helper functions

1. **val len : 'a list -> int = <fun> :** this function gives length of list by using following recurrence relation
   len(h::t) = 1 + len(t)

2. **val checkm : matrix -> int -> int -> bool = <fun>**
   :this function checks that given float list list is valid matrice or not. It checks for every row length as. It uses following recurrence relation.
   Checkm(*h::t,n,m*) = (checkm(t,*n,m*)) && (len *h* == *n*)

3. **val addlist : float list -> float list -> float list = <fun>**
   the function add two list element by element

4. **val mkidlist : int -> int -> int -> float list -> float list = <fun>**
   make list with given length and set all elements to zero except one element 1.0

5. **val makezerol : int -> float list = <fun>**
   make list with all elements 0.0

6. **val iszerol : float list -> bool = <fun>**
   check whether given list is zero or not

7. **val get1 : 'a * 'b -> 'a = <fun>**
   val get2 : 'a * 'b -> 'b = <fun>
   this functions give first and second term of tuple respectively

8. **val split : float list list -> float list * matrix = <fun>**
   given matrice give tuple of its first column*(remaining matrice)

9. **val mullist : float list -> float -> float list = <fun>**
   multiplies given float to each list element (scalar multiplication)

10. **val isidlist : int -> int -> float list -> bool = <fun>**
    check whether given list has 1 at designated position or not

11. **val product : float list -> float list -> float = <fun>**
    multiplies two list(dot product)

12. **val first_list : float list -> matrix -> float list = <fun>**
    given list and matrice multiplies (dot product) list to each column of matrice and return list of product

13. **val printm : float list list -> unit = <fun>**
    prints the matrice

14. **val fndnzrow : float list list -> int -> int -> int * float = <fun>**
    finds first row with non zero element in its jth column

15. **val changeithelement : 'a list -> int -> 'a -> int -> 'a list = <fun>**
    replaces the ith element of list with given element

16. **val swap : 'a list -> int -> int -> 'a list = <fun>**
    swap two elements of list

17. **val dividel : float list -> float -> float list = <fun>**
    divide each element of list by given float value

18. **val sub : float list -> float list -> float list = <fun>**
    subtracts the second list from first element wise

19. **val mkcolumunit : float list list ->int ->int ->int ->float list ->float list list -> float list -> float list list * float list list = <fun>**
    this function is used to form ***RRE.*** It makes zero in jth column of all rows except ith one

20. **val split3 : matrix -> int -> matrix = <fun>**
    this function give matrix output by eliminating ith column from input one

MAIN FUNCTIONS

1. **val vdim : vector -> int = <fun>**
   same as len

2. **val mkzerov : int -> vector = <fun>**
   same as makezerol

3. **val iszerov : vector -> bool = <fun>**
   same as iszerol

4. **val addv : vector -> vector -> vector = <fun>**
   same as addlist

5. **val scalarmultv : float -> vector -> vector = <fun>**
   same as mullist

6. **val dotprodv : vector -> vector -> float = <fun>**
   same as product

7. **val mdim : matrix -> int * int = <fun>**
   same as
   (len m),(len (hd m))

8. **val mkzerom : int -> int -> matrix = <fun>**
   mkzerom(n,m) =(mkzerol m):: mkzero(n-1,m)

9. **val iszerom : matrix -> bool = <fun>**
   same as
   iszerom(h::t) = (iszerol(h)) && iszerom(t)

10. **val mkunitm : int -> matrix = <fun>**
    recursively added idlist by increasing i

11. **val isunitm : matrix -> bool = <fun>**
    check every row with isidlist function with appropriate i

12. **val addm : matrix -> matrix -> matrix = <fun>**
    recursively add rows together

13. **val scalarmultm : float -> matrix -> matrix = <fun>**
    recursively use mullist on each row

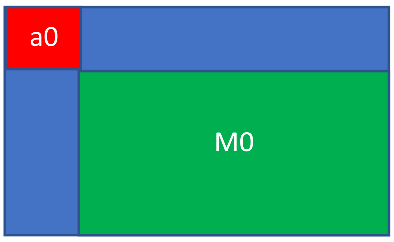
14. **val transm : matrix -> matrix = <fun>**
    trans(m) = (get1 (split m))::trans(get2 (split m))

15. **val multm : matrix -> matrix -> matrix = <fun>**
    multm(h::t,m2) = first_list(h,m2)::multm(t,m2)

16. **val detm : matrix -> float = <fun>**
    converted into the ***RRE*** and tracked all multiplications

17. **val invm : matrix -> matrix = <fun>**
    Did gaussian elimination

18. **val crossprodv : vector list -> vector = <fun>**
    ith element of vector is as follows



$V_i = (-1)^i * a_i * \mathrm{detm}(M_i)$