LAB 5 | Artificial Intelligence

Aim : To study about controlling execution in prolog using cut and fail predicate

Fail:

- Failure can be forced in any rule by using the built-in fail predicate.
- The fail forces backtracking in an attempt to unify with other clauses.

Cut:

- The primary purpose of cut is to prevent or block backtracking based on a specified condition.
- The cut predicate is specified as an exclamation point(!).

1. Implement a prolog program to find minimum and maximum of two integers using cut and/ or fail predicate. Program must have three arguments and it must handle all cases.

Code :

```
predicates
        max(integer,integer,integer).
        min(integer,integer,integer).
clauses
        max(X, Y, Max):-X>=Y,!,Max=X;Max=Y.
        min(X, Y, Min):-X<=Y,!,Min=X;Min=Y.
```

Output:

```
Goal : max(100,200,Max)
Max=200
1 Solution

Goal : min(100,200,Min)
Min=100
1 Solution

Goal : max(100,200,100)
No

Goal : min(100,200,100)
Yes
```

2. Write a prolog program to verify that a given year is leap year or not using cut and/ or fail predicate.
   Note: A year is a leap year if it is divisible by 4, but century years are not leap years unless they are divisible by 400. So, the years 1700, 1800, and 1900 were not leap years, but the year 2000 was.

Code:

```
domains
        year=integer

predicates
        leap_check(year).
        check(year).

clauses
        leap_check(Year):-
                Year mod 4 = 0,
                Year mod 100 = 0,
                Year mod 400 = 0.

        leap_check(Year):-
                Year mod 4 = 0,
                Year mod 100<>0.

        check(Year):-
                Year<0,!,
                write("Year cannot be negative"),nl.

        check(Year):-
                leap_check(Year),!,
                write(Year, " is a leap year."),nl;write(Year, " is not a leap year.").
```

Output:

```
Goal : check(1700)
1700 is not a leap year.
Yes

Goal : check(2000)
2000 is  a leap year.
Yes
```

3. Write a prolog program to verify that a given number is prime or not using cut and/ or fail predicate.

Code:

```
predicates
        isprime(integer)
        check(integer,integer)

clauses
        isprime(0):-!,fail.
        isprime(1):-!,fail.
        isprime(2):-!.
        isprime(X):-check(X,2).

        check(X,Y):-Y>sqrt(X),!.
        check(X,Y):-Y<=sqrt(X),X mod Y=0,!,fail.
        check(X,Y):-YY=Y+1,check(X,YY).
```

Output :

```
Goal : isprime(3)
Yes

Goal : isprime(25)
No
```