



Meerut Institute of Engineering & Technology

Department of Computer Science & Engineering

DR. A.P.J KALAM TECHNICAL UNIVERSITY
LUCKNOW



LAB MANUAL

B.TECH
COMPUTER SCIENCE AND ENGINEERING
SESSION – 2018-2019

Lab Code: RCS-553
Lab: Principal of Programming Language

Faculty Name:
Ms. Anuradha Taluja (Assistant Professor)
Ms Mariya Khurshid (Assistant Professor)
Ms. Pragya Gaur (Assistant Professor)

Index

- Time Table
- Lab course outcome
- List of programs
- List of value added programs
- Programs

List of Lab Outcomes

LO1: To learn the basics of different types of programming.

LO2: To understand the syntax and building blocks of the Functional Programming Language.

LO3: To learn to solve a problem using the Meta Language (ML).

LO4: To compile and debug a Meta Language (ML) Program.

LO5: To sort the numbers by using different algorithm using ML.



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

List of Program

1. Simple LISP queries
2. LISP queries to print, control structures (if then else) and defining a function
3. PROLOG queries for understanding impact of Rule Order and Goal Order in PROLOG Programming languages.
4. Write a Program to Swapping of two no's with or without using third variable in Meta Language.
5. Write a Program to implement Fibonacci Series in ML.
6. Write a Program to Sum of a digit. (e.g. digit=213, sum= 6)
7. Write a Program to find the Factorial of a number in ML
8. Write a Program to find Difference of squares. (if $x > y$ return $x^2 - y^2$, otherwise $y^2 - x^2$)
9. Program for linear search in Meta Languages (ML)
10. Program for Binary search in Meta Languages (ML)
11. Program for insertion sort in ML
12. Program for bubble sort in ML
13. Program for Merge sort in ML
14. Program for Quick sort in ML

Value Addition:

15. Program for Making Dictionary in ML
16. Program for understanding cuts in PROLOG Programming languages



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-1

Objective: Simple LISP queries.

In this we have to write simple Lisp queries performing the simple operation like calculating the first, rest etc.

LISP programs are made up of three basic building blocks –

- atom
- list
- string

An atom is a number or string of contiguous characters. It includes numbers and special characters.

A list is a sequence of atoms and/or other lists enclosed in parentheses.

A string is a group of characters enclosed in double quotation marks.

The list function is rather used for creating lists in LISP.

The list function can take any number of arguments and as it is a function, it evaluates its arguments.

The first and rest functions give the first element and the rest part of a list



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

EXPERIMENT NO. 2

Objective: LISP queries to print, control structures (if then else), setf, let, loop control structures and defining a function.

Our aim is to write LISP queries to print and queries based on the control structures like if, then, else and also to define any function.

1. **Print** : Print is an Output function. All output functions in LISP take an optional argument called output-stream, where the output is sent. If not mentioned or nil, output-stream defaults to the value of the variable *standard-output*. The print function prints the object with a preceding newline and followed by a space. It returns object.
2. **PROG**: PROG works like this: Before it begins it sets all the locals to NIL. The old values of these locals are saved, and these are used only while Lisp is doing the PROG. Thus, when the PROG is done, whatever old values were in the locals come back. PROG begins with the first object. If it is an atom, then it simply ignores it. If the object is not an atom then it evaluates it and if it can, goes on to the next object organization to the program.
3. **If then else**: Control Structures are something that most programming languages provide in order to help the programmer organize their thoughts and, thus, lend better. The if construct has various forms. In simplest form it is followed by a test clause, a test action and some other consequent action. If the test clause evaluates to true, then the test action is executed otherwise, the consequent clause is evaluated.
4. **Setf**: setf returns the value of the last form
5. **Let**: When let is executed, each variable is assigned the respective value and lastly the s-expression is evaluated. The value of the last expression evaluated is returned. If you don't include an initial value for a variable, it is assigned to nil.
6. **Dotimes**: The dotimes construct allows looping for some fixed number of iterations.

7. Functions: The macro named **defun** is used for defining functions. The **defun** macro needs three arguments –

- Name of the function
- Parameters of the function
- Body of the function



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-3

Objective: Program for understanding impact of Rule Order and Goal Order in PROLOG Programming languages.

Procedure:

Rule: A rule can be viewed as an extension of a fact with added conditions that also have to be satisfied for it to be true. It consists of two parts. The first part is similar to a fact (a predicate with arguments). The second part consists of other clauses (facts or rules that are separated by commas) which must all be true for the rule itself to be true. These two parts are separated by ":-". You may interpret this operator as "if" in English.

Goals

A goal is a statement starting with a predicate and probably followed by its arguments. In a valid goal, the predicate must have appeared in at least one fact or rule in the consulted program, and the number of arguments in the goal must be the same as that appears in the consulted program. Also, all the arguments (if any) are constants.



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT

N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-4

Objective:

Write a Program to Swapping of two no's with or without using third variable in Meta Language.

Procedure 1:

Given two variables, x and y, swap two variables with using a third variable.

```
x = 5  
y = 7  
t = x  
x = y  
y = t
```

Output:

```
x = 7  
y = 5
```

Procedure 2:

Given two variables, x and y, swap two variables without using a third variable.

```
x = 5  
y = 7  
x = x + y  
y = x - y  
x = x - y
```

Output:

$$x = 7$$
$$y = 5$$



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-5

Objective:

Write a Program to implement Fibonacci Series in ML.

Procedure:

The Fibonacci numbers are the numbers in the following integer sequence.
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.....

In case of Fibonacci series, *next number is the sum of previous two numbers*

$$F_n = F_{n-1} + F_{n-2}$$

Step 1: Start

Step 2: Declare variable a, b, c, n, i

Step 3: Initialize variable $a = 1, b = 1, i = 2$

Step 4: Read n from user

Step 5: Print a and b

Step 6: Repeat until $i < n$

6.1 $c = a + b$

6.2 *print c*

6.3 $a = b, b = c$

6.4 $i = i + 1$

Stop 7: Stop



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-6

Objective:

Write a Program to Sum of a digit. (E.g. digit=213, sum= 6)in ML.

Procedure:

1. Take the integer as input.
2. Divide the input integer by 10; obtain its remainder and quotient.
3. Increment the new variable with the remainder got at step 2.
4. Repeat the step 2 & 3 with the quotient obtained until the quotient becomes zero.
5. Print the output and exit.

Algorithm:

Step 1: Input N

Step 2: Sum = 0

Step 3: While ($N \neq 0$)

 Rem = $N \% 10$;

 Sum = Sum + Rem;

$N = N / 10$;

Step 4: Print Sum



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-7

Objective:

Write a Program to find the Factorial of a number in ML

Procedure

Step 1: Declare n and Factorial as integer variable.

Step 2: Initialize Factorial = 1.

Step 2: Enter the value of N.

Step 3: Check whether $N > 0$, if not then Factorial = 1.

Step 4: If yes then, Factorial = Factorial * N

Step 5: Decrease the value of N by 1 .

Step 6: Repeat step 4 and 5 until $N = 0$.

Step 7: Now print the value of F.



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-8

Objective:

Write a Program to find Difference of squares. (if $x > y$ return $x^2 - y^2$, otherwise $y^2 - x^2$).

Procedure

Step 1: Take variable x, y as an input

Step 2: By taking function name as fun_diffs

Step 3: If $x > y$ then $x*x - y*y$

Step 4: else $y*y - x*x$



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT

N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-9

Objective: Program for linear search in Meta Languages (ML)

Procedure

- Step 1: Take an input variable x
- Step 2: Take an input list z
- Step 3: By taking function name as fun search
- Step 4: if x= head of the list
- Step 5 : Then display print x is print
- Step 6: Else search tale of the list
- Step 7: Print the result



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghat Road Bypass Crossing, Meerut-250005

Program No.-10

Objective: Program for Binary search in Meta Languages (ML)

Binary Search Algorithm

Binary Search is applied on the sorted array or list of large size. Its time complexity of $O(\log n)$ makes it very fast as compared to other sorting algorithms. The only limitation is that the array or list of elements must be sorted for the binary search algorithm to work on it.

Implementing Binary Search Algorithm

Following are the steps of implementation that we will be following for Binary search algorithm:

1. Start with the middle element:
 - If the **target** value is equal to the middle element of the array, then return the index of the middle element.
 - If not, then compare the middle element with the target value,
 - If the target value is greater than the number in the middle index, then pick the elements to the right of the middle index, and start with Step 1.
 - If the target value is less than the number in the middle index, then pick the elements to the left of the middle index, and start with Step 1.
2. When a match is found, return the index of the element matched.
3. If no match is found, then return **-1**



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-11

Objective: Program for insertion sort in ML

Insertion Sort Algorithm

Consider you have 10 cards out of a deck of cards in your hand and they are sorted, or arranged in the ascending order of their numbers.

If I give you another card, and ask you to **insert** the card in just the right position, so that the cards in your hand are still sorted. What will you do?

Well, you will have to go through each card from the starting or the back and find the right position for the new card, comparing its value with each card. Once you find the right position, you will **insert** the card there.

Similarly, if cards that are more new are provided to you, you can easily repeat the same process, insert the new cards, and keep the cards sorted too.

This is exactly how **insertion sort** works. It starts from the index **1**(not **0**), and each index starting from index **1** is like a new card, that you have to place at the right position in the sorted sub array on the left.

Implementing Insertion Sort Algorithm

Following are the **steps** involved in insertion sort:

1. We start by making the second element of the given array, i.e. element at index **1**, the **key**.
The **key** element here is the new card that we need to add to our existing sorted set of cards(remember the example with cards above).
2. We compare the **key** element with the element(s) before it, in this case, element at index **0**:

- If the *key* element is less than the first element, we insert the *key* element before the first element.
 - If the *key* element is greater than the first element, then we insert it after the first element.
3. Then, we make the third element of the array as *key* and will compare it with elements to its left and insert it at the right position.
 4. And, we go on repeating this, until the array is sorted.



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghat Road Bypass Crossing, Meerut-250005

Program No.-12

Objective: Program for bubble sort in ML

Bubble Sort Algorithm

Bubble Sort is a simple algorithm, which is used to sort a given set of n elements provided in form of an array with n number of elements. Bubble Sort compares all element one by one and sort them based on their values.

If the given array has to be sorted in ascending order, then bubble sort will start by comparing the first element of the array with the second element, if the first element is greater than the second element, it will **swap** both the elements, and then move on to compare the second and the third element, and so on.

If we have total n elements, then we need to repeat this process for $n-1$ times.

It is known as **bubble sort**, because with every complete iteration the largest element in the given array, bubbles up towards the last place or the highest index, just like a water bubble rises up to the water surface.

Sorting takes place by stepping through all the elements one-by-one and comparing it with the adjacent element and swapping them if required.

Implementing Bubble Sort Algorithm

Following are the steps involved in bubble sort (for sorting a given array in ascending order):

1. Starting with the first element (index = 0), compare the current element with the next element of the array.
2. If the current element is greater than the next element of the array, swap them.
3. If the current element is less than the next element, move to the next element. **Repeat Step 1.**



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-13

Objective: Program for Merge sort in ML

Merge Sort: Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being $O(n \log n)$, it is one of the most respected algorithms. Merge sort first divides the array into equal halves and then combines them in a sorted manner.

How Merge Sort Works?

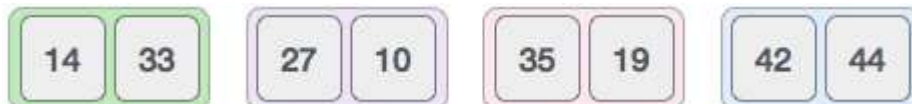
To understand merge sort, we take an unsorted array as the following –



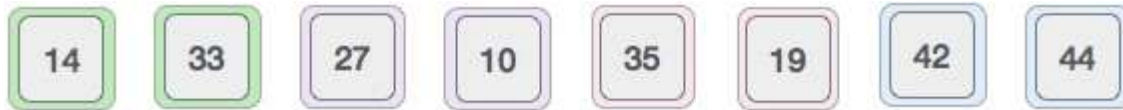
We know that merge sort first divides the whole array iteratively into equal halves unless the atomic values are achieved. We see that an array of 8 items is divided into two arrays of size 4.



This does not change the sequence of appearance of items in the original. Now we divide these two arrays into halves.



We further divide these arrays and we achieve atomic value which can no more be divided.



Now, we combine them in exactly the same manner as they were broken down. Please note the color codes given to these lists.

We first compare the element for each list and then combine them into another list in a sorted manner. We see that 14 and 33 are in sorted positions. We compare 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.



In the next iteration of the combining phase, we compare lists of two data values, and merge them into a list of found data values placing all in a sorted order.



After the final merging, the list should look like this –



Now we should learn some programming aspects of merge sorting.

Algorithm

Merge sort keeps on dividing the list into equal halves until it can no more be divided. By definition, if it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

Step 1 – if it is only one element in the list it is already sorted, return.

Step 2 – divide the list recursively into two halves until it can no more be divided.

Step 3 – merge the smaller lists into new list in sorted order.



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-14

Objective: Program for Quick sort in ML

Quick Sort

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Quick sort partitions an array and then calls itself recursively twice to sort the two resulting subarrays. This algorithm is quite efficient for large-sized data sets as its average and worst case complexity are of $O(n^2)$, where n is the number of items.

Partition in Quick Sort

Following representation explains how to find the pivot value in an unsorted array.



The pivot value divides the list into two parts. And recursively, we find the pivot for each sub-lists until all lists contains only one element.

Quick Sort Pivot Algorithm

Based on our understanding of partitioning in quick sort, we will now try to write an algorithm for it, which is as follows.

Step 1 – Choose the highest index value has pivot

Step 2 – Take two variables to point left and right of the list excluding pivot

Step 3 – left points to the low index

Step 4 – right points to the high

Step 5 – while value at left is less than pivot move right

Step 6 – while value at right is greater than pivot move left

Step 7 – if both step 5 and step 6 does not match swap left and right

Step 8 – if $\text{left} \geq \text{right}$, the point where they met is new pivot



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-15

Objective: Program for Making Dictionary in ML

A dictionary, sometimes known as a wordbook, is a collection of words in one or more specific languages, often arranged alphabetically which might include information on definitions, usage, etymologies, pronunciations, translation, etc. or a book of words in one language with their equivalents in another, sometimes known as a lexicon. A lexicographical reference shows inter-relationships among the data.



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

Program No.-16

Objective: Program for understanding cuts in PROLOG Programming languages.

Cut: The cut, in Prolog, is a goal, written as `!,` which always succeeds, but cannot be backtracked past. It is used to prevent unwanted backtracking, for example, to prevent extra solutions being found by prolog. The cut should be used sparingly. There is a temptation to insert cuts experimentally into code that is not working correctly. If you do this, bear in mind that when debugging is complete, you should understand the effect of, and be able to explain the need for, every cut you use. The use of a cut should thus be commented.



Program No.-1

Objective: Simple LISP queries.

(a) Evaluate the following forms:

(first '(p h w))

(rest '(b k p h))

(first '((a b) (c d)))

(rest '((a b) (c d)))

(first (rest '((a b) (c d))))

(rest (first '((a b) (c d))))

(rest (first (rest '((a b) (c d))))))

(first (rest (first '((a b) (c d))))))

OUTPUT

CL-USER 6 > (first '(p h w))

P

CL-USER 7 > (rest '(b k p h))

(K P H)

CL-USER 8 > (first '((a b) (c d)))

(A B)

CL-USER 9 > (rest '((a b) (c d)))

((C D))

CL-USER 10 > (first (rest '((a b) (c d))))

(C D)

(B)

CL-USER 12 > (rest (first (rest '((a b) (c d))))))

(D)

CL-USER 13 > (first (rest (first '((a b) (c d))))))

B



MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

EXPERIMENT NO. 2

Objective: LISP queries to print, control structures (if then else) and defining a function.

Description: Our aim is to write LISP queries to print , and queries based on the control structures like if, then ,else and also to define any function.

PRINT

1. (print (+ 2 3 4 5))

OUTPUT

14

14

2. (+ (print (* 2 3)) (print (/ 3 2))9)

OUTPUT

6

3/2

33/2

PROGN

- ```
(if (> 3 2)
 (progn (print "hello") (print "yo")
 (print "whassup?") 9)
 (+ 4 2 3))
```

**OUTPUT**

"hello"

"yo"

"whassup?"

9

## **IF**

1. (if (<= 3 2) (\* 3 9) (+ 4 2 3))

**OUTPUT**

9

2. (if (> 2 3) 9)

**OUTPUT**

NIL

3. (if (= 2 2) (if (> 3 2) 4 6) 9)

**OUTPUT**

4

4. (+ 4 (if (= 2 2) (\* 9 2) 7))

**OUTPUT**

22

## **LISP queries of setf, let, loop control structures**

### **SETF**

1. (setf x (\* 3 2))

**OUTPUT**

6

2. X

**OUTPUT**

6

3. (setf y (+ x 3))

**OUTPUT**

9

4. (\* x y)

**OUTPUT**

54

5. (setf sin 9)

**OUTPUT**

9

6. (sin sin)

**OUTPUT**

0.4121185

## **LET**

1. (setf x 4)

**OUTPUT**

4

2. (let ((x 3))

(print x)

(setf x 9)

(print x)

(print "hello"))

**OUTPUT**

3

9

"hello"

"hello"

3. (let ((x 3))

(print x)

(let (x)

(print x)

(let ((x "hello"))

(print x))

(print x))

(print x)

(print "yo"))

**OUTPUT**

3

NIL

"hello"

NIL

3

```
"yo"
"yo"
```

1. (setf x 3)

**OUTPUT**

```
3
```

2. (dotimes (x 4 "yo") (print "hello"))

**OUTPUT**

```
"hello"
"hello"
"hello"
"hello"
"yo"
```

3. (setf bag 2)

**OUTPUT**

```
2
```

4. (dotimes (x 6) (setf bag (\* bag bag)))

**OUTPUT**

```
NIL
```

5. bag

**OUTPUT**

```
18446744073709551616
```

**Function that computes the double, triple, square and cube of a number.**

**DOUBLE**

```
(defun double (n) (*n 2))
```

**OUTPUT**

```
(DOUBLE 4)
```

```
8
```

```
(DOUBLE 7)
```

```
14
```

## **TRIPLE**

```
(defun triple (n) (* n 2))
```

### **OUTPUT:**

```
(TRIPLE 4)
```

```
12
```

```
(TRIPLE 7)
```

```
21
```

## **SQUARE**

```
(defun square (n) (* n n))
```

### **OUTPUT**

```
(SQUARE 5)
```

```
25
```

## **CUBE**

```
(defun cube (n) (* n n n))
```

### **OUTPUT**

```
(SQUARE 5)
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.-3**

**Objective:** Program for understanding impact of Rule Order and Goal Order in PROLOG Programming languages.

**Knowledge Base 1**

woman(mia).  
woman(jody).  
woman(yolanda).  
playsAirGuitar(jody).  
party.

?- woman(mia).

true

?- playsAirGuitar(jody)

true

?- playsAirGuitar(mia).

false

?- playsAirGuitar(vincent).

false

?- party

true

?- rockConcert.

false

**Knowledge Base 2**



Here is KB2, our second knowledge base:

```
happy(yolanda).
listens2Music(mia).
listens2Music(yolanda):- happy(yolanda).
playsAirGuitar(mia):- listens2Music(mia).
playsAirGuitar(yolanda):- listens2Music(yolanda).

?- playsAirGuitar(mia).
```

true

```
?- playsAirGuitar(yolanda).
```

true

### **Knowledge Base 3**

KB3, our third knowledge base, consists of five clauses:

```
happy(vincent).
listens2Music(butch).
playsAirGuitar(vincent):-
 listens2Music(vincent),happy(vincent).
playsAirGuitar(butch):-happy(butch).
playsAirGuitar(butch):- listens2Music(butch).
```

```
?- playsAirGuitar(vincent).
```

false

```
?- playsAirGuitar(butch).
```

false

### **Knowledge Base 4**

Here is KB4, our fourth knowledge base:

```
woman(mia).
woman(jody).
woman(yolanda).
 likes (vincent,mia).
likes(marsellus,mia).
likes (pumpkin,honey_bunny).
likes (honey_bunny,pumpkin).
?- woman(X).
```

mia

?- likes (marsellus,X), woman(X).

mia

### **Knowledge Base 5**

likes (vincent,mia).

likes (marsellus,mia).

likes (pumpkin,honey\_bunny).

likes (honey\_bunny,pumpkin).

jealous(X,Y):- likes (X,Z), likes (Y,Z).

?- jealous(marsellus,W).

### **Knowledge base implement family relationship.**

male(sunny).

male(ashok).

male(sheetal).

male(sk).

female(saroj).

female(sadhana).

female(divya).

parent(sheetal,sunny).

parent(sheetal,ashok).

parent(sheetal,divya).

parent(sadhana,sunny).

parent(sadhana,ashok).

parent(sadhana,divya).

parent(sk,sheetal).

father(X,Y) :- parent(X,Y), male(X).

mother(X,Y):-parent(X,Y),female(X).

grandfather(X,Y):- parent(X,Z),parent(Z,Y),male(X).

grandmother(X,Y):- parent(X,Z),parent(Z,Y),female(X).

sibling(X,Y) :- parent(Z,X), parent(Z,Y) ,X\=Y.

granddaughter(X,Y):- parent(Z, Y),parent(Y,X),female(X).

grandson(X,Y):-parent(Z, Y),parent(Y,X),male(X).

brother(X,Y):- parent(Z,X),parent(Z, Y),male(X), X\=Y.

sister(X,Y):-parent(Z,X),parent(Z, Y),female(X),X\=Y.

### **Output:**

?-female(X).

X=saroj;

X=sadhana;

X=divya;

false.

?-father(X,sunny).

X=sheetal.

?-grandfather(X,sunny).

X=sk;

false

?-grandfather(X,Y).

X=sk, Y=sunny;

X=sk Y=divya;

X=sk Y=ashok;

false

?-sibling(X,sunny).

X=divya;

X=ashok;

X=divya;

X=ashok;

false

?-female(divya)

true

?-female(div).

false

?-parent(X,sunny).

X=sheetal;

X=sadhana.

?-parent(X,Y).

X=sheetal

Y=sunny;

X=sheetal

Y=divya;

X=sheetal

Y=ashok;

X=sadhana

Y=sunny;

X=sadhana

Y=divya;

X=sadhana

Y=ashok;

X=sk

Y=sheetal.



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.-4**

**Objective:**

Write a Program to Swapping of two no's with or without using third variable in Meta Language.

**Program:**

```
val x =5 : int;
val y = 7 : int;
val t=1 :int ;
val t=x;
val x=y;
val y=t;
```

**Output:**

```
miet@miet-Veriton-M200-H61:~$ sml
Standard ML of New Jersey v110.78 [built: Thu Jul 23 11:20:34 2015]
- use"swap.sml";
[opening swap.sml]
val x = 5 : int
val y = 7 : int
val t = 1 : int
val t = 5 : int
val x = 7 : int
val y = 5 : int
val it = () : unit
-
```

## Program No.-5

### Objective:

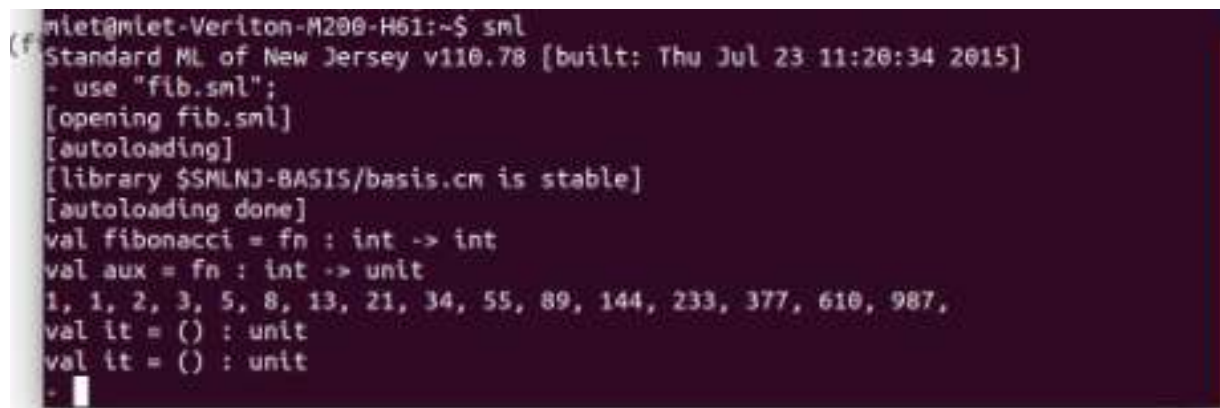
Write a Program to implement Fibonacci Series in ML.

### Program:

```
fun fibonacci n =
 if n < 3 then 1
 else
 fibonacci (n-1) + fibonacci (n-2)

fun aux n =
 if n > 16 then
 print "\n"
 else (
 print (Int.toString (fibonacci n) ^ ", ");
 aux (n + 1)
);
aux 1;
```

### Output:



```
mlet@mlet-Veriton-M200-H61:~$ sml
Standard ML of New Jersey v110.78 [built: Thu Jul 23 11:20:34 2015]
- use "fib.sml";
[opening fib.sml]
[autoloading]
[library $SMLNJ-BASIS/basis.cm is stable]
[autoloading done]
val fibonacci = fn : int -> int
val aux = fn : int -> unit
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987,
val it = () : unit
val it = () : unit
- |
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.- 6**

**Objective:**

Write a Program to Sum of a digit. (eg. digit=213, sum= 6) in ML.

**Program:**

```
fun sumDigits (n) =
 if n < 10 then n
 else
 n mod 10 + sumDigits(n div 10);
```

**Output:**

A screenshot of a terminal window titled 'SML of New Jersey'. The window shows the following text:

```
Standard ML of New Jersey v110.82 [built: Wed Oct 25 10:36:05 2017]
- fun sumDigits (n) =
= if n<10 then n
= else
= n mod 10 + sumDigits (n div 10);
val sumDigits = fn : int -> int
- sumDigits (1234)
= ;
val it = 10 : int
-
```





**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghat Road Bypass Crossing, Meerut-250005

---

**Program No.-7**

**Objective:**

Write a Program to find the Factorial of a number in ML

.

**Program:**

```
factorial n =
 if n <= 1 then
 1
 else
 factorial (n-1) * n;
```

```
fun aux n =
 if n > 16 then
 ()
 else (
 print (Int.toString n ^ "! = " ^ Int.toString (factorial n) ^ "\n");
 aux (n + 1)
);
aux 0;
```

**Output:**

```
terminal
miet@miet-Veriton-M200-H61: ~
10! = 3628800
11! = 39916800
12! = 479001600
uncaught exception Overflow [overflow]
 raised at: <file fact.sml>
- use "fact.sml";
[opening fact.sml]
val factorial = fn : int -> int
val aux = fn : int -> unit
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
val it = () : unit
val it = () : unit
-
```

## Program No.-8

### Objective:

Write a Program to find Difference of squares. (if  $x > y$  return  $x^2 - y^2$ , otherwise  $y^2 - x^2$ ).

### Program:

```
Fun diffs (x,y)=
 If x>y then
 x*x - y*y
 else
 y*y-x*x
```

### Output:

```
- fun diffs (x,y)=
= if x>y then
= x*x-y*y
= else
= y*y-x*x;
val diffs = fn : int * int -> int
- diffs (5,3);
val it = 16 : int
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

### **Program No.-9**

**Objective:** Program for linear search in Meta Languages (ML)

**Program:**

```
fun search(x: int, z) =
 if x = hd(z) then
 print(x is present)
 else search(x, tl(z));
```

**Output:**

**At terminal**

```
search(2, [1,2,3,4]);
```

```
val search = fn : int * int list -> unit
x is present
val it = () : unit
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.-10**

**Objective:** Program for Binary search in Meta Languages (ML)

**Program:**

```
open Array;
fun binsearch (A, x) =
 let val n = length A;
 val lo = ref 0 and hi = ref n;
 val mid = ref ((!lo + !hi) div 2);
 in
 while ((!hi - !lo > 1) andalso (x <> sub (A, !mid))) do
 (
 if x < sub (A, !mid) then hi := !mid - 1
 else lo := !mid + 1;
 mid := (!lo + !hi) div 2
);
 if x = sub (A, !mid) then SOME (!mid)
 else NONE
 end;
 open Array;
 val A = fromList [~24, ~24, ~12, ~12, 0, 0, 1, 20, 45, 123];
 binsearch (A, 0);
 binsearch (A, ~24);
 binsearch (A, 123);
 binsearch (A, 100);
 binsearch (A, ~25);
 binsearch (A, 124);
```

**Output:**

**At terminal**

```
val it = SOME 5 : int option
val it = SOME 0 : int option
val it = SOME 9 : int option
val it = NONE : int option
```

```
val it = NONE : int option
val it = NONE : int option
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

### Program No.-11

**Objective:** Program for insertion sort in ML

**Program:**

```
fun insert x [] = [x]
 | insert x (y::ys) =
 if x < y
 then x :: y :: ys
 else y :: (insert x ys)
```

```
fun isort [] = []
 | isort (x::xs) =
 insert x (isort xs)
```

**Output:**

**At terminal**

```
isort [6, 2, 4, 3]; {Input}
```

```
Val it = [2, 3, 4, 6] : int list
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.-12**

**Objective:** Program for bubble sort in ML

**Program:**

```
fun issorted [] = true |
 issorted [x] = true |
 issorted (x::y::t) = x <= y andalso issorted(y::t);
```

(\* Function that does the bubbling \*)

```
fun bubble [] = [] |
 bubble [x] = [x] |
 bubble (x::y::t) = if (x > y) then y::(bubble (x::t))
 else x::(bubble (y::t));
```

(\* Call bubble on list until it is sorted \*)

```
fun bubbleSort [] = [] |
 bubbleSort l = if (issorted l) then l else bubbleSort (bubble l);
```

**Output:**

**At terminal**

```
bubblesort [6,2,4,3]; {Input}
```

```
Val it = [2,3,4,6] : int list
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.-13**

**Objective:** Program for Merge sort in ML

**Program:**

```
fun merge [] M = M
 | merge L [] = L
 | merge (L as x::xs) (M as y::ys) = if x < y then x :: (merge xs M)
 else y :: (merge L ys);

fun split L =
 let
 val t = (length L) div 2
 in
 (List.take (L,t), List.drop (L,t))
 end;

fun mergesort [] = []
 | mergesort [x] = [x]
 | mergesort xs =
 let
 val (ys,zs) = split xs
 in
 merge (mergesort ys) (mergesort zs)
 end;
```

**Output:**

**At terminal**

```
mergesort [6,2,4,3]; {Input}
```

```
Val it = [2,3,4,6] : int list
```





**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.-14**

**Objective:** Program for Quick sort in ML

**Program:**

```
fun Quicksort [] = []
 | Quicksort (x::xs) =
 let
 val (S,B) = partition (x,xs)
 in
 (Quicksort S) @ (x :: (Quicksort B))
 end
```

```
fun partition (p,[]) = ([],[])
 | partition (p,x::xs) =
 let
 val (S,B) = partition (p,xs)
 in
 if x < p then (x::S,B)
 else (S,x::B)
 end
```

**Output:**

**At terminal**

```
Quicksort [6,2,4,3]; {Input}
```

```
Val it = [2,3,4,6] : int list
```



**MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT**  
N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing, Meerut-250005

---

**Program No.-15**

**Objective:** Program for Making Dictionary in ML

**Program:**

```
signature ORDERED =
sig
 type t
 val compare : t * t -> order (* LESS, EQUAL, or GREATER *)
end
```

```
signature DICTIONARY =
sig
 type key
 type 'v dict
 val empty : 'v dict
 val insert : 'v dict -> key * 'v -> 'v dict
 val lookup : 'v dict -> key -> 'v option
 val remove : 'v dict -> key -> 'v dict option
end
```

```
functor ListDict (Key : ORDERED) : DICTIONARY =
struct
 type key = Key.t
 type 'v dict = (Key.t * 'v) list
 val empty = nil
```

```
 fun insert D (k, v) = (k, v)::D
```

```
 fun lookup nil k = NONE
 | lookup ((k', v)::D) k =
 (case Key.compare (k, k') of EQUAL => SOME v | _ => lookup D k)
```

```
 fun remove nil k = NONE
 | remove ((k', v)::D) k =
 (case Key.compare (k, k') of
 EQUAL => (case remove D k of NONE => SOME D | SOME D' => SOME D')
```

```
|_ => (case remove D k of NONE => NONE | SOME D' => SOME D'))
end
```

```
structure IntOrdered : ORDERED =
struct
 type t = int
 val compare = Int.compare
end
```

```
structure IntListDict = ListDict(IntOrdered)
```

## Program No.-15

**Objective:** Program for understanding cuts in PROLOG Programming languages.

### Program:

```
teaches(dr_fred, history).
teaches(dr_fred, english).
teaches(dr_fred, drama).
teaches(dr_fiona, physics).
```

```
studies(alice, english).
studies(angus, english).
studies(amelia, drama).
studies(alex, physics).
```

### Output:

```
? – teaches(dr_fred, Course), studies(Student, Course).
Course = english
Student = alice ;
Course = english
Student = angus ;
Course = drama
Student = amelia ;
false.
```

```
? – teaches(dr_fred, Course), !, studies(Student, Course).
false.
```

```
? – teaches(dr_fred, Course), studies(Student, Course), !.
Course = english
Student = alice ;
false.
```