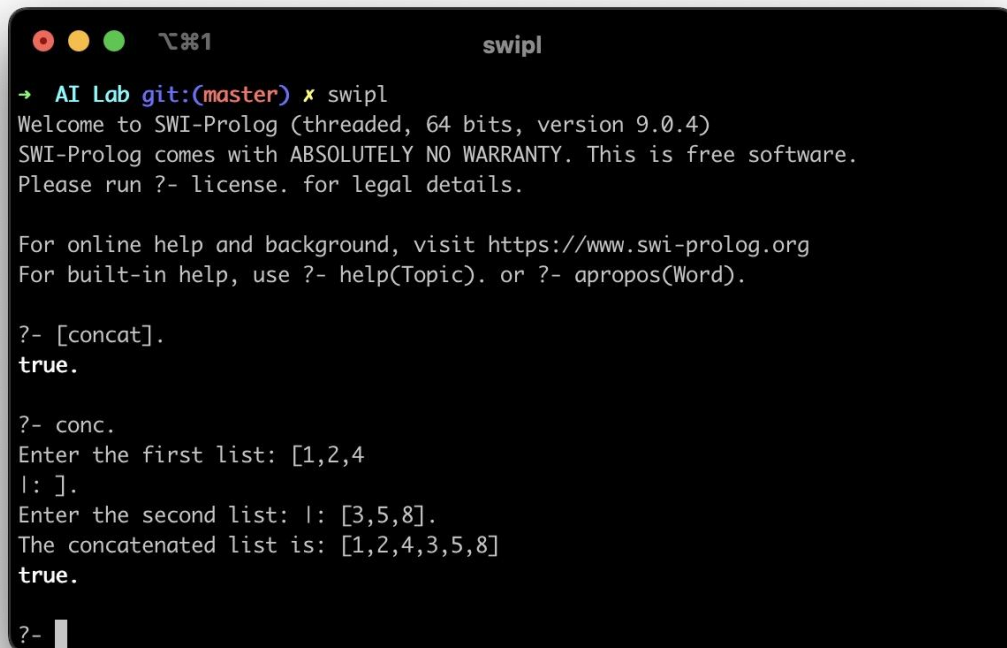
9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.

Code:

```
conc :-
    write('Enter the first list: '),
    read(L1),
    write('Enter the second list: '),
    read(L2),
    conc(L1, L2, L3),
    write('The concatenated list is: '), write(L3), nl.

conc([], L, L).
conc([H|T], L2, [H|L3]) :- conc(T, L2, L3).
```

Output:



```
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [concat].
true.

?- conc.
Enter the first list: [1,2,4
l: ].
Enter the second list: l: [3,5,8].
The concatenated list is: [1,2,4,3,5,8]
true.

?- 
```

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

Code:

```
reverse([], []) :-
    !.
reverse([H|T], R) :-
    reverse(T, TR),
    append(TR, [H], R).

reverse(L, R) :-
    write('Enter the list: '),
    read(L),
    reverse(L, R),
    write('The reversed list of '), write(L), write(' is '), write(R).
```

Output:

```
amartya@Amartyas-MacBook-Air:~/AI Lab
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [reverse].
Enter a list: [4,5,3,2]
l: [].
Reversed list: [2,3,5,4]
→ AI Lab git:(master) x
```

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


Code:

```
palindrome :-
    write('Enter a list: '),
    read(L),
    palindrome(L),
    write('The list is a palindrome.').

palindrome(L) :- reverse(L, L).

reverse([], []).
reverse([H|T], R) :- reverse(T, TR), append(TR, [H], R).
```

Output:



```
AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [palindrome].
true.

?- palindrome.
Enter a list: [4,3,1,3,4]
[]: ].
The list is a palindrome.
true.

?- 
```

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

Code:

```
sumlist([], 0).
sumlist([X|Xs], S) :-
    write('Enter the next element of the list: '),
    read(X),
    sumlist(Xs, S1),
    S is X + S1.
```

Output:



```
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [sumlist].
Enter a list of numbers: [4,3,5,2
|: ].
Sum: 14
true.

?-
```

13. 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.

Code:

```
evenlength([]).
evenlength([_,Xs|T]) :- evenlength(T).

oddlength([_]).
oddlength([_,Xs|T]) :- oddlength(T).

main :-
    write('Enter a list: '),
    read(L),
    (evenlength(L) -> write('List has even length.') ; write('List does
not have even length.')),
    nl,
    (oddlength(L) -> write('List has odd length.') ; write('List does not
have odd length.')),
    nl.
```

Output:

```
swipl
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [evenlength_oddlength].
Warning: /Users/amartya/AI Lab/evenlength_oddlength.pl:2:
Warning: Singleton variables: [Xs]
Warning: /Users/amartya/AI Lab/evenlength_oddlength.pl:5:
Warning: Singleton variables: [Xs]
true.

?- main.
Enter a list: [4,3,2,8
|: ].
List has even length.
List does not have odd length.
true.

?-
```

14. 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 `N`th element of `L`.

Code:

```
:- initialization(main). % Call main predicate on program start

main :-
    write('Enter a list of elements: '), % Prompt for input
    read(L), % Read the list from user input
    write('Enter the desired position (N): '), % Prompt for N
    read(N), % Read N from user input
    nth_element(N, L, X), % Call nth_element predicate to retrieve the
nth element
    write('The element at position '), write(N), write(' is: '),
write(X), nl, % Display the result
    halt. % Terminate the program

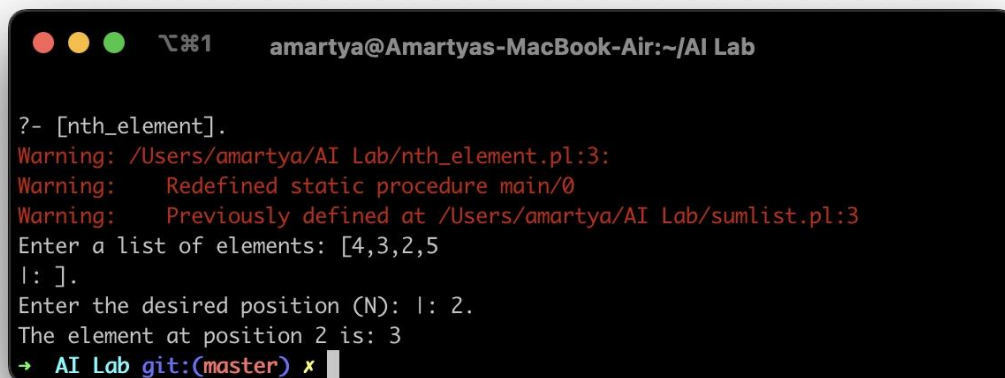
nth_element(1, [X | _], X). % Base case: N=1, the desired position,
```

```

retrieve the head of the list
nth_element(N, [_ | L], X) :-
    N > 1, % Ensure N is a positive integer
    N1 is N - 1, % Decrement N by 1
    nth_element(N1, L, X). % Recursive rule: retrieve the nth element
from the tail of the list

```

Output:



```

amartya@Amartyas-MacBook-Air:~/AI Lab
?- [nth_element].
Warning: /Users/amartya/AI Lab/nth_element.pl:3:
Warning:   Redefined static procedure main/0
Warning:   Previously defined at /Users/amartya/AI Lab/sumlist.pl:3
Enter a list of elements: [4,3,2,5
l: ].
Enter the desired position (N): l: 2.
The element at position 2 is: 3
→ AI Lab git:(master) *

```

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

Code:

```

% Read a list from the user
read_list(L) :-
    write('Enter a list: '),
    read(L).

% Find the maximum number in a list
maxlist(L, M) :-
    max_list(L, M).

% Main predicate to read the list and find the maximum number
main :-
    read_list(L),
    maxlist(L, M),
    write('The maximum number in '), write(L), write(' is '), write(M).

```

Output:

```
swipl

?- [maxlist].
Warning: /Users/amartya/AI Lab/maxlist.pl:2:
Warning:   Redefined static procedure read_list/1
Warning:   Previously defined at /Users/amartya/AI Lab/nth_element.pl:2
Warning: /Users/amartya/AI Lab/maxlist.pl:11:
Warning:   Redefined static procedure main/0
Warning:   Previously defined at /Users/amartya/AI Lab/nth_element.pl:16
true.

?- main.
Enter a list: [4,5,3,2
l: ].
The maximum number in [4,5,3,2] is 5
true.

?- 
```

16. 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.

Code:

```
:- initialization(main). % Call main predicate on program start

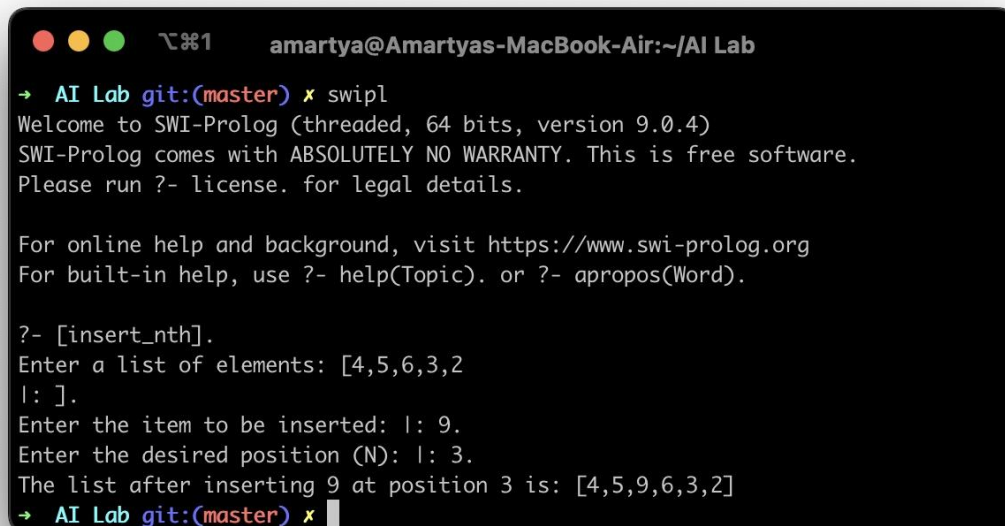
main :-
    write('Enter a list of elements: '), % Prompt for input
    read(L), % Read the list from user input
    write('Enter the item to be inserted: '), % Prompt for I
    read(I), % Read I from user input
    write('Enter the desired position (N): '), % Prompt for N
    read(N), % Read N from user input
    insert_nth(I, N, L, R), % Call insert_nth predicate to insert I into
the nth position of L
    write('The list after inserting '), write(I), write(' at position
'), write(N), write(' is: '), write(R), nl, % Display the result
    halt. % Terminate the program

insert_nth(I, 1, L, [I | L]). % Base case: insert I as the head of the
list when N=1
insert_nth(I, N, [X | L], [X | R]) :-
    N > 1, % Ensure N is a positive integer
```



```
N1 is N - 1, % Decrement N by 1
insert_nth(I, N1, L, R). % Recursive rule: insert I into the (N-1)th
position of the tail of the list
```

Output:



```
amartya@Amartyas-MacBook-Air:~/AI Lab
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [insert_nth].
Enter a list of elements: [4,5,6,3,2
l: ].
Enter the item to be inserted: l: 9.
Enter the desired position (N): l: 3.
The list after inserting 9 at position 3 is: [4,5,9,6,3,2]
→ AI Lab git:(master) x
```

17. 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.

Code:

```
:- initialization(main). % Call main predicate on program start

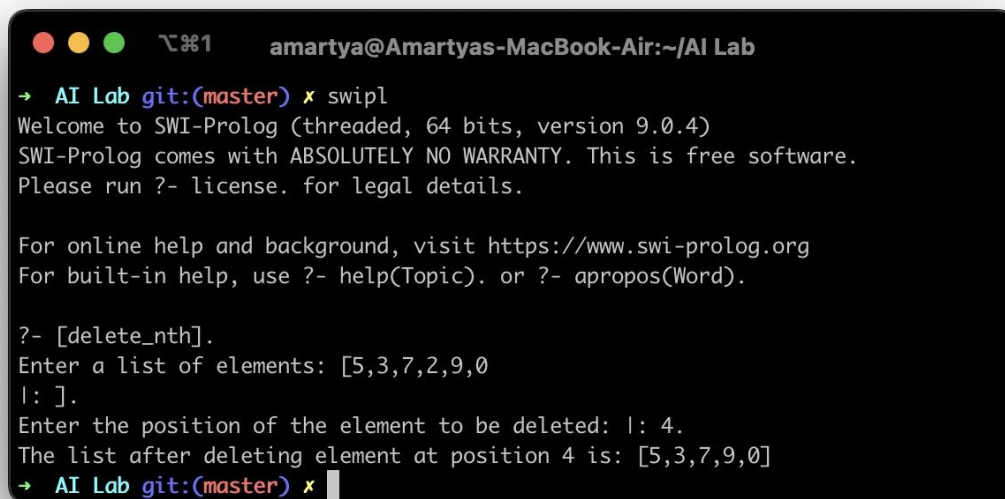
main :-
    write('Enter a list of elements: '), % Prompt for input
    read(L), % Read the list from user input
    write('Enter the position of the element to be deleted: '), % Prompt
    for N
    read(N), % Read N from user input
    delete_nth(N, L, R), % Call delete_nth predicate to remove Nth
    element from L
    write('The list after deleting element at position '), write(N),
    write(' is: '), write(R), nl, % Display the result
    halt. % Terminate the program
```

```

delete_nth(1, [_ | L], L). % Base case: remove head element when N=1
delete_nth(N, [X | L], [X | R]) :-
    N > 1, % Ensure N is a positive integer
    N1 is N - 1, % Decrement N by 1
    delete_nth(N1, L, R). % Recursive rule: remove Nth element from tail
of the list

```

Output:



```

amartya@Amartyas-MacBook-Air:~/AI Lab
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [delete_nth].
Enter a list of elements: [5,3,7,2,9,0]
|: ].
Enter the position of the element to be deleted: |: 4.
The list after deleting element at position 4 is: [5,3,7,9,0]
→ AI Lab git:(master) x

```

18. 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.

Code:

```

:- initialization(main). % Call main predicate on program start

main :-
    write('Enter the first ordered list: '), % Prompt for input
    read(L1), % Read the first ordered list from user input
    write('Enter the second ordered list: '), % Prompt for input
    read(L2), % Read the second ordered list from user input
    merge(L1, L2, L3), % Call merge predicate to merge L1 and L2
    write('The merged list is: '), write(L3), nl, % Display the result
    halt. % Terminate the program

merge([], L, L). % Base case: if L1 is empty, merged list is L2

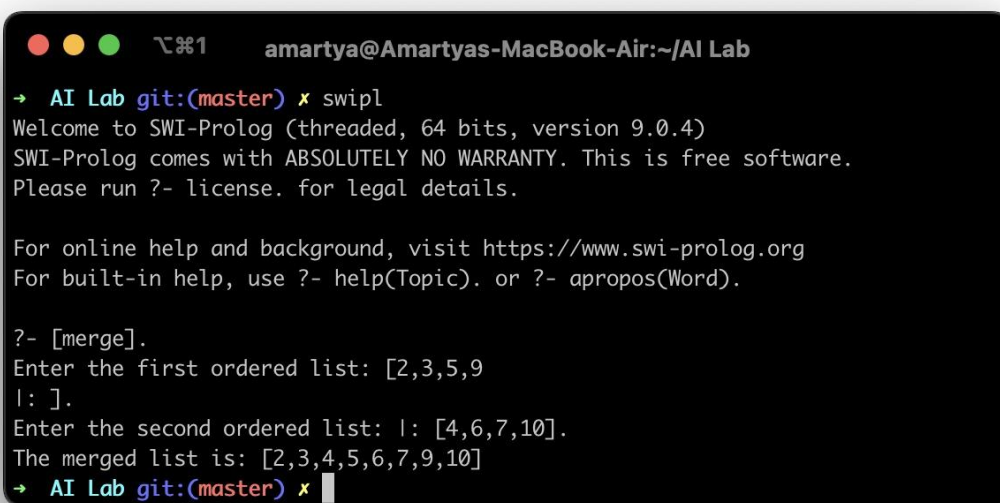
```

```

merge(L, [], L). % Base case: if L2 is empty, merged list is L1
merge([X | L1], [Y | L2], [X | L3]) :-
    X <= Y, % Compare the heads of L1 and L2
    merge(L1, [Y | L2], L3). % Recursive rule: if X <= Y, merge the
tails of L1 and L2
merge([X | L1], [Y | L2], [Y | L3]) :-
    X > Y, % Compare the heads of L1 and L2
    merge([X | L1], L2, L3). % Recursive rule: if X > Y, merge the tails
of L1 and L2

```

Output:



```

amartya@Amartyas-MacBook-Air:~/AI Lab
→ AI Lab git:(master) x swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [merge].
Enter the first ordered list: [2,3,5,9
|: ].
Enter the second ordered list: |: [4,6,7,10].
The merged list is: [2,3,4,5,6,7,9,10]
→ AI Lab git:(master) x

```