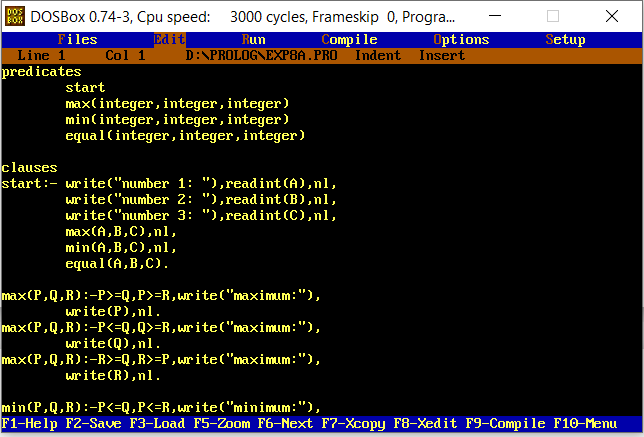
# Lab-8 WAP to study Using logical and arithmetic operators,strings in Prolog.

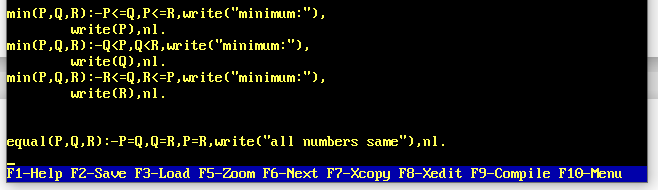
Procedure:-

Write a predicate max(num1,num2,num3) which finds and displays maximum number from three given numbers and min(num1,num2,num3) which finds and displays minimum number of three given numbers ,use logical operators.

Example : Output Enter three numbers : 1 2 3 maximum is “3” , minimum is “1”

**Code:**





**predicates**

**start**

**max(integer,integer,integer)**

**min(integer,integer,integer)**

**equal(integer,integer,integer)**

**clauses**

**start:-write("number 1: "),readint(A),nl,**

**write("number 2: "),readint(B),nl,**

**write("number 3: "),readint(C),nl,**

**max(A,B,C),nl,**

**min(A,B,C),nl,**

**equal(A,B,C).**

**max(P,Q,R):-P>=Q,P>=R,write("maximum:"),**

**write(P),nl.**

**max(P,Q,R):-P<=Q,Q>=R,write("maximum:"),**

**write(Q),nl.**

**max(P,Q,R):-R>=Q,R>=P,write("maximum:"),**

**write(R),nl.**

**min(P,Q,R):-P<=Q,P<=R,write("minimum:"),**

**write(P),nl.**

**min(P,Q,R):-Q<P,Q<R,write("minimum:"),**

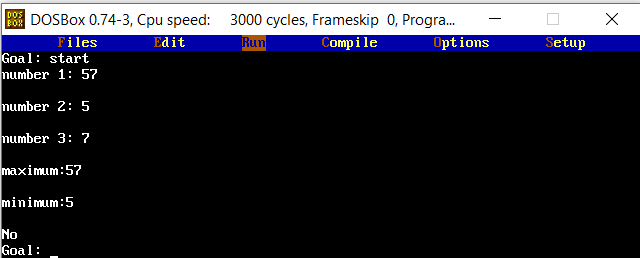
**write(Q),nl.**

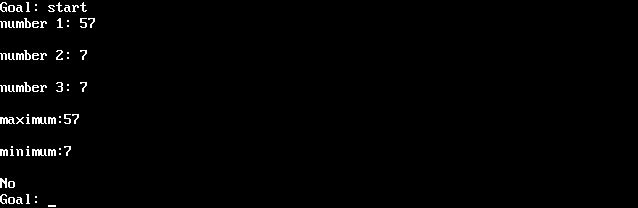
**min(P,Q,R):-R<=Q,R<=P,write("minimum:"),**

**write(R),nl.**

**equal(P,Q,R):-P=Q,Q=R,P=R,write("all numbers same"),nl.**

**Output:**





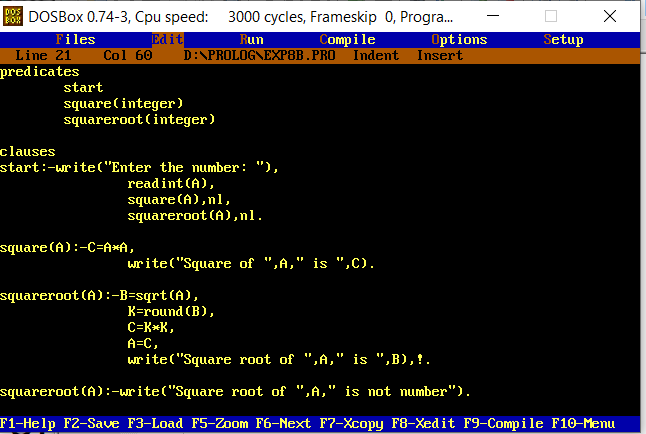


Write a predicate which accepts integer number as an input and displays its square .It should also find its positive square root value ,if its sqrt is integer, otherwise display “NA‟ .Use arithmetic operators /in-built conversion predicates to achieve this.

Example1 :- Output:- Enter no. : 3 3(number) , 9 (square) ,NA(square root not possible)

Example1 :- Output:- Enter no. 4 4 (number), 16 (square) , 2(square root)

**Code:**



**predicates**

**start**

**square(integer)**

**squareroot(integer)**

**clauses**

**start:-write("Enter the number: "),**

**readint(A),**

**square(A),nl,**

**squareroot(A),nl.**

**square(A):-C=A\*A,**

**write("Square of ",A," is ",C).**

**squareroot(A):-B=sqrt(A),**

**K=round(B),**

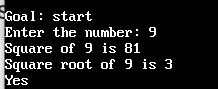
**C=K\*K,**

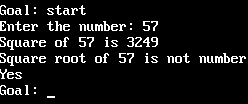
**A=C,**

**write("Square root of ",A," is ",B),!.**

**squareroot(A):-write("Square root of ",A," is not number").**

**Output:**

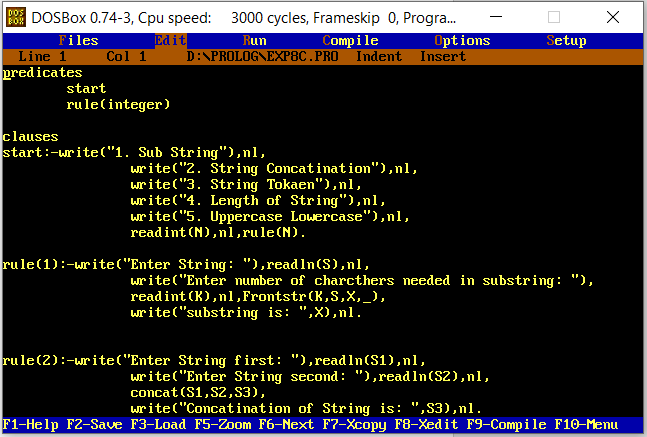


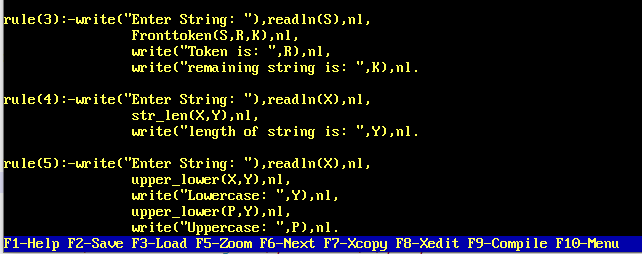


Write a program to find substring from a given string. The substring should start from 1st location of source string and should contain the entered number of characters from the source string.

Example:- Output:-Enter source string: “tested” Enter number of charcthers needed in substring: “4” Original String is : “tested” Substring is “test”

**Code:**





**predicates**

**start**

**rule(integer)**

**clauses**

**start:-write("1. Sub String"),nl,**

**write("2. String Concatination"),nl,**

**write("3. String Tokaen"),nl,**

**write("4. Length of String"),nl,**

**write("5. Uppercase Lowercase"),nl,**

**readint(N),nl,rule(N).**

**rule(1):-write("Enter String: "),readln(S),nl,**

**write("Enter number of charcthers needed in substring: "),**

**readint(K),nl,Frontstr(K,S,X,\_),**

**write("substring is: ",X),nl.**

**rule(2):-write("Enter String first: "),readln(S1),nl,**

**write("Enter String second: "),readln(S2),nl,**

**concat(S1,S2,S3),**

**write("Concatination of String is: ",S3),nl.**

**rule(3):-write("Enter String: "),readln(S),nl,**

**Fronttoken(S,R,K),nl,**

**write("Token is: ",R),nl,**

**write("remaining string is: ",K),nl.**

**rule(4):-write("Enter String: "),readln(X),nl,**

**str\_len(X,Y),nl,**

**write("length of string is: ",Y),nl.**

**rule(5):-write("Enter String: "),readln(X),nl,**

**upper\_lower(X,Y),nl,**

**write("Lowercase: ",Y),nl,**

**upper\_lower(P,Y),nl,**

**write("Uppercase: ",P),nl.**

**Output:**

