EXPERIMENT-8

Aim - To write a clisp program to add/subtract/multiply/divide two numbers.

Code-

```
( setq x(read) )
( setq y(read) )
(format t "x=~d~%" x)
(format t "y=~d~%" y)

(print ( + x y ) )
(print ( - x y ) )
(print ( * x y ) )
(print ( / x y ) )
```

Input –

x=4
y=5

Output-

x=4
y=5
9
-1
20
.8

****

EXPERIMENT-9

Aim - To write a clisp program to check whether number is positive or negative using cond construct.

Code-

```
( setq a(read) )
(cond ( (> a 0)
  (format t "~% a is positive") )
  ( (< a 0)
  (format t "~% a is negative") )
)
```

Input –

1) x=-1
2) x=2

Output-

1) a is negative.
2) a is positive.

****

EXPERIMENT-10

Aim - To write a clisp program to create a function for addition of two numbers.

Code-

```
( defun fi( x y )
        ( format t " x is ~A,y is ~A ~%" x y)
        ( + x y )
)

( setf z( fi 10 5) )
( format t "z is ~A ~%" z)
```

Input –

1) x=50
   y=60
2) x=10
   y=-15

Output-

1) z is 110
2) z is -5

****

EXPERIMENT-11

Aim - To write a clisp program to check whether a number is greater or not using if construct.

Code-

```
(setq a(read))
(setq b(read))
(if (> a b)
  (format t "~% a is greater than b")
  (format t "~% b is greater than a"))
```

Input –

1) a=10
   b=5
2) a=3
   b=7

Output-

1) a is greater than b
2) b is greater than a

****

EXPERIMENT-12

Aim - To write a clisp program to check whether a number is greater or not using when construct.

Code-

```
(setq a(read))
(setq b(read))

(when (> a b)
  (format t "~% a is greater than b"))
```

Input –

a=5
b=2

Output-

a is greater than b

****

EXPERIMENT-13

Aim – Write a program to check whether a number is even or odd using function in Lisp. The function returns 0 if the number is even and 1 if the number is odd.

Code-

```
(defun even(x)
        (mod x 2)
)

(print "Enter number")
(setq x(read))

( if ( = (even x) 0 )
        (print "Number is Even")
        (print "Number is Odd")
)
```

Input –

1) "Enter Number" 7
2) "Enter Number" 8

Output-

1) "Number is Odd"
2) "Number is Even"

****

EXPERIMENT-14

Aim - To write a clisp program to print day by using case construct.

Code-

```
(setq day(read))
(case day
(1 (format t "~% Monday"))
(2 (format t "~% Tuesday"))
(3 (format t "~% Wednesday"))
(4 (format t "~% Thursday"))
(5 (format t "~% Friday"))
(6 (format t "~% Saturday"))
(7 (format t "~% Sunday")))
```

Input –

1) day=7
2) day=2

Output-

1) Sunday
2) Tuesday

****

EXPERIMENT-15

Aim – Write a prolog program to find length of list.

Code-

```
len(X):-
        findlen(X,Count),
        write("\nLength Of List : "),
        write(Count).

    findlen([],X):-
        X=0.

    findlen([X|Tail],Count):-
        findlen(Tail,Prev),
        Count is Prev + 1.
```

Input –

1) len([1,2,3,4,6,7]).

2) len([1,2,'a','b']).

Output-

1) Length Of List : 6
2) Length Of List : 4

****

EXPERIMENT-16

Aim – Write a prolog program to find solution of monkey banana problem.

Code-

```
move(state(middle,onbox,middle,hasnot),grasp,state(middle,onbox,middle,has)).
move(state(P,onfloor,P,hasnot),climb,state(P,onbox,P,hasnot)).
move(state(P,onfloor,P,hasnot),push,state(P1,onfloor,P1,hasnot)).
move(state(P1,onfloor,B,hasnot),walk,state(P2,onfloor,B,hasnot)).

canget(state(_,_,_,has)) :-
    write("get").

canget(State1) :-
    move(State1,Move,State2),
    canget(State2),
    write(State2),nl.
```

Input –

1) canget(state(atdoor, onfloor, atwindow, hasnot)).
2) canget(state(atwindow, onbox, atwindow, hasnot)).
3) canget(state(Monkey, onfloor, atwindow, hasnot)).

Output-

1) getstate(middle,onbox,middle,has)
   state(middle,onbox,middle,hasnot)
   state(middle,onfloor,middle,hasnot)
   state(atwindow,onfloor,atwindow,hasnot)
   true .
2) false.
3) getstate(middle,onbox,middle,has)
   state(middle,onbox,middle,hasnot)
   state(middle,onfloor,middle,hasnot)
   Monkey = atwindow .

****

EXPERIMENT-17

Aim – Write a prolog program to find the solution of tower of Hanoi problem.

Code-

```
move(1,X,Y,_):-write('Move disk from '),write(X),write(' to '),write(Y),nl.
move(N,X,Y,Z):-N>1,M is N-1,
  move(M,X,Z,Y),
  move(1,X,Y,_),
  move(M,Z,Y,X).
```

Input –

```
move(3,'A','B','C').
```

Output-

Move disk from A to B

Move disk from A to C

Move disk from B to C

Move disk from A to B

Move disk from C to A

Move disk from C to B

Move disk from A to B

****