Linear Searching

- In this each element of an array is checked in order from lower bound to upper bound until the item is found or upper bound is reached.

Pseudocode

```
FOR Index ← 1 TO 50
    IF Names[Index] = "AHMAR"
        THEN
            OUTPUT "Found"
    END IF
END FOR
```

8 A binary search or a linear search can be used to look for a specific value in an array.

(a) Complete this pseudocode algorithm for a linear search.

```
DECLARE MyList : ARRAY[0:9] OF INTEGER
DECLARE MaxIndex : INTEGER
DECLARE Index : INTEGER
DECLARE Found : BOOLEAN
DECLARE ValueToFind : Integer

INPUT ValueToFind
Found ← FALSE
Index ← 0
MaxIndex ← 9

REPEAT
   IF MyList[Index] = ValueToFind THEN
      Found ← TRUE
   ENDIF
   Index ← Index + 1
UNTIL Found OR Index > MaxIndex

IF Found THEN
   OUTPUT "Value found at position ", Index
ELSE
   OUTPUT "Value Not Found"
ENDIF
```

[4]

Binary Searching

Purpose

- If there are 10240 elements in an array, then to search specific value, 10240 comparisons will be made but binary searching can do the same task in less time and less comparisons.

Explanation

Q- Output "Present" if 60 is in the array.

- We find the mid-value

CASE 1 : Data is at the mid-position

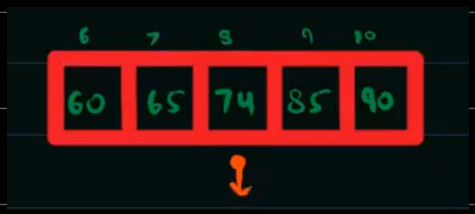
CASE 2 : Data < value at mid-position

CASE 3: Data > value at mid-position

60 > 47 ; so the values till element 5 are not required

If value required is > mid value then

- Lowerbound ← midpoint +1

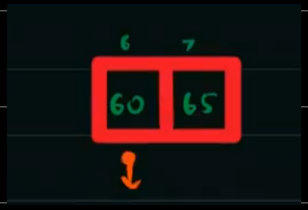


L	U	Mid
1	10	11/2 = 5
6	10	16/2 = 8
6	7	13/2 = 6

- Repeat the process
- 60 < 74
- So element 8,9,10 are not required

- If the value required is < mid value then
- Upperbound ← midpoint -1

↳ Upperbound ← 7



- 60 = 60
- OUTPUT "Present"

Pseudocode

```
Upperbound ← 10
Lowerbound ← 1
ValueFound ← FALSE
NotInList ← FALSE

WHILE ValueFound = FALSE AND NotInList = FALSE DO
    Midpoint ← INT((Upperbound + Lowerbound) / 2)   // Round(" " ")

    IF Numbers[Midpoint] = 60
      THEN
        ValueFound ← TRUE
```

```
        ELSE
            IF Numbers[Midpoint] < 60  ← Value to be searched
                THEN
                    Lowerbound ← Midpoint +1
                ELSE
                    Upperbound ← Midpoint - 1
            END IF
            IF Lowerbound > Upperbound
               THEN
                    NotInList ← TRUE
            END IF
        END IF
END WHILE
```

Q- Explain why an array needs to be sorted before a binary search algorithm can be used?

- It doesn't check every value
- The midpoint is the middle element, not the middle numerical value
- When the higher or lower elements are discarded, they will not be higher/lower element
- It might discard the value you are looking for.

File 1: Q28, 42

- As the size of list increases, the time taken to search the item increases
- Binary search takes fewer comparisons than a linear search.