BHASKARACHARAYA COLLEGE OF APPLIED SCIENCES

UNIVERSITY OF DELHI

**NAME – HARSH COURSE – B.SC. (HONS.) COMPUTER**

**SCIENCE**

**YEAR – 3RD YEAR (VI SEMESTER)**

***ARTIFICIAL INTELLIGENCE PRACTICAL FILE***

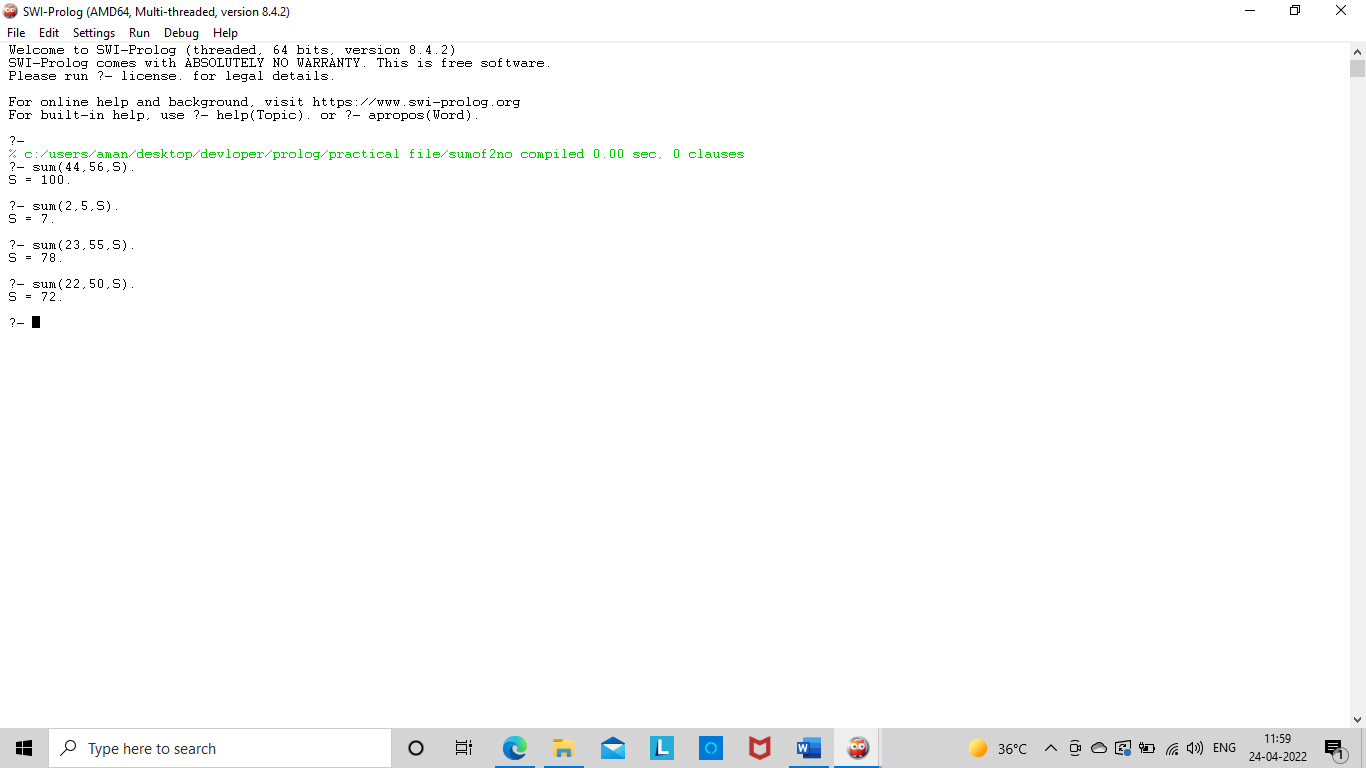
**ROLL NO – 2002026 EXAMINATION ROLL NO - 20009570024**

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

Knowledge Base:

**sum(X,Y,S):- S is X+Y**

Output:

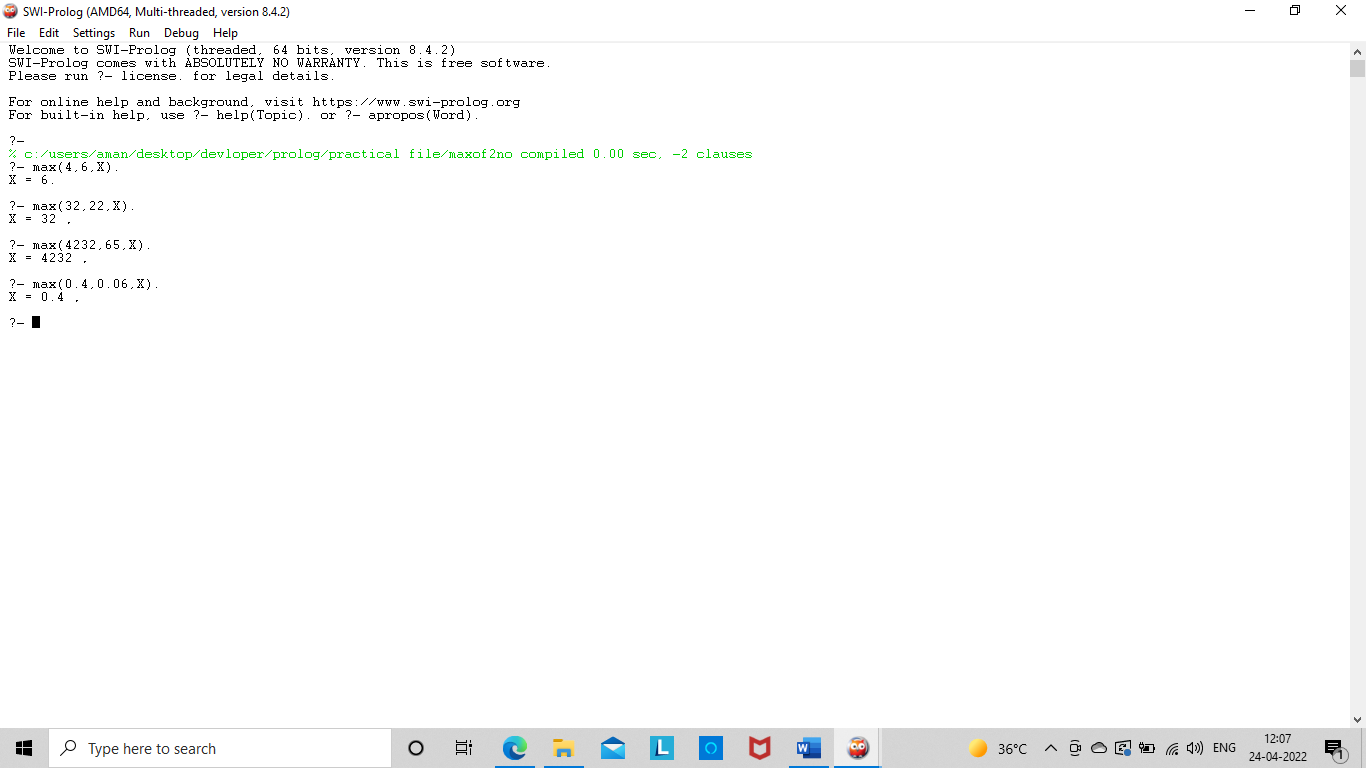


2.Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.

Knowledge Base:

**max(X,Y,M):- X>Y, M is X. max(\_,Y,M):- M is Y.**

Output:

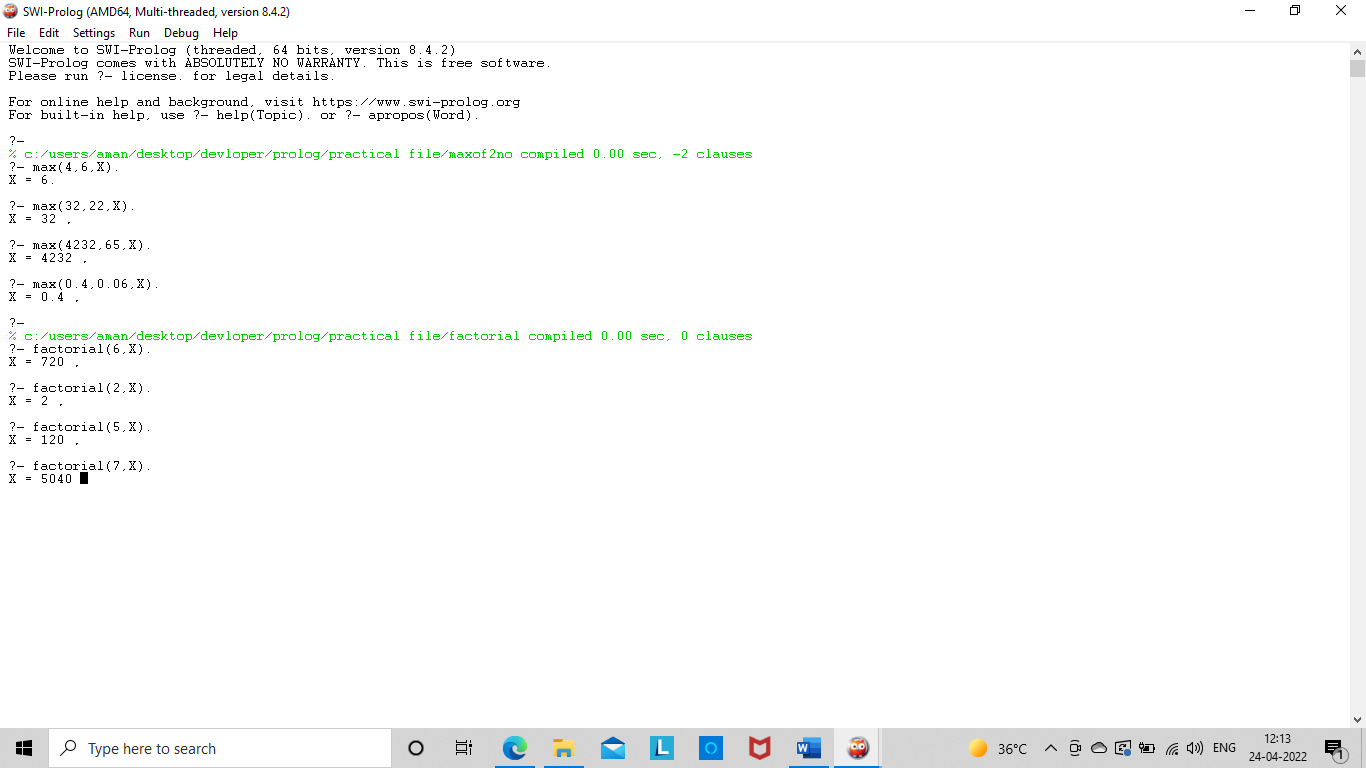


3. Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N

Knowledge Base:

**factorial(0,1). factorial(N,X):- N1 is N-1, factorial(N1,X1), X is X1\*N**

Output:



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

Knowledge Base:

**generate\_fib(0,1).**

**generate\_fib(1,1).**

**generate\_fib(N,T):-**

**N1 is N-1,**

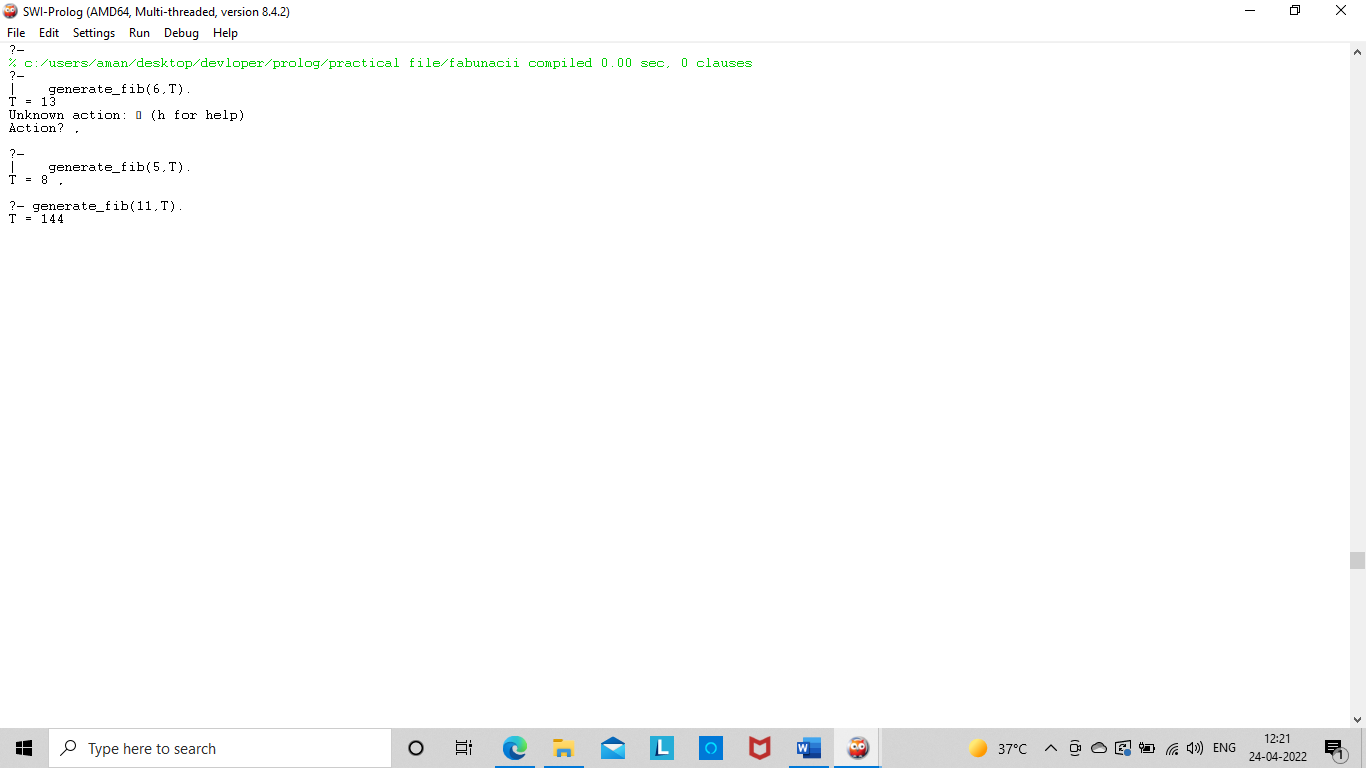
**generate\_fib(N1,T1),**

**N2 is N-2,**

**generate\_fib(N2,T2),**

**T is T1+T2.**

Output:



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

Knowledge Base:

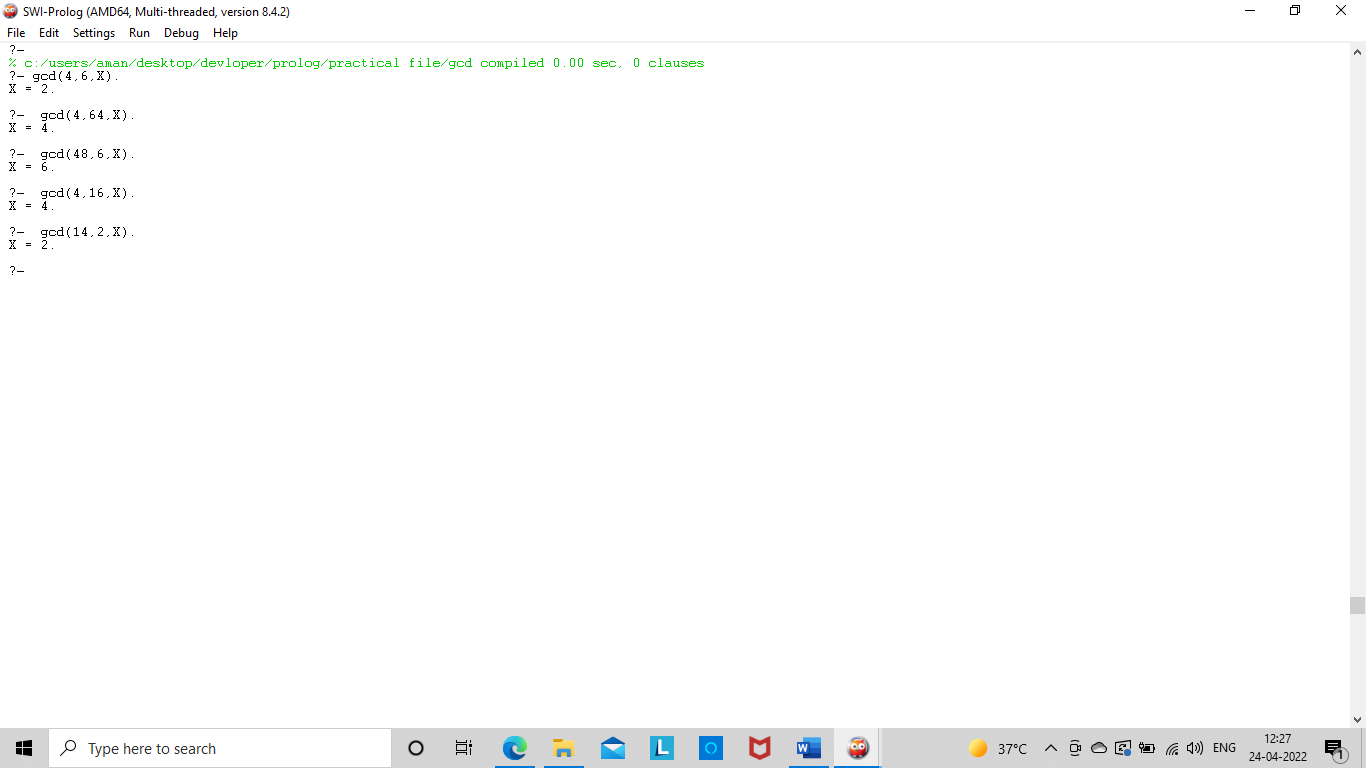
**gcd(M,0,M):-!.**

**gcd(M,N,D):-N > 0,**

**X is mod(M,N),**

**gcd(N,X,D).**

Output:



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

Knowledge Base:

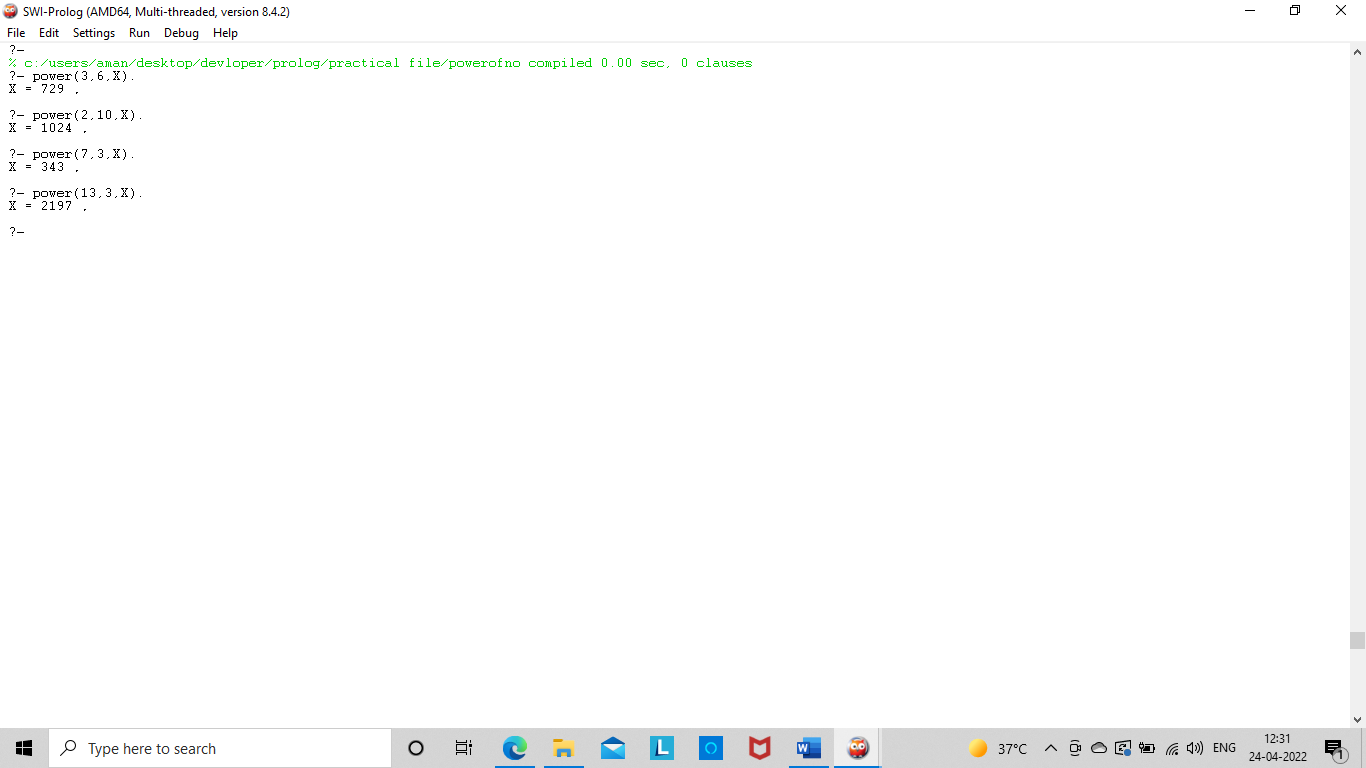
**power(Num,1,Num).**

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

**power(Num,Pow1,Ans1),**

**Ans is Ans1\*Num.**

Output:



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

Knowledge Base:

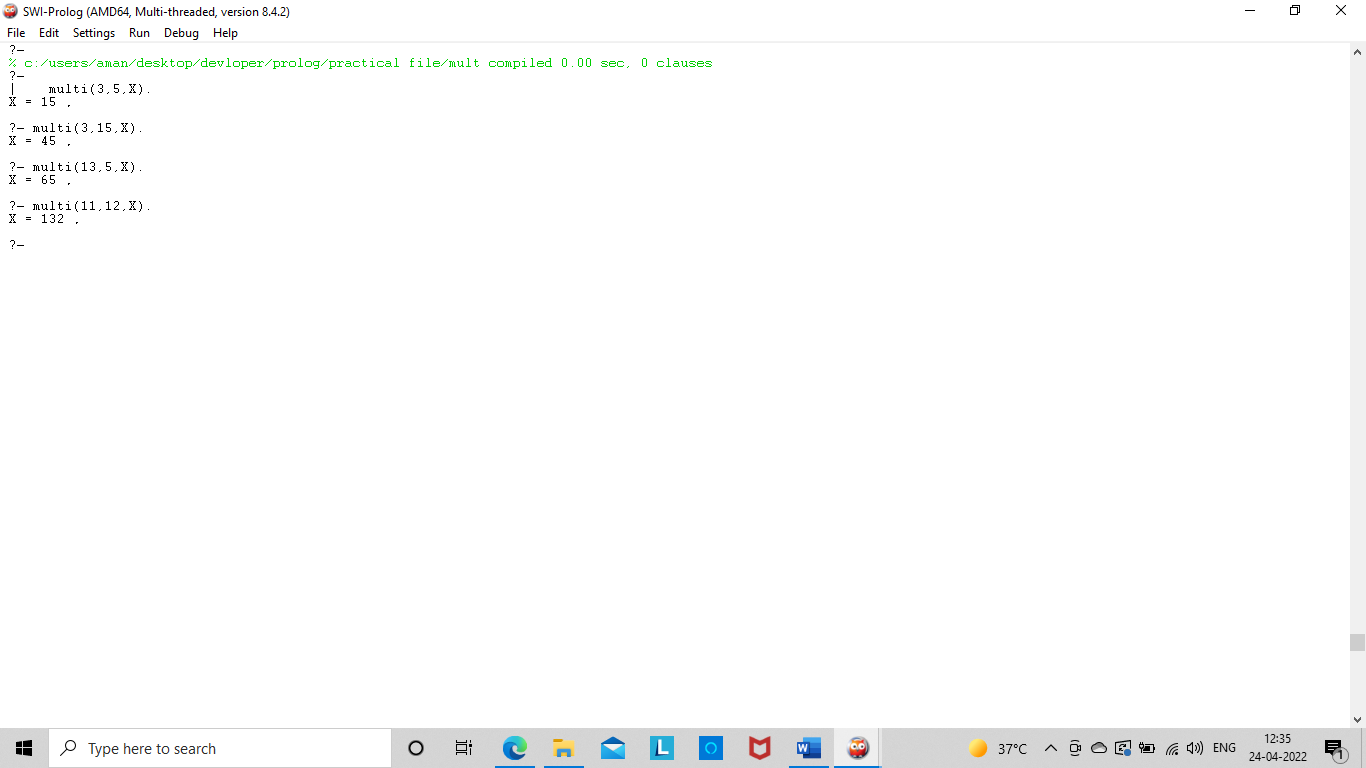
multi(N1,1,N1).

multi(N1,N2,Ans):- Temp is N2-1,

multi(N1,Temp,Ans1),

Ans is Ans1+N1

Output:



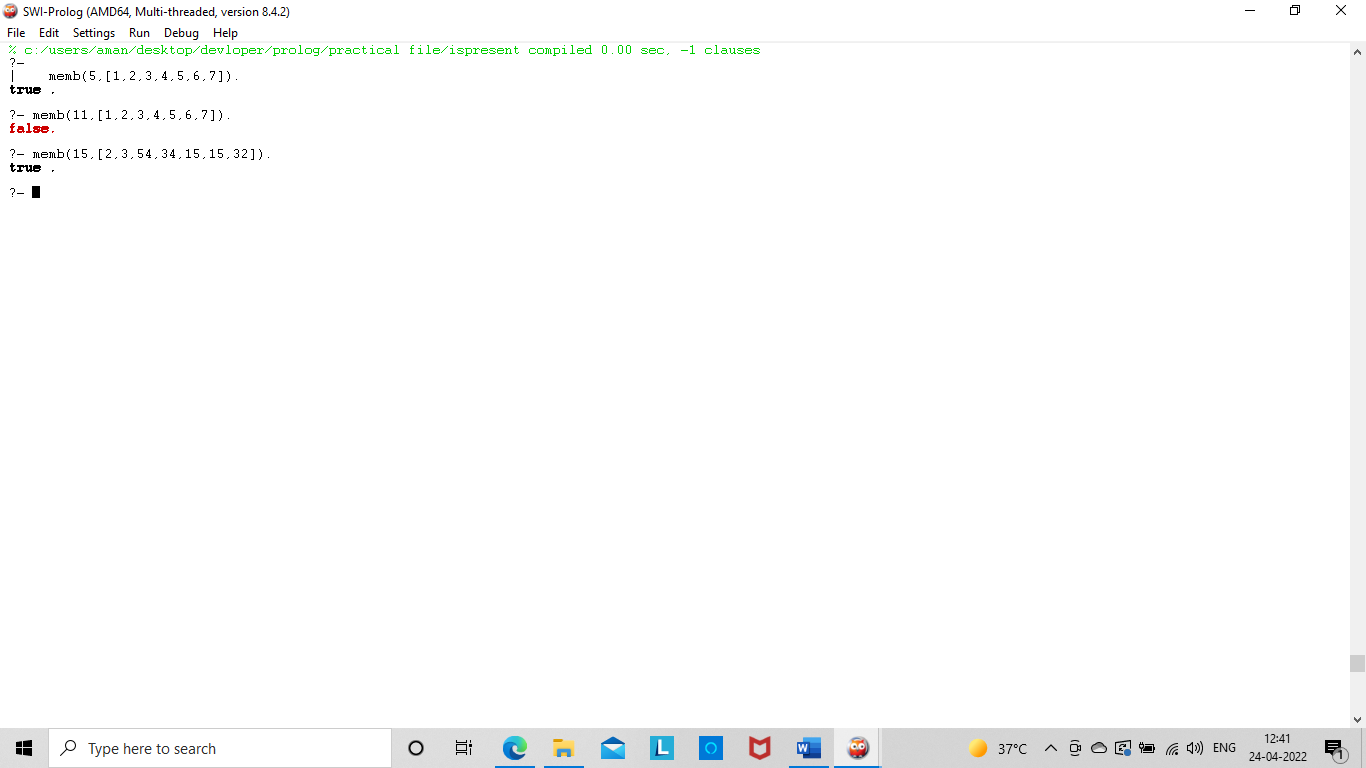
**8.** Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.

Knowledge Base:

**memb(X, [X|Tail]).**

**memb(X, [Head|Tail]):- memb(X, Tail).**

Output:



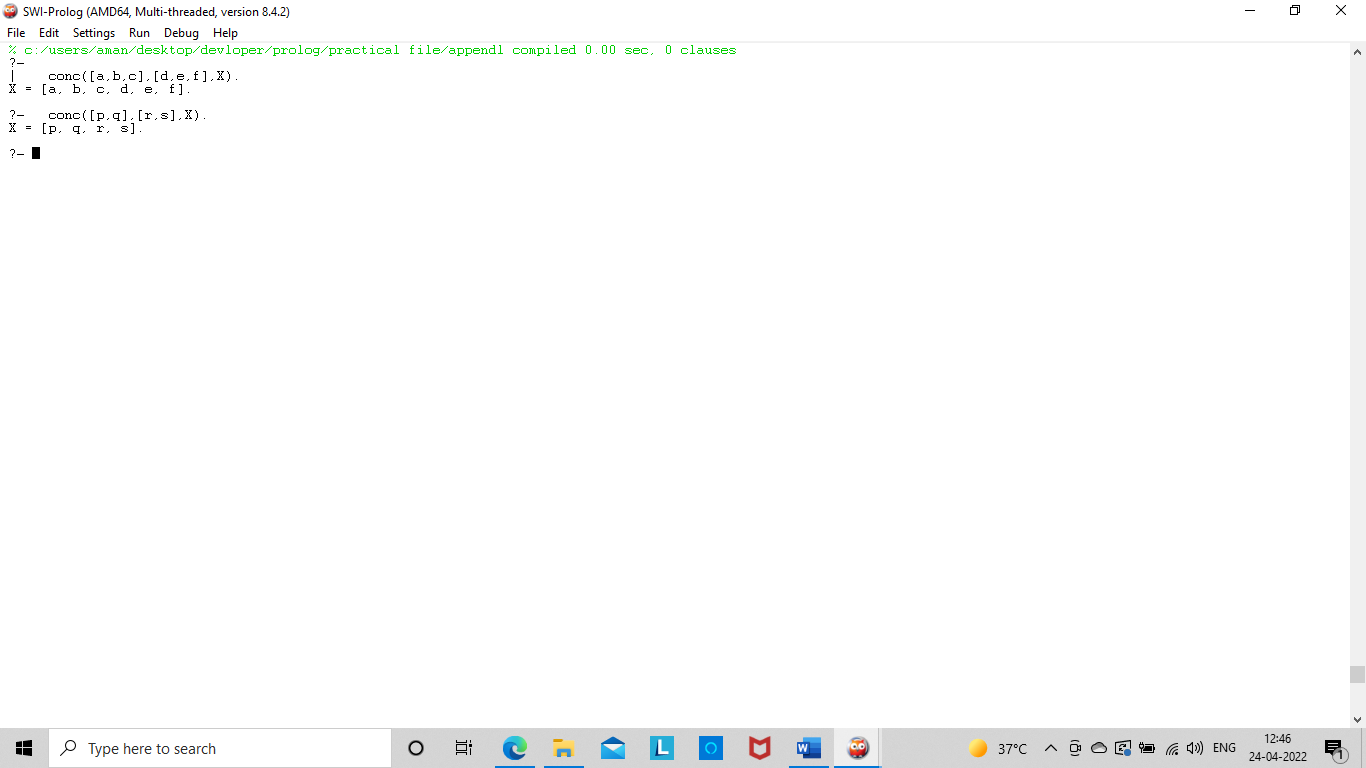
**9.** Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3

Knowledge Base:

**conc([],L,L).**

**conc([X|M],N,[X|Q]):- conc(M,N,Q)**

Output:



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

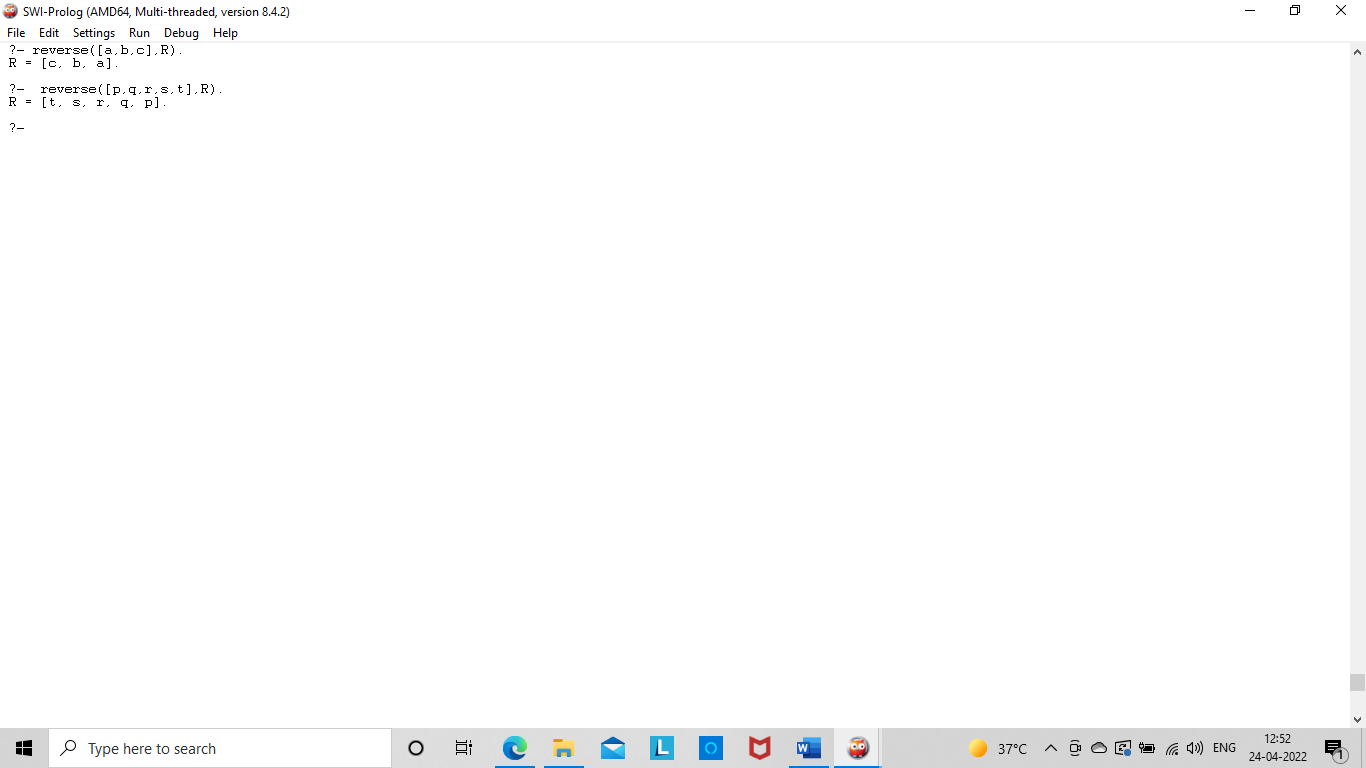
Knowledge Base:

**reverse([H|T],R):- length(T,L),**

**L>0 ->(reverse(T,R1),R is H) ;**

**R is H.**

Output:



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

Knowledge Base:

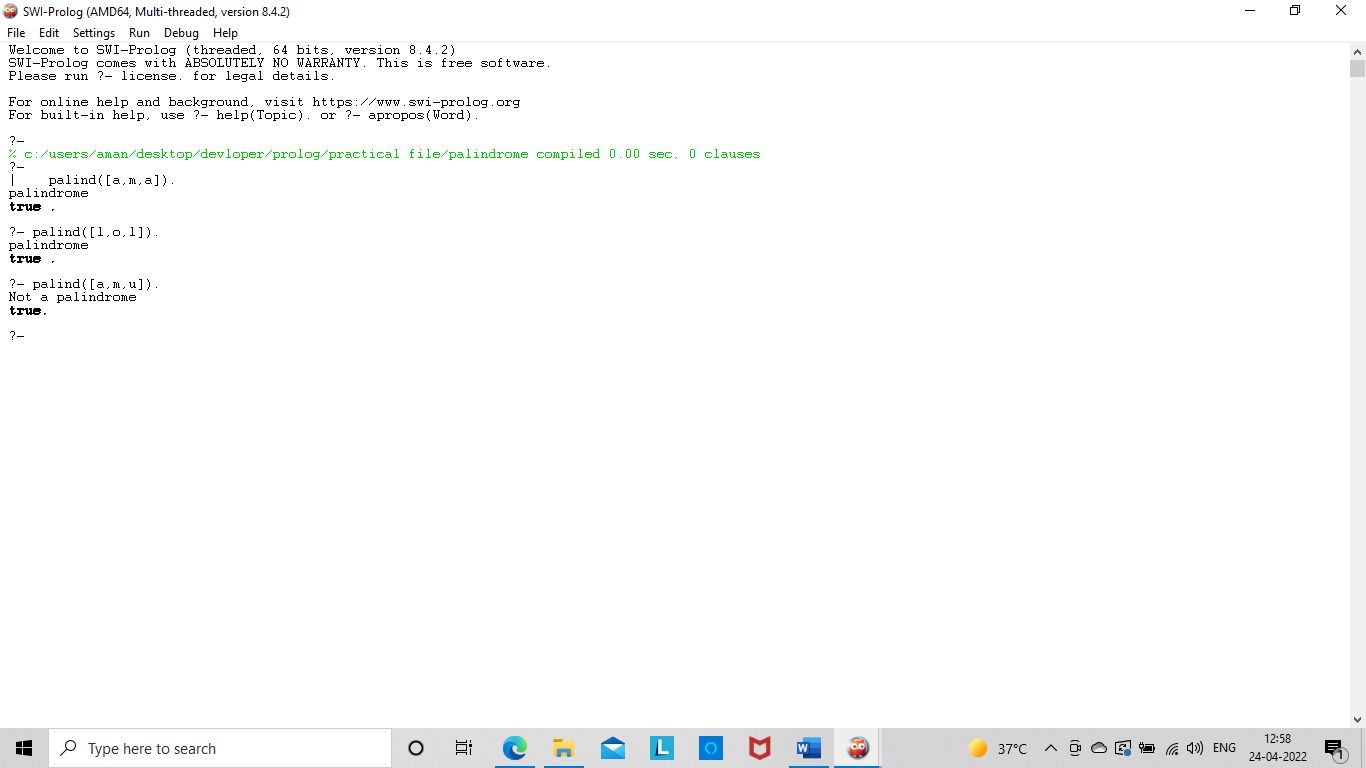
**palind([]):- write('palindrome').**

**palind([\_]):- write('palindrome').**

**palind(L) :- append([H|T], [H], L),**

**palind(T) ; write('Not a palindrome').**

Output:



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

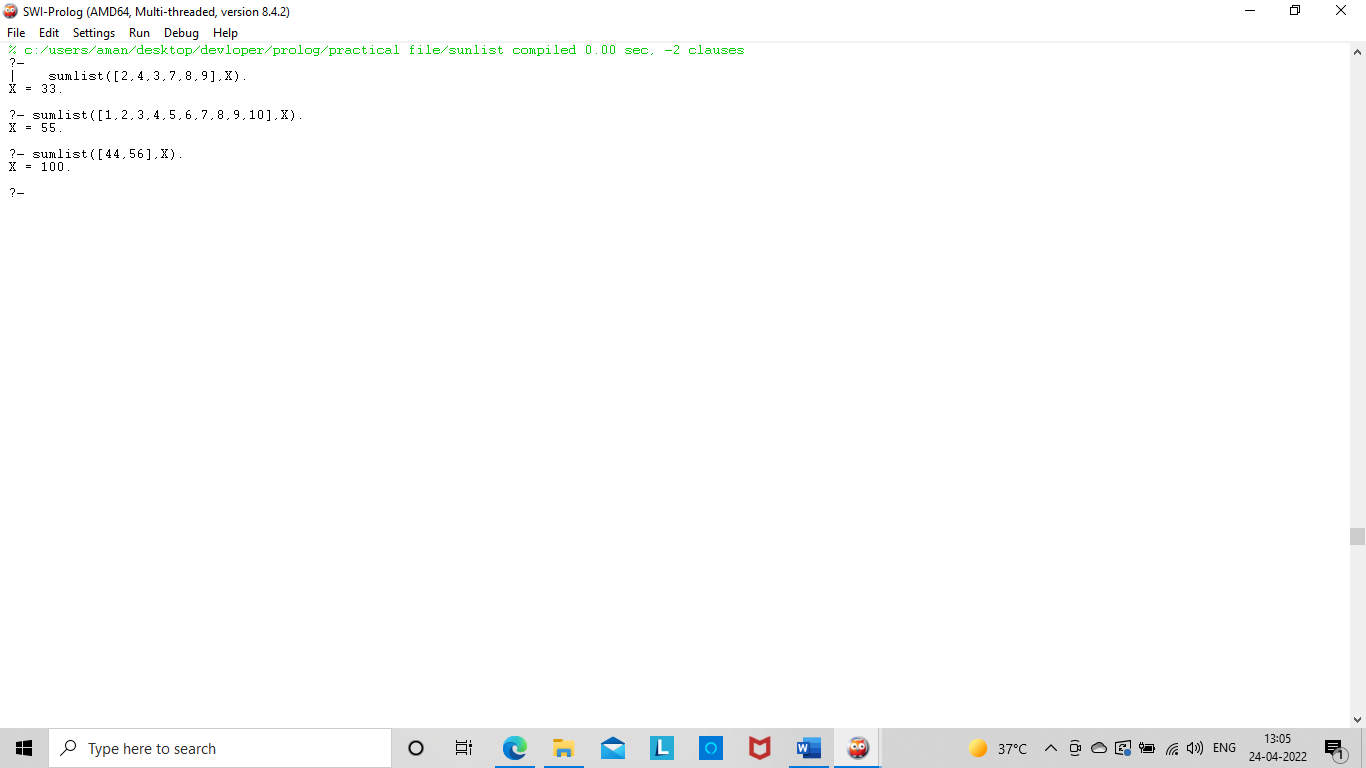
Knowledge Base:

**sumlist([],0).**

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

**S is H+S1.**

Output:



**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

Knowledge Base:

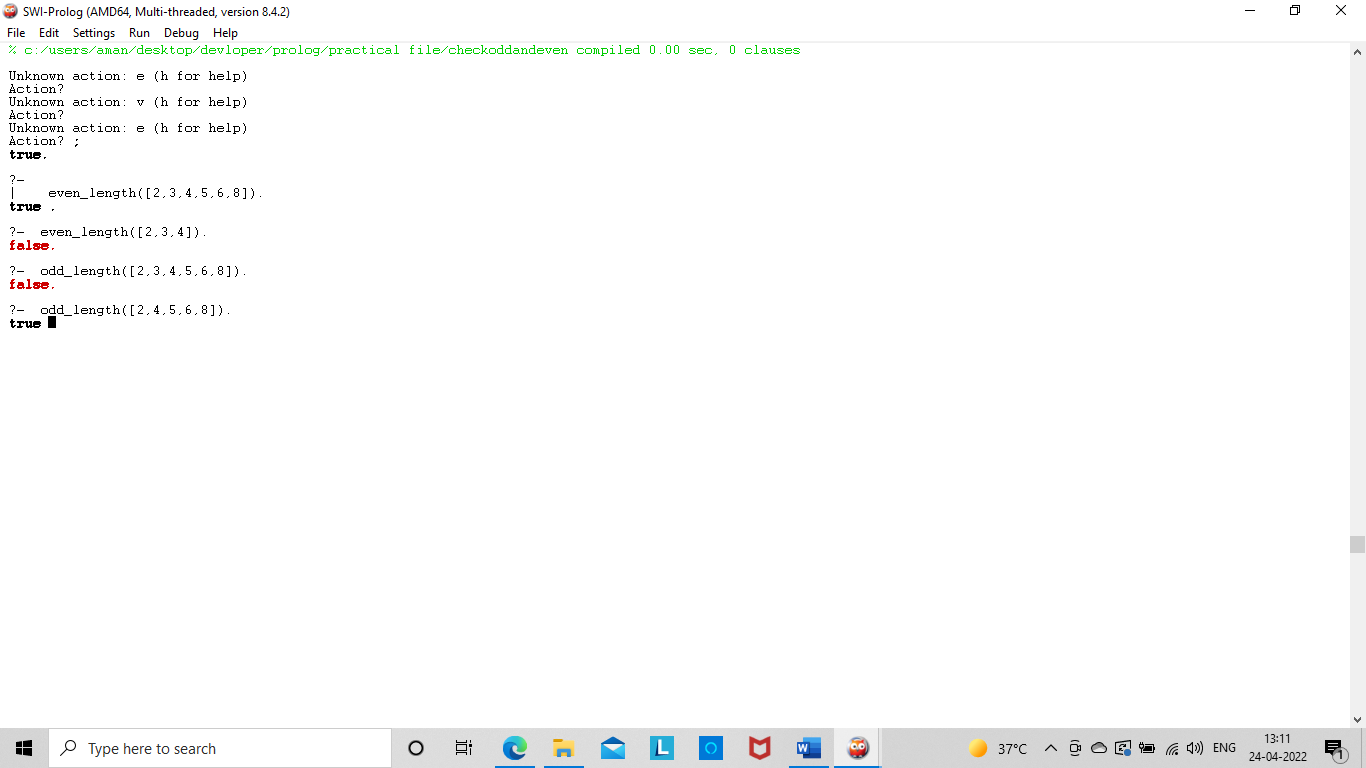
**even\_length([]).**

**even\_length([\_|T]):- odd\_length(T).**

**odd\_length([\_]).**

**odd\_length([\_|T]):- even\_length(T).**

Output:



**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 Nth element of L.

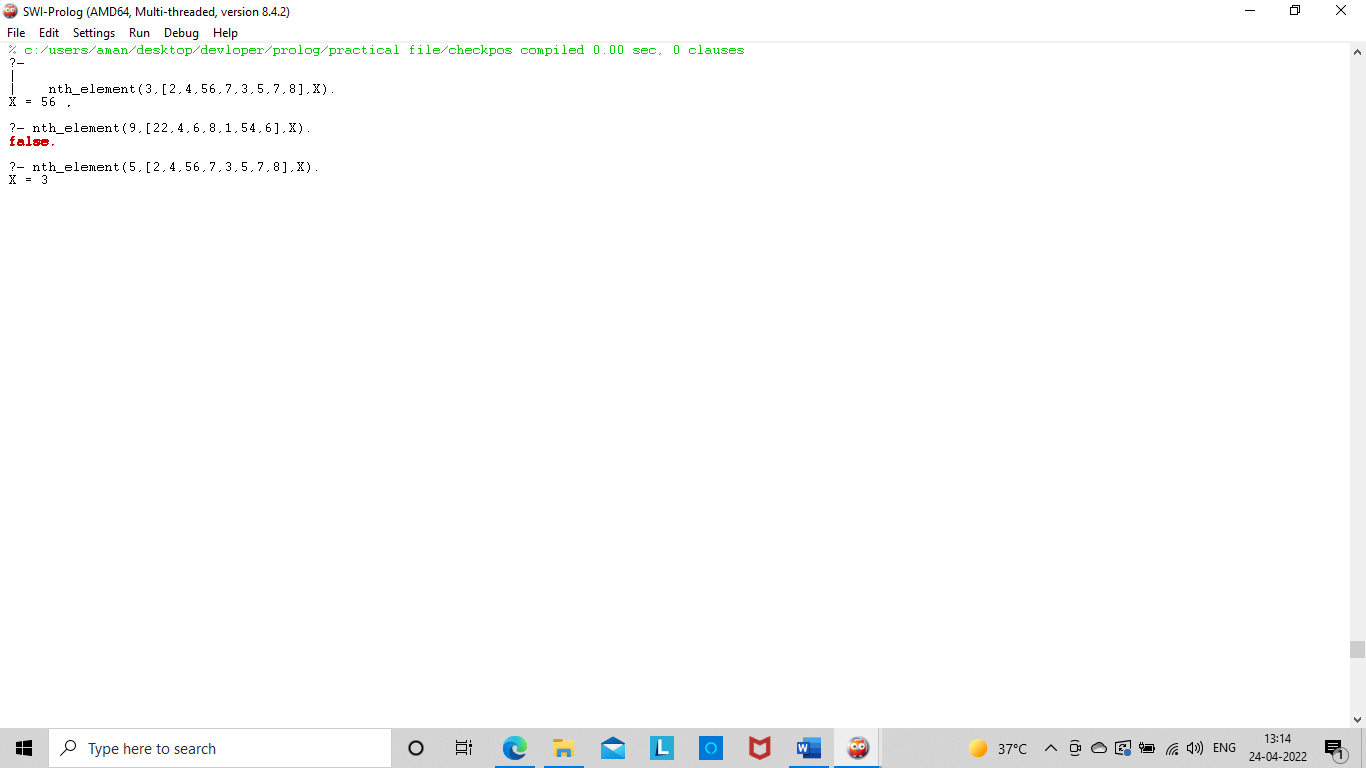
Knowledge Base:

**nth\_element(1,[H|T],H).**

**nth\_element(N,[H|T],X):- N1 is N-1,**

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

Output:



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

Knowledge Base:

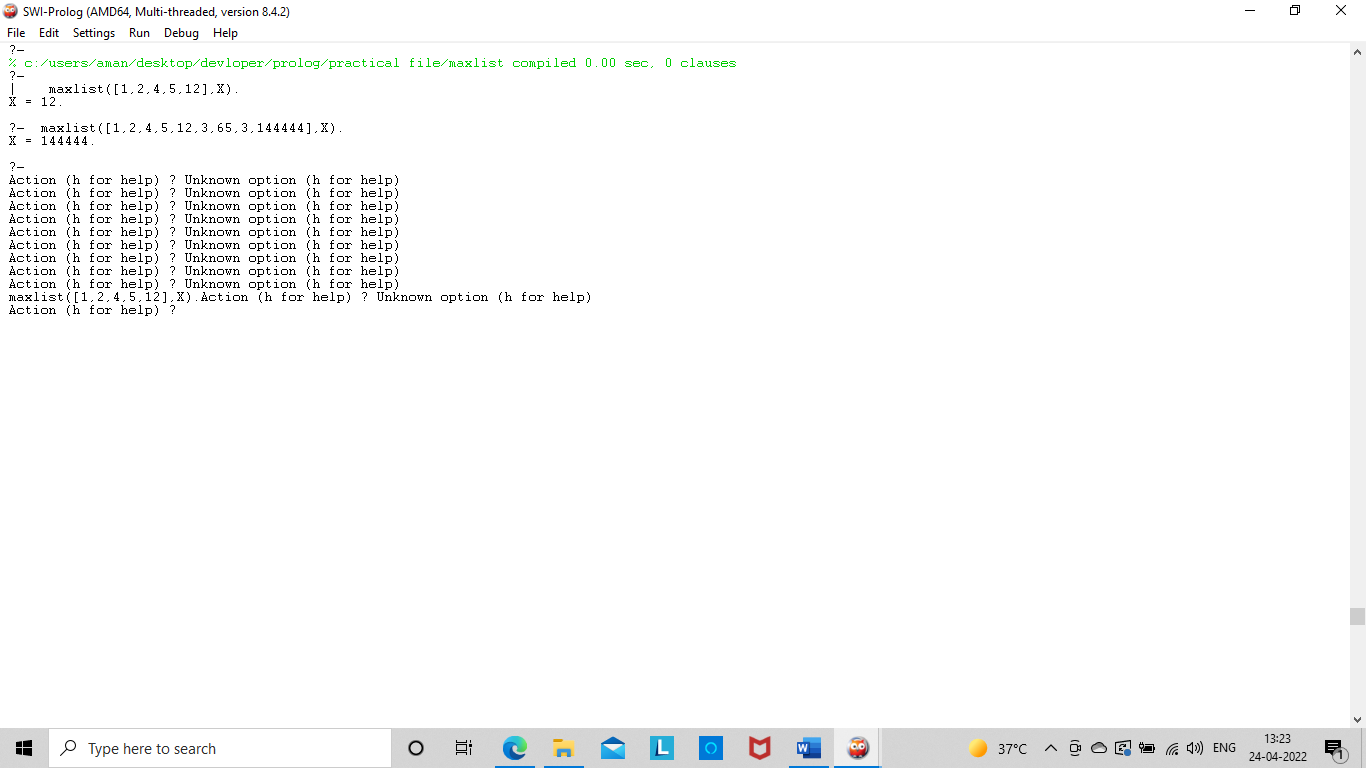
**maxlist([H],H).**

**maxlist([H|T],M):- maxlist(T,M1),**

**H M is M1;**

**M is H.**

Output:



**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.

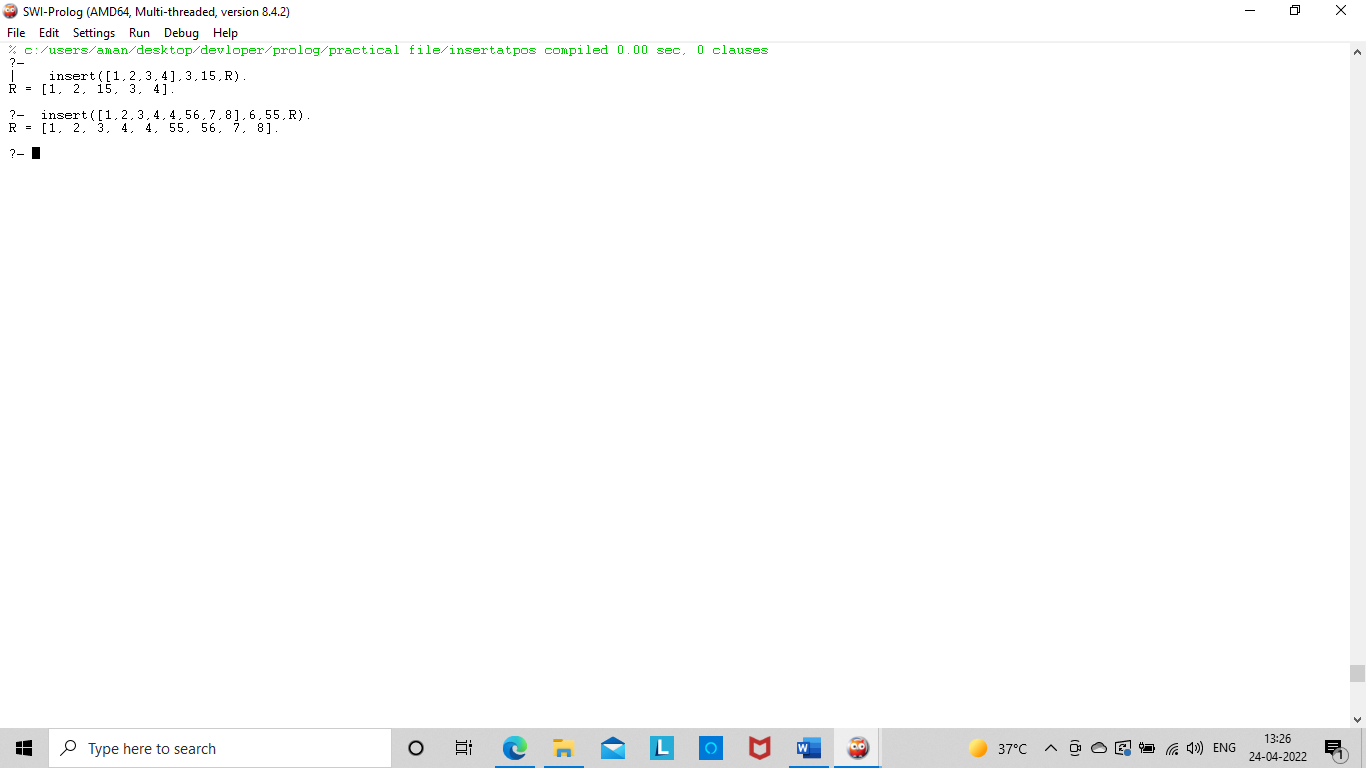
Knowledge Base:

**insert(L,1,Elem,[Elem|L]):-!. insert([],\_,Elem,[Elem]).**

**insert([H|T],N,Elem,[H|R]):- C is N-1,**

**insert(T,C,Elem,R).**

Output:



**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.

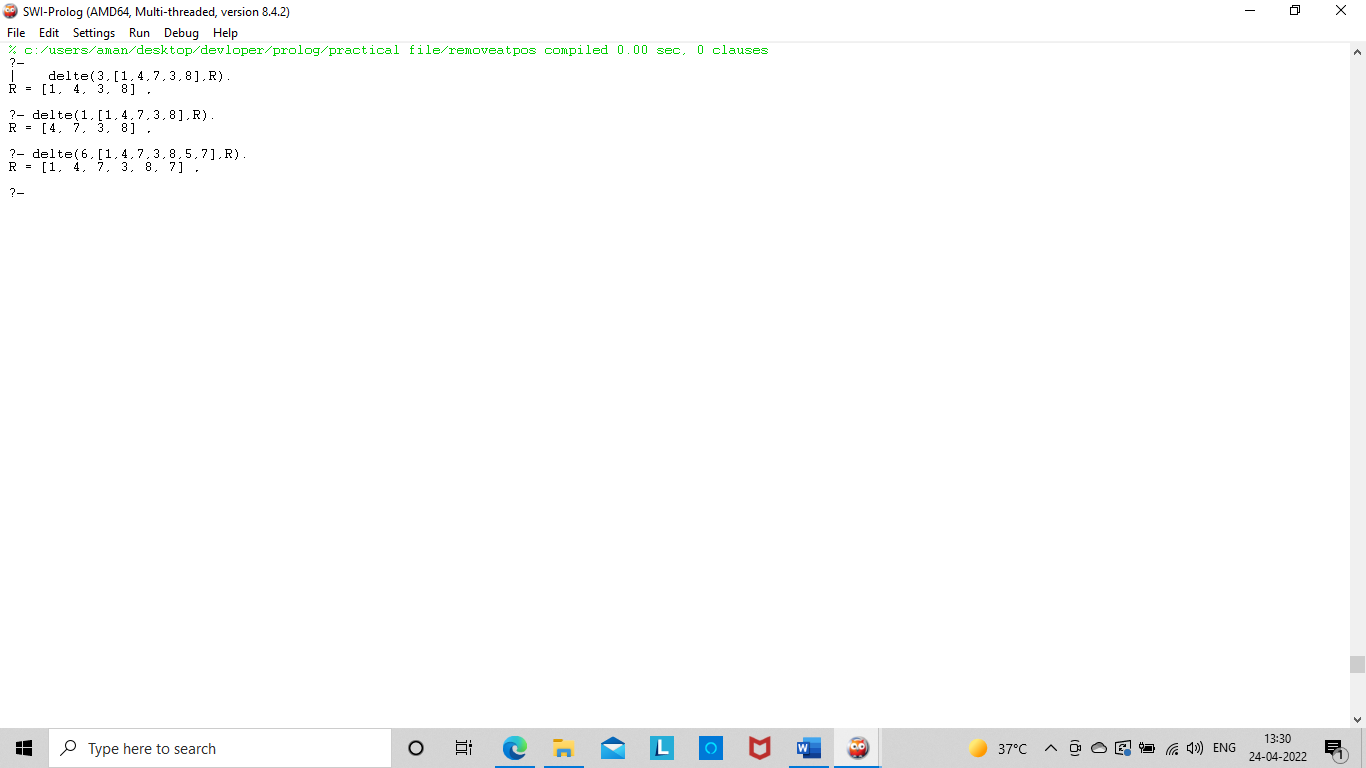
Knowledge Base:

**delte(1,[\_|T],T).**

**delte(P,[X|Y],[X|R]):- P1 is P-1,**

**delte(P1,Y,R).**

Output:



**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.

Knowledge Base:

**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).**

Output:

