**MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES**



**ARTIFICIAL INTELLIGENCE LAB FILE**

**SUBMITTED TO: SUBMITTED BY:**

**DR. POONAM NANDAL SAPNA SINHA**

**DEPT. Of CSE 1/18/FET/BCS/079**

**INDEX**

|  |  |  |
| --- | --- | --- |
| S No. | Practical | Date |
| 1 | Write the steps for installation of prolog and also write the features of prolog. |  |
| 2 | Define the following: predicates, domains, facts, rules, clauses, and goal. |  |
| 3 | WAP to find addition, multiplication, division and subtraction of two numbers. |  |
| 4 | WAP for medical symptom example. |  |
| 5 | WAP to find the brand, color, price, mileage of at least 5 cars. |  |
| 6 | WAP to find the maximum and minimum between two numbers. |  |
| 7 | WAP to find the factorial of a number. |  |
| 8 | WAP to append a list. |  |
| 9 | WAP to reverse a list. |  |
| 10 | WAP to delete a particular number from the list. |  |
| 11 | WAP to find the particular number is a member of the list. |  |
| 12 | WAP to find the number and its position in a list. |  |
| 13 | Writea program to find the length of the list. |  |
| 14 | Write a program to replace an element from a list. |  |
| 15 | WAP to replace every occurrence from the list. |  |
| 16 | Write a program to demonstrate the cut predicate. |  |
| 17 | WAP how to read the integer variable in prolog. |  |
| 18 | Write a program to divide the list into head and tail. |  |
| 19 | Write a program to print all the permutations of a given list. |  |
| 20 | Write a program to print the last element of the list. |  |
| 21 | WAP to print two separate list i.e. positive elements list and negative elements list, from a given list. |  |
| 22 | Write a program to print sublist from a list. |  |
| 23 | Write a program to merge two ordered list. |  |
| 24 | WAP to implement QuickSort. |  |
| 25 | WAP to implement BubbleSort. |  |

**Program No: 01**

**Aim:** Write the steps for installation of prolog and also write the features of prolog.

**Theory:**

**Step 1:** Install DosBox from here:

<http://www.dosbox.com/download.php?main=1>

install it.

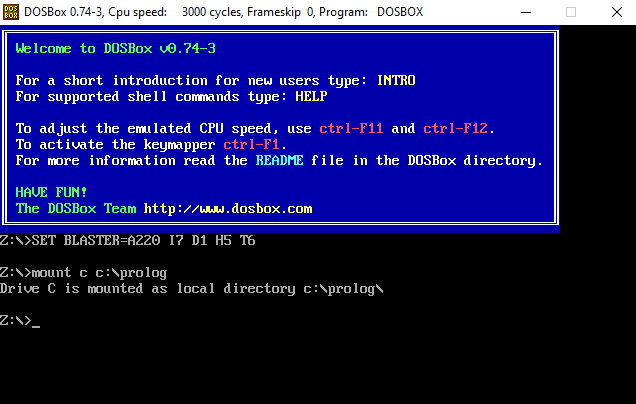
**Step 2:** Now download and copy – paste this file into your C drive.

prolog

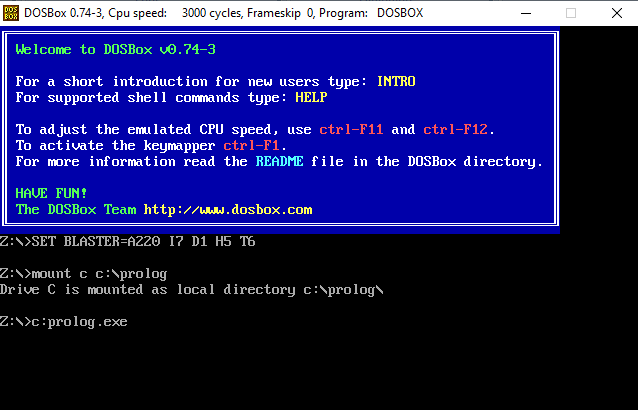
**Step 3:** Open DosBox.

Write – mount c c:\prolog

then enter.

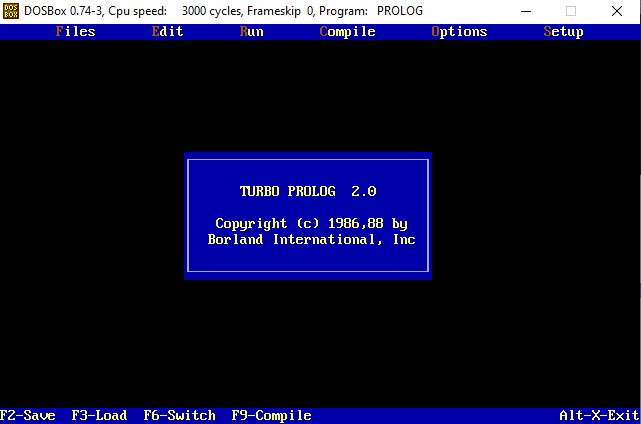


Then write c:prolog.exe



Then enter

You will see a window like this



**Program No: 02**

**Aim:**  Define following: predicates, domains, facts, rules, clauses, goal.

**Theory:**

**(a) Predicates:**

A predicate is a function that tests for some condition involving its arguments and returns nil if the condition is false, or some non-nil value if the condition is true.

**(b) Domains:**

Domains enable you to give distinctive names to different kinds of data that would otherwise look alike. In a Visual Prolog program, objects in a relation (the arguments to a predicate) belong to domains; these can be pre-defined domains, or special domains that you specify.

**(c) Facts:**

 A fact is a predicate expression that makes a declarative statement about the problem domain.

**(d) Rules:**

A Prolog program consists of a number of clauses. Each clause is either a fact or a rule.

**(e) Clauses:**

A program, or database, in Prolog consists of one or more predicates; each predicate consists of one or more clauses.

**(f) Goal:**

A goal is something that Prolog tries to satisfy by finding values of the variables that make the goal succeed.

**Program No: 03**

**Aim:** Write a program to find addition, multiplication, division and subtraction of two numbers.

**Source Code:**

Domains

A, B, C = integer

predicates

sum (A, B, C)

subtract(A, B ,C)

multiply(A, B, C)

division (A, B, C)

clauses

sum(A,B,C):-C=A+B.

subtract(A,B,C):-C=A-B.

multiply(A,B,C):-C=A\*B.

divison(A,B,C):-C=A/B.

**Output:**



**Program No: 04**

**Aim:** Write a program for medical symptom example.

**Source Code:**

domains

D=symbol

S1, S2= symbol

Predicates

disease (D, S1 ,S2)

Clauses

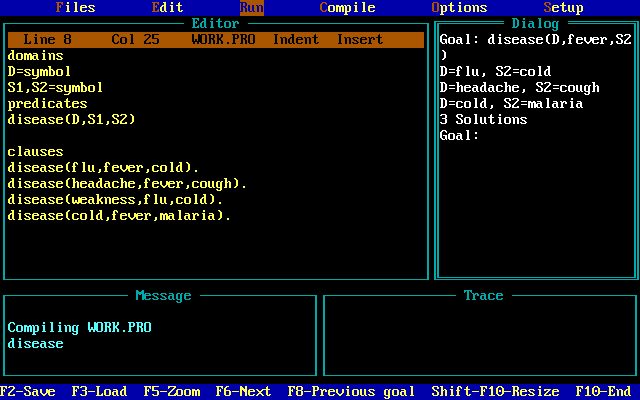
disease (flu , fever ,cold).

disease (headache , fever , cough).

disease (weakness , flu , cold).

disease (cold, fever, malaria).

**Output:**

****

**Program No: 05**

**Aim:** Write a program to find the brand, color, price, mileage of at least 5 cars.

**Source Code:**

Domains

Name, brand, color, price = symbol

Mileage = integer

Predicates

Car(name, color, brand , price , mileage)

Clauses

Car(a3, black , audi , tweenty\_lacs , 60).

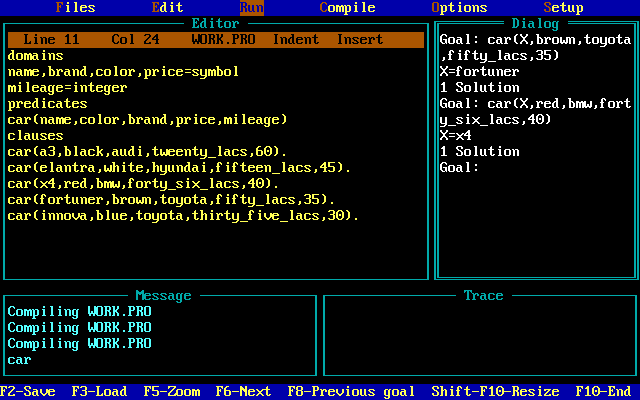
Car (elantra, white, Hyundai, fifteen\_lacs, 45).

Car(X4, red, bmw, forty\_six\_lacs, 40).

Car(fortuner ,brown , Toyota , fifty\_lacs, 35).

Car (innova, blue, Toyota, thirty\_five\_lacs, 30).

**Output:**

****

**Program No: 06**

**Aim:** Write a program to find the maximum and minimum between two numbers.

**Source Code:**

Domains

X, Y = integer

Predicates

Max (integer, integer, integer)

Min (integer, integer, integer)

Clauses

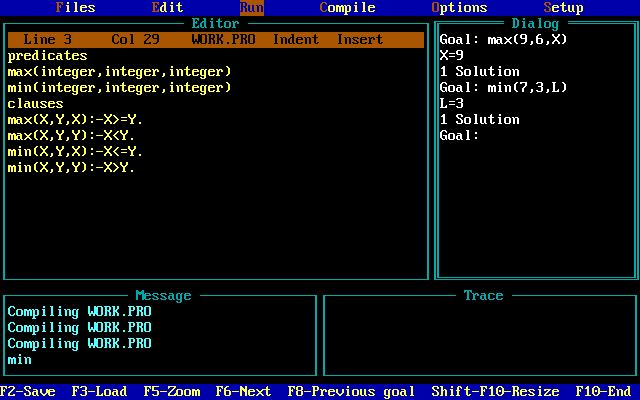
Max (X, Y, X):- X>= Y.

Max (X, Y, Y):- X< Y.

Min (X, Y, X):- X<= Y.

Min (X, Y, Y):- X> Y.

**Output:**

****

**Program No: 07**

**Aim:** Write a program to find the factorial of a number.

**Source Code:**

Domains

X, Y = integer

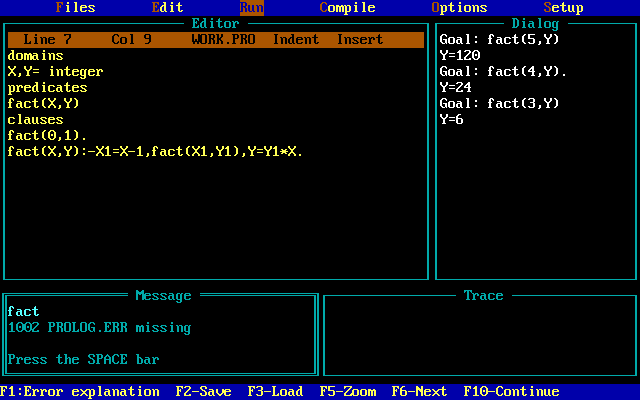
Predicates

Fact(X, Y)

Clauses

Fact (0, 1).

Fact(X, Y):-X1=X-1, fact (X1,Y1), Y= y1\*X.

**Output: **

**Program No: 08**

**Aim:** Write a program to append a list.

**Source Code:**

Domains

List = integer\*

Predicates

Append (list, list, list)

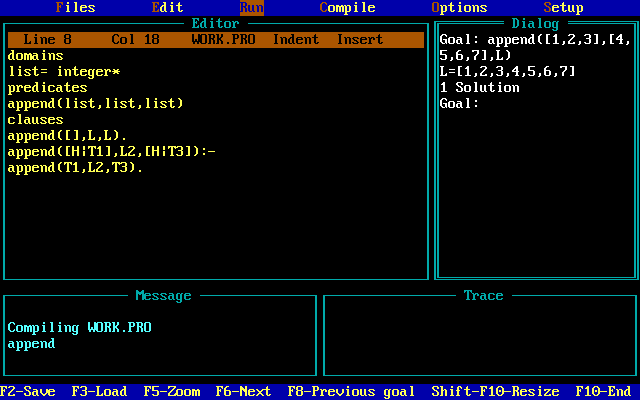
Clauses

Append ([], L, L).

Append ([H|T1], L2, [H|T3]):-

Append (T1, L2, T3).

**Output:**

****

**Program No: 09**

**Aim:** Write a program to reverse a list.

**Source Code:**

Domains

List= integer\*

Predicates

Reverse\_list (list, list)

Reverse (list, list, list)

Clauses

Reverse\_list (Inputlist, Outputlist):-

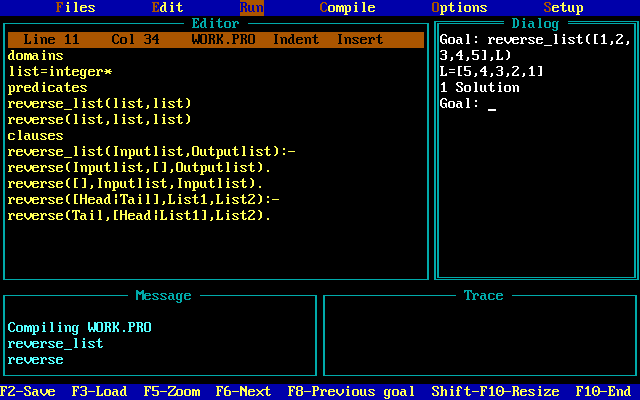
Reverse (Inputlist, [], Outputlist).

Reverse ([], Inputlist, Inputlist).

Reverse ([Head|Tail], List1, List2):-

Reverse (Tail, [Head|List1], List2).

**Output:**

****

**Program No: 10**

**Aim:** Write a program to delete a particular number from the list.

**Source code:**

Domains

X= integer

List= integer\*

Predicates

Delete(X, list, list)

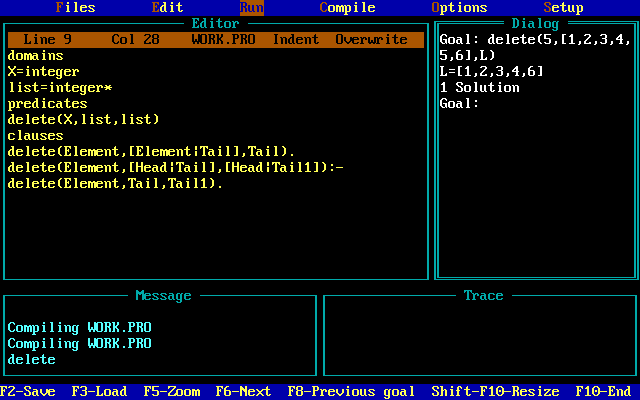
Clauses

Delete (Element, [Element|Tail], Tail).

Delete (Element, [Head|Tail], [Head|Tail1]):-

Delete (Element, Tail, Tail1).

**Output:**

****

**Program No: 11**

**Aim:** Write a program to find the particular number is a member of the list.

**Source Code:**

Domains

List= integer\*

Predicates

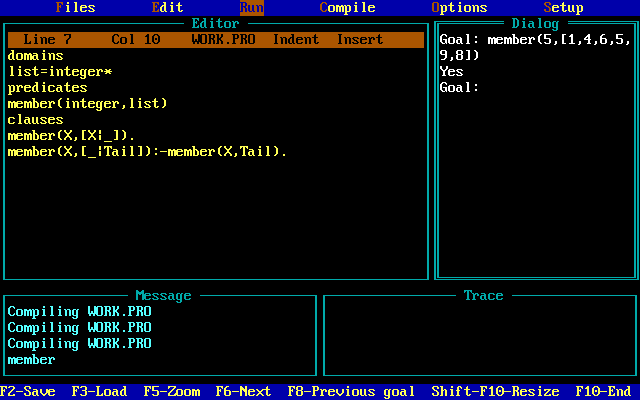
Member (integer, list)

Clauses

Member (X, [X|\_]).

Member (X, [\_|Tail1]):- member (X, Tail).

**Output:**

****

**Program No: 12**

**Aim:** Write a program to find the number and its position in a list.

**Source Code:**

Domains

List= integer\*

N, P= integer

Predicates

Member (list, N, P)

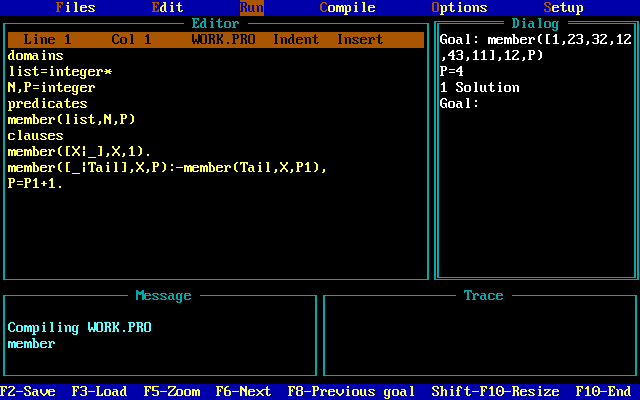
Clauses

Member ([X|\_], X, 1).

Member ([\_|Tail], X, P):- member (Tail, X, P1),

P= P1+1.

**Output:**

****

**Program No: 13**

**Aim:** Writea program to find the length of the list.

**Source Code:**

Domains

List= integer\*

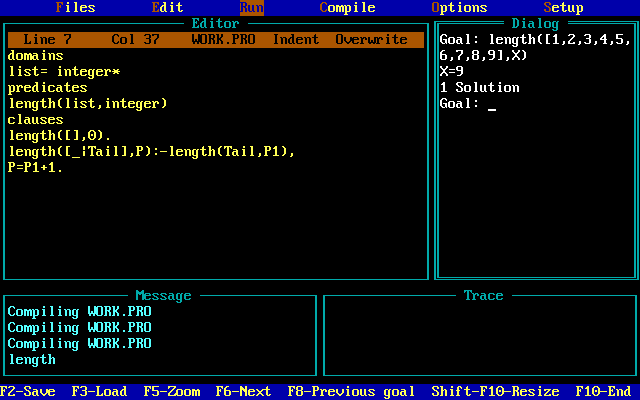
Predicates

Length ([], 0).

Length ([\_|Tail], P):- length (Tail, P1),

P= P1+1.

**Output:**

****

**Program No: 14**

**Aim:** Write a program to replace an element from a list.

**Source Code:**

Domains

List=integer\*

Predicates

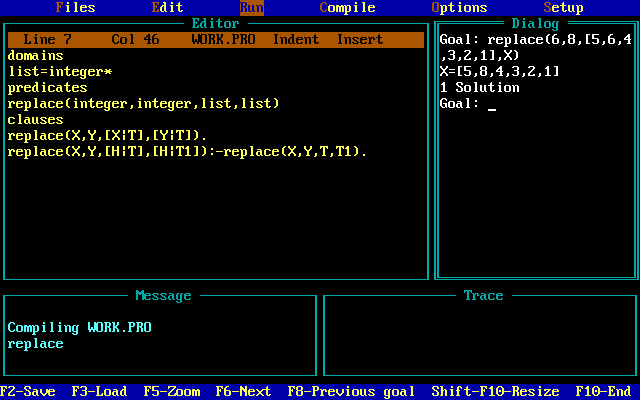
Replace ( integer , integer, list, list)

Clauses

Replace ( X, Y, [X|T] , [Y|T]).

Replace ( X, Y, [H|T], [H|T1]):- replace( X , Y, T, T1).

**Output:**

****

**Program No: 15**

**Aim:** Write a program to replace every occurrence from the list.

**Source Code:**

Domains

List=integer\*

Predicates

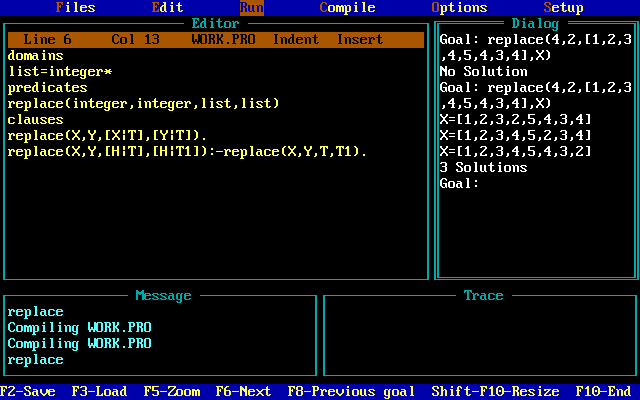
Replace ( integer , integer, list, list)

Clauses

Replace ( X, Y, [X|T] , [Y|T]).

Replace ( X, Y, [H|T], [H|T1]):- replace( X , Y, T, T1).

**Output:**

****

**Program No: 16**

**Aim:** Write a program to demonstrate the cut predicate.

**Source Code:**

Predicates

Num (integer, integer)

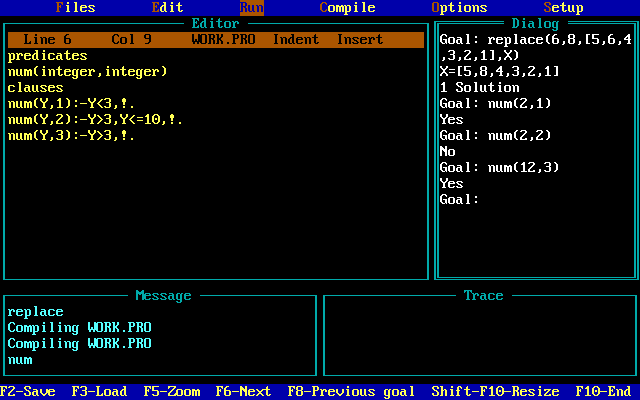
Clauses

Num(Y, 1):- Y< 3 , !.

Num(Y, 2):- Y> 3, Y<= 10 , !.

Num(Y, 3):- Y> 3 , ! .

**Output:**

****

**Program No: 17**

**Aim:** Write a program how to read the integer variable in prolog.

**Source Code:**

Predicates

Go

Clauses

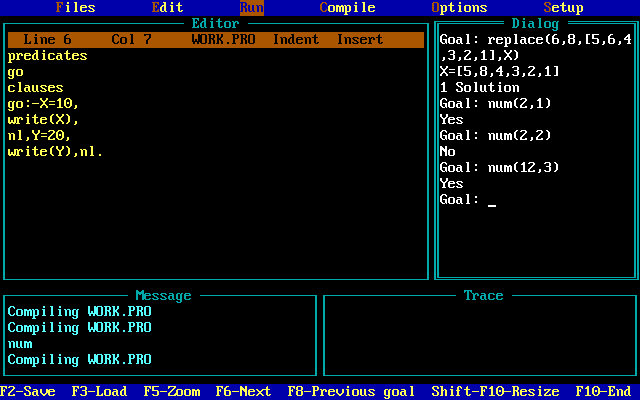
Go :- X=10

Write(X)

Nl , Y= 20

Write(Y) , nl.

**Output:**

****

**Program No: 18**

**Aim:** Write a program to divide the list into head and tail.

**Source Code:**

Domains

List=symbol\*

Predicates

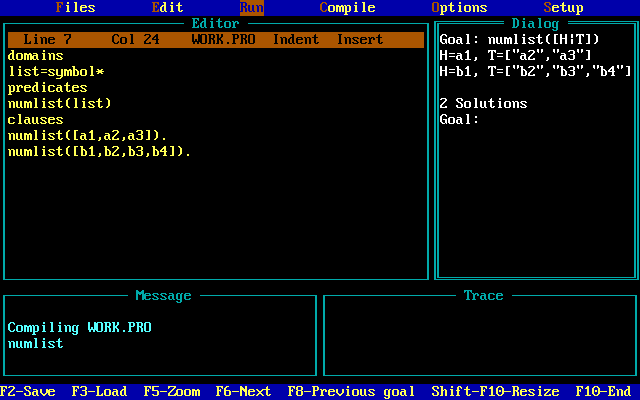
Numlist (list)

Clauses

Numlist ([a1, a2, a3]).

Numlist ([b1, b2, b3, b4]).

**Output:**

****

**Program No: 19**

**Aim:** Write a program to print all the permutations of a given list.

**Source Code:**

domains

list=symbol\*

predicates

permute(list,list)

del(symbol,list,list)

clauses

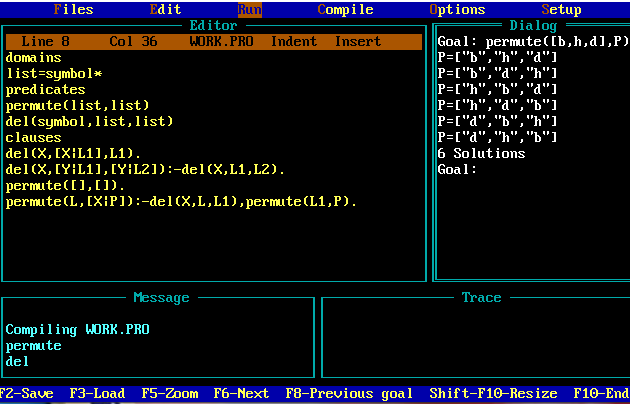
del(X,[X|L1],L1).

del(X,[Y|L1],[Y|L2]):- del(x,L1,L2).

Permute([],[]).

Permute(L,[X|P1]):-del(X,L,L1),permute(L1,P).

**Output:**

****

**Program No: 20**

**Aim:** Write a program to print the last element of the list.

**Source Code:**

domains

list=symbol\*

predicates

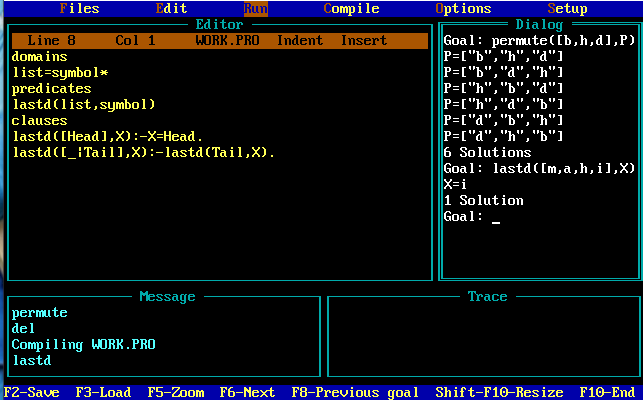
lastd(list,symbol)

clauses

lastd([Head],X):-X=Head.

Lastd([\_Tail],X):-lastd(Tail,X).

**Output:**



**Program No: 21**

**Aim:** Write a program to print two separate list i.e. positive elements list and negative elements list, from a given list.

**Source Code:**

domains

list=integer\*

predicates

separate(list,list,list)

clauses

separate([],[],[]).

Separate([X|L],[X|L1],L2):-

X>=0.

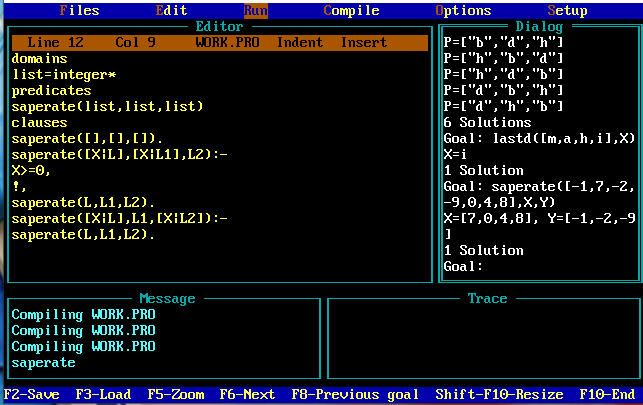
!.

Separate(L,L1,L2).

Separate([X|L],L1,[X|L2]):-

Separate(L,L1,L2).

**Output:**

****

**Program No: 22**

**Aim:** Write a program to print sublist from a list.

**Source Code:**

List=integer\*

Predicates

sublist(list,list)

Clauses

sublist([],[]).

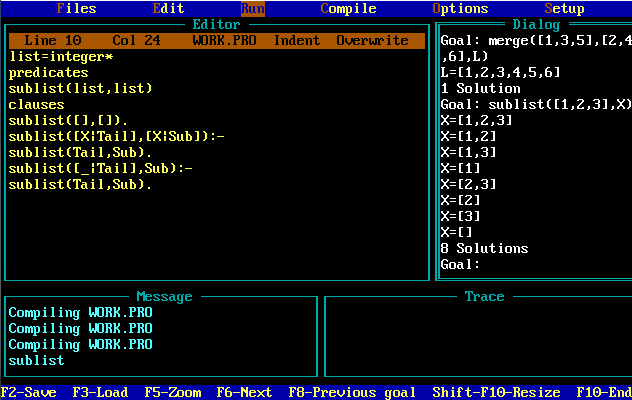
sublist([X|Tail],[X|sub]):-

sublist(Tail,sub).

ublist([\_|Tail],sub):-

sublist(Tail,sub).

**Output:**

****

**Program No: 23**

**Aim:** Write a program to merge two ordered list.

**Source Code:**

domains

X=integer

list=integer\*

predicates

merge(list,list,list)

clauses

merge([],[],[]).

Merge([X],[],[X]).

Merge([],[Y],[Y]).

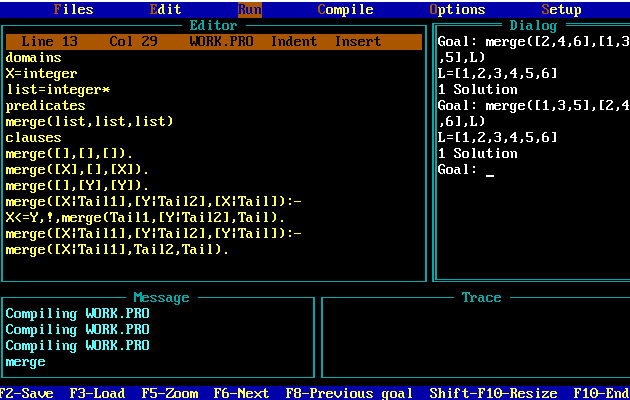
Merge([X|Tail1],[Y|Tail2],[X|Tail]):-

X<Y,!,merge(Tail1,[Y|Tail2],Tail).

Merge([X|Tail1],[Y|Tail2],[Y|Tail]):-

Merge([X|Tail1],Tail2,Tail).

**Output:**

****

**Program No: 24**

**Aim:** Write a program to implement Quick Sort.

**Source Code:**

Domains

list = integer\*.

Predicates

quicksort(list,list).

split(integer,list,list,list).

concatenate(list,list,list).

printlist(list).

quicksort([],[]).

quicksort([Head|Tail],SortedList) :-

split(Head,Tail,SList,BList),

quicksort(SList,SList1),

quicksort(BList,BList1),

concatenate(SList1,[Head|Blist1],SortedList),

printlist(SortedList).

split(\_,[],[],[]).

split(Item,[Head1|Tail1],[Head1|SList],BList) :-

Item > Head1 , ! ,

split(Item,Tail1,SList,BList).

split(Item,[Head1|Tail1],SList,[Head1|BList]) :-

split(Item,Tail1,SList,BList).

concatenate([],List,List).

concatenate([Item|List1],List2,[Item|List3]) :-

concatenate(List1,List2,List3).

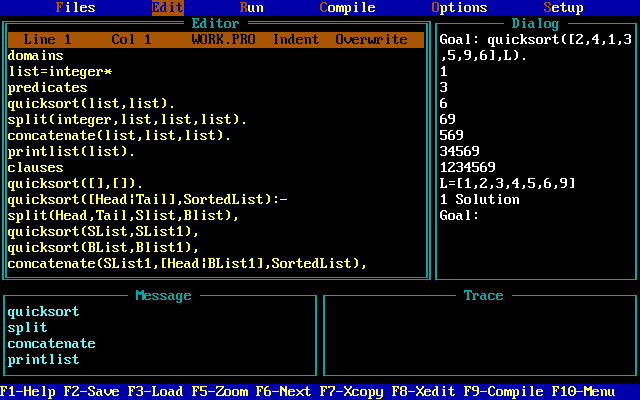
printlist([]) :- nl.

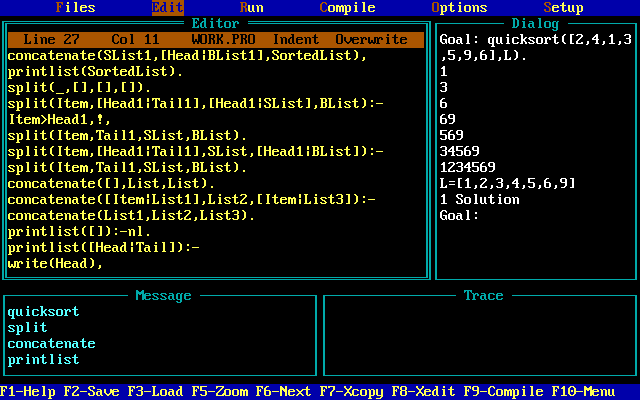
printlist([Head|Tail]) :-

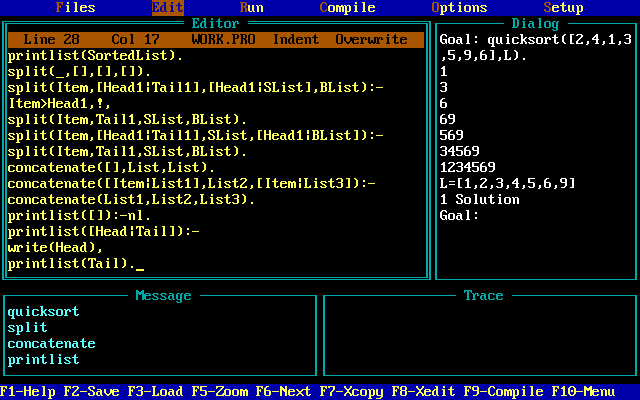
write(Head),

printlist(Tail).

**Output:**

****

****

****

**Program No: 25**

**Aim:** Write a program to implement Bubble Sort.

**Source Code:**

Domains

list = integer\*.

Predicates

bubblesort(list,list).

swap(list,list).

printlist(list).

Clauses

bubblesort(InputList,SortList) :-

swap(InputList,List) , ! ,

printlist(List),

bubblesort(List,SortList).

bubblesort(SortList,SortList).

swap([X,Y|List],[Y,X|List]) :- X > Y.

swap([Z|List],[Z|List1]) :- swap(List,List1).

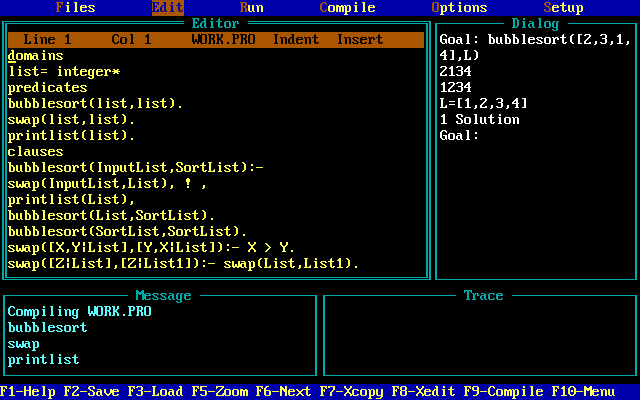
printlist([]) :- nl.

printlist([Head|List]) :-

write(Head),

printlist(List).

**Output:**

****