**Name : UNNATI BORA**

**College Roll No. : 2019353**

**Examination Roll No. : 19066570046**

**ARTIFICIAL INTELLIGENCE PRACTICAL FILE**

**Q1.) Write a prolog program to calculate the sum of two numbers.**

**Code**

**start:-**

**write("Enter the first number : "),**

**read(A),**

**write("Enter the second number : "),**

**read(B),**

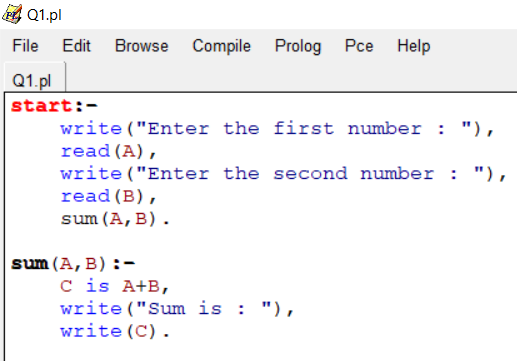
**sum(A,B).**

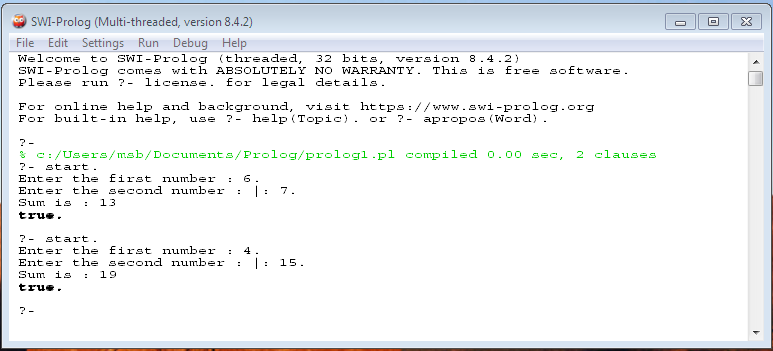
**sum(A,B):-**

**C is A+B,**

**write("Sum is : "),**

**write(C).**





**Q2.) Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.**

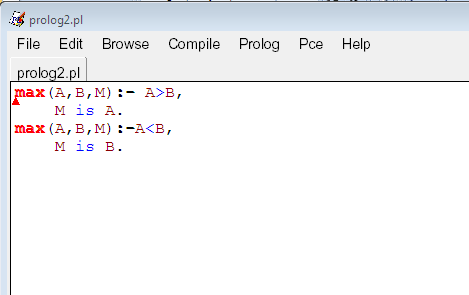
**Code**

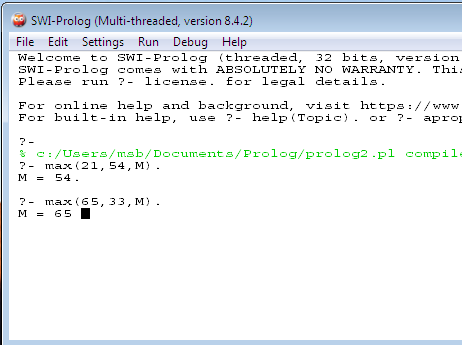
**max(A,B,M):- A>B,**

**M is A.**

**max(A,B,M):-A<B,**

**M is B.**





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

**Code**

**factorial(0,1).**

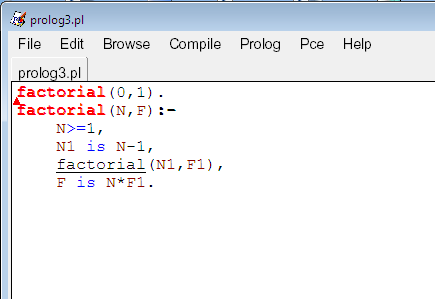
**factorial(N,F):-**

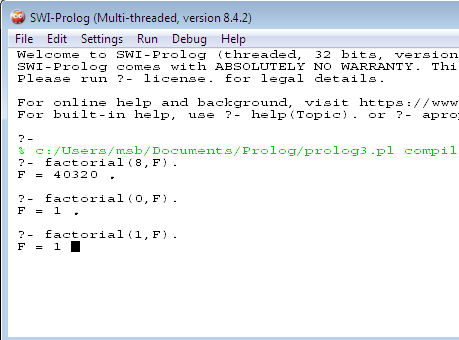
**N>=1,**

**N1 is N-1,**

**factorial(N1,F1),**

**F is N\*F1.**





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

**Code**

**fibonacci(1,0).**

**fibonacci(2,1).**

**fibonacci(N,X):-**

**N>2,**

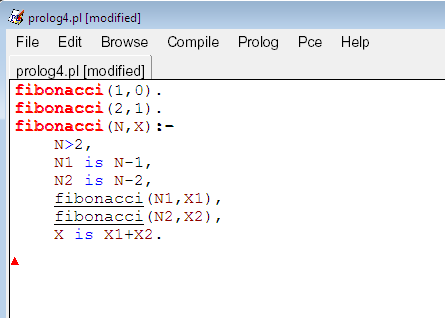
**N1 is N-1,**

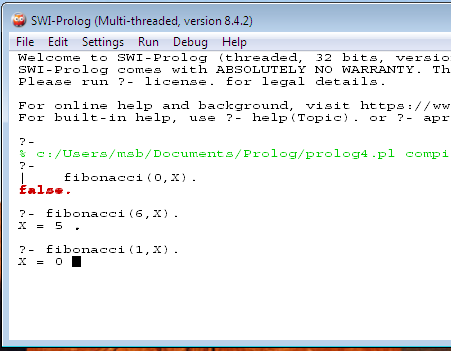
**N2 is N-2,**

**fibonacci(N1,X1),**

**fibonacci(N2,X2),**

**X is X1+X2.**





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

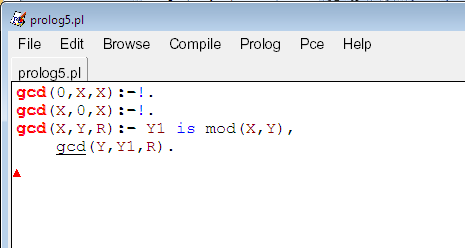
**Code**

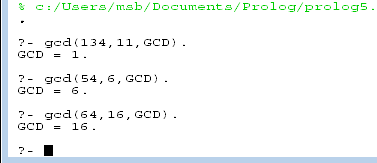
**gcd(0,X,X):-!.**

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

**gcd(X,Y,R):- Y1 is mod(X,Y),**

**gcd(Y,Y1,R).**



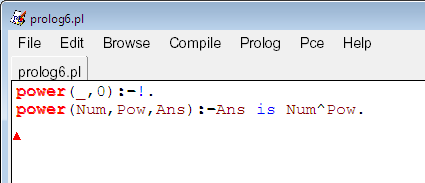


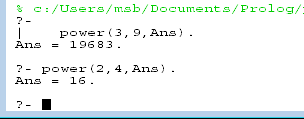
**6. Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.**

**Code**

**power(\_,0):-!.**

**power(Num,Pow,Ans):-Ans is Num^Pow.**

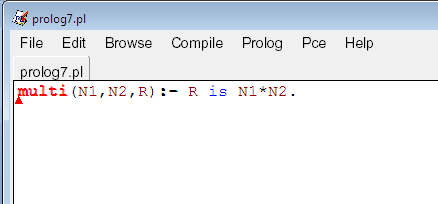


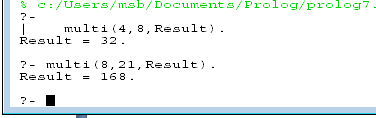


**7. Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.**

**Code**

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





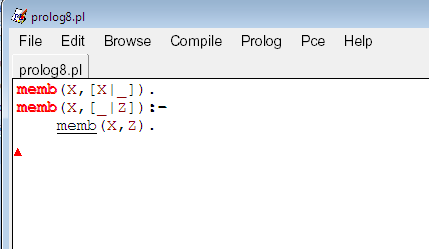
**8. Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.**

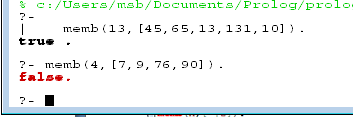
**Code**

**memb(X,[X|\_]).**

**memb(X,[\_|Z]):-**

**memb(X,Z).**



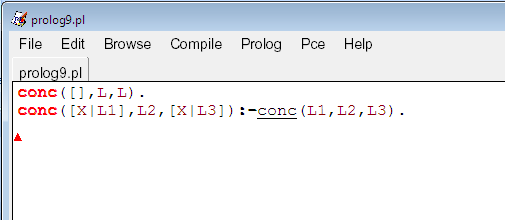


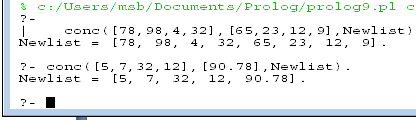
**9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.**

**Code**

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

**conc([X|L1],L2,[X|L3]):-conc(L1,L2,L3).**





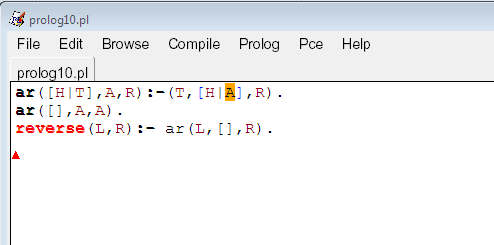
**10. Write a Prolog program to implement reverse (L, R) where List L is original and List R is reversed list.**

**Code**

**ar([H|T],A,R):-(T,[H|A],R).**

**ar([],A,A).**

**reverse(L,R):- ar(L,[],R).**



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

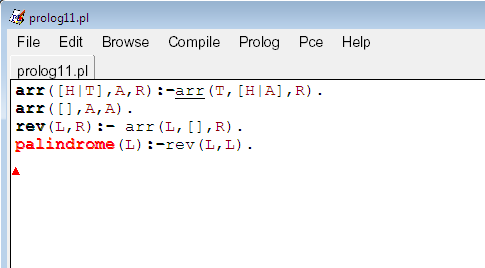
**Code**

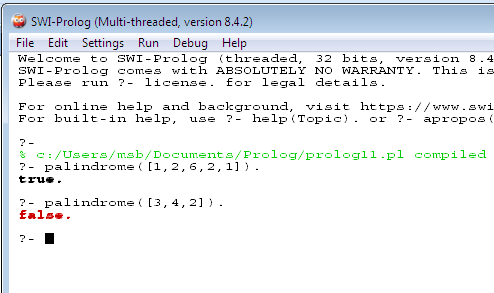
**arr([H|T],A,R):-arr(T,[H|A],R).**

**arr([],A,A).**

**rev(L,R):- arr(L,[],R).**

**palindrome(L):-rev(L,L).**





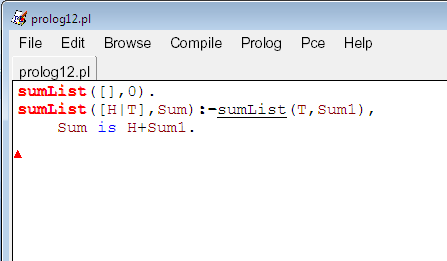
**12. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.**

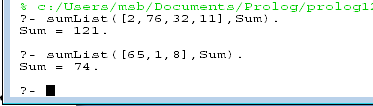
**Code**

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

**sumList([H|T],Sum):-sumList(T,Sum1),**

**Sum is H+Sum1.**





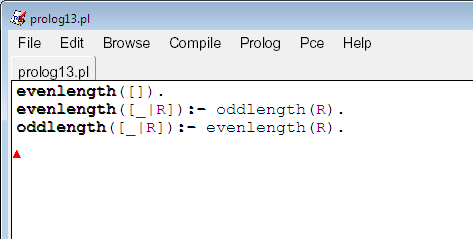
**13. Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.**

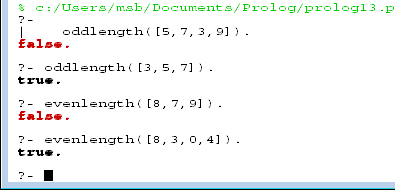
**Code:**

**evenlength([]).**

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

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





**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.**

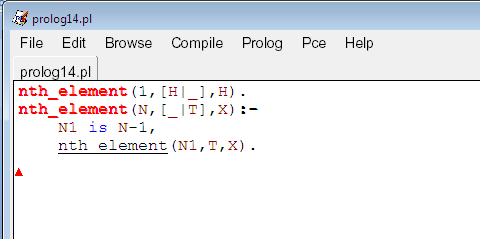
**Code**

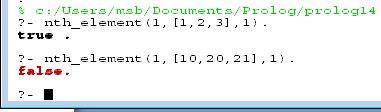
**nth\_element(1,[H|\_],H).**

**nth\_element(N,[\_|T],X):-**

**N1 is N-1,**

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



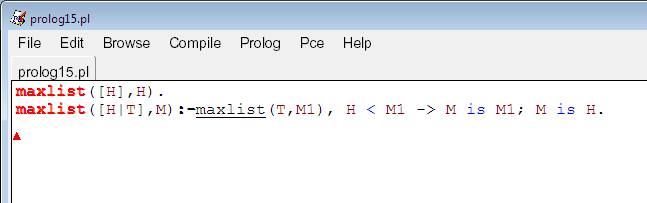


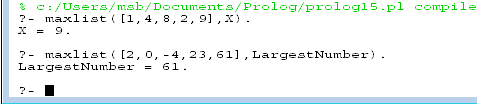
**15. Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.**

**Code**

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

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





**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.**

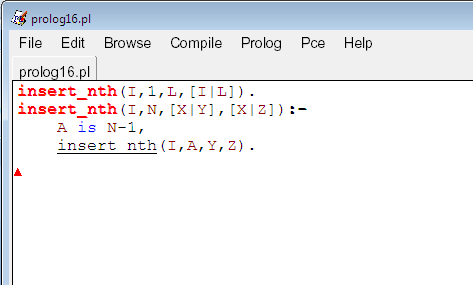
**Code**

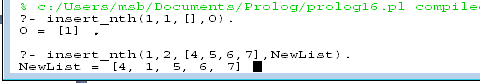
**insert\_nth(I,1,L,[I|L]).**

**insert\_nth(I,N,[X|Y],[X|Z]):-**

**A is N-1,**

**insert\_nth(I,A,Y,Z).**





**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**

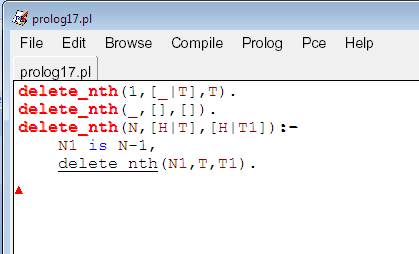
**delete\_nth(1,[\_|T],T).**

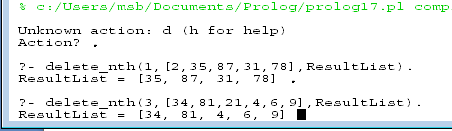
**delete\_nth(\_,[],[]).**

**delete\_nth(N,[H|T],[H|T1]):-**

**N1 is N-1,**

**delete\_nth(N1,T,T1).**





**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**

**merge\_list([],[],[]).**

**merge\_list([],L2,L2).**

**merge\_list(L1,[],L1).**

**merge\_list([H1|T1],[H2|T2],[H1|T3]):-**

**H1=<H2,**

**merge\_list(T1,[H2|T2],T3).**

**merge\_list([H1|T1],[H2|T2],[H2|T3]):-**

**merge\_list([H1|T1],T2,T3).**

