Chapter-1

⊛ First palindrome string in an array

Aim:

To find and return the first palindromic string in an array of strings. If no palindromic string exists, return an empty string

* **pseudocode.**

```
for (i=0 ; i<n ; i++) {
    if (words[i] == reverse (words[i]))
      return words[i];
    }
  return" ";
```

* Input (Example-1)

words = ["abc", "car", "ada", "racecar", "cool"]

* output

ada

(Example-2)

* Input

words = ["Notpalindrome", "racecar"]

* output

racecar

* Aim: To count:

- Answer 1: the Number of elements in nums1, that also exist in nums2
- Answer 2: The number of elements in nums2 that also exist in nums1

* pseudocode:

```
answer1 = 0
answer2 = 0
for i=0 to n-1
{
  for j=0 to m-1
  {
    if nums1[i]==nums2[j]
    {
      answer1++
      break.
    }
  }
}
for i=0 to m-1
{
  for j=0 to n-1
  {
    if nums2[i]==nums1[j]
    {
      answer2++
      break
    }
  }
}
```

Input

Example 1
nums1 = [2,3,2]
nums2 = [1,2]

Example-2
nums1 = [4, 3, 2, 3, 1]
nums2 = [2, 2, 5, 2, 3, 6]

* output

example-1
[2, 1]

Example-2
[3, 4]

③

* Aim:

To find the Sum of the squares of distinct element Counts for all non-empty contiguous sub arrays of a given integer array

* Pseudocode

```
Sum=0
for i=0 to n-1
  clear freq
  d=0
  for j=i to n-1
    if freq[nums[j]]==0 then d++
    freq[nums[j]]++
      sum + =d*d
      return sum
```

* Input

ex-1

nums=[1,2,1]

Ex-②

nums=[1,1]

* output

ex-①

15

Ex-②

3

④ 0-Indexed integer.

-Aim.

To count the number of pairs (i,j) in an array such that

① nums[i]==nums[j]

② (i*j) is divisible by K.

③ 0<=i<j<n

* Pseudocode

```
Count=0
for i=0 to n-2
  for j=i+1 to n-1
    if nums[i]==nums[j] AND (i*j)%K==0
      count++
  return count
```

* Input

① nums=[3,1,2,2,2,1,3), K=2

② nums=[1,2,3,4],K=1

* output

① 4

② 0

(5) Aim:-

To find and print the maximum element in a given integer array

*pseudocode

```
Max = nums[0]
for i=1 to n-1
  if nums[i] > max
    max = nums[i]
  return max
```

*Input

Testcase 1

nums = {1,2,3,4,5}

*output

5

* Test case -2

*Input:

nums = {7,7,7,7,7}

*output

7

* Test case-3

nums = {-10, 2,3, -4, 5}

*output

5

(6) Aim:

To sort a given integer array and then find the maximum element in it, handling special using efficient sorting algorithm.

*pseudocode

```
If array isempty
  print "List empty"
else
  Sort array
  print Last element
```

Inputs

Testcase	Array Input
1	[ ]
2	[5]
3	[3,3,3,3,3)

*outputs

List empty

maximum:5

maximum: 3

⑦ Unique elements

Aim:
To create a new List containing only the unique elements from a given List of numbers

* Pseudocode.

```
Input: array arr of size n
output: array unique [] with unique elements
initialize unique[] as empty
For each element x in arr
 If each element x in arr
 add If x not in unique[]
 add x to unique[]
 print unique []
```

* Input & output

Testcase	Input	output
1	[3,7,3,5,2,5,9,2]	[3,7,5,2,9]
2	[-1,2,-1,3,2,-2)	[-1,2,3,-2)
3	[100000, 999999, 1000000]	[1000000, 99999

⑧ Bubble Sort

Aim: To sort an array of integers using the Bubble sort technique and analyze its time complexity

* pseudocode

```
Input : array arr of size n
output: sorted array arr
for i=0 to n-1
 for j = 0 to n-i-2
  If arr [j] > arr [j+1]
   Swap arr [j] and arr (j+1)
End if
 end for
 end for
 print arr
```

* Input

[64, 34, 25, 12, 22, 11, 90]

* output

Sorted array : 11 12 22 25 34 64 90

9 Binary Search

Aim:
To check if a given number exists in a sorted array using binary search and analyze its time complexity

* Pseudocode

```
Input: Sorted array arr[ ], size n, Key
output: position of Key or "not found"
Low=0
high=n-1
while Low<=high
    mid =(Low+high)/2
  If arr [mid]==Key
  print "Element found at position", mid+1,
  Return
  else if arr[mid]<Key
     Low=mid+1
  else
     high =mid-1
end while
  print "element not found"
```

* Input
[3,4,6,-9,10,8,9,30]
Key=10 -> case 1
Key=100 -> case 2

* output
Element found in the position 7

10 Sort the array in asc order

Aim: To sort an array of integers in ascending order using Heap Sort with O(n log n) time complexity and minimal extra space.

* Pseudocode

```
HeapSort(arr, n):
  Build max heap for arr [0..n-1]
  for i=n-1 down to 1:
  Swap arr [0] and arr[i]
   Heapify arr [0..i-1]
end
```

* Input
[64,34,25,12, 22,11,90]

* output
Sorted array: 11 12 22 25 34 64 99

11 out of the grid boundary

Aim: To count the number of ways a ball can move out of an mxn grid boundary in exactly N steps

* pseudocode

```
Function ways(m,n,N,i,j):
  If (i,j) outside grid: return 1
  If N==0 : return 0
Return ways(N-1, i+1,j) + ways(N-1,i-1,j) + ways(N-1, i,j+1) + ways(N-1, i,j-1)
```

* Input

m=2, n=2, N=2, i=0, j=0

m=1, n=3, N=3, i=0, j=1

* output

6

12

(12) House Robbes

Aim: find max-money without robbing adj. houses

* pseudocode

```
Function rob(nums):
If n==1: return nums[0]
Return max (rob_Linear(nums[0..n-2]), robLinear(nums[1..n-1]))
Function rob_Linear(arr):
prev=0, curr=0
for each x in arr:
  temp = max(curr, prev+x)
  prev = curr
  curr = temp
Return curr
```

* Input

[2,3,2]

[1,2,3,1]

* output

3

4

③ Staircase

AIM:
To find the number of distinct ways to reac the top of a staircase with n steps, when yo can climb 1 step or 2 steps at a time

*pseudocode

```
Start
Read n
If n==0 OR n==1
 print 1
else
 a=1
 b=1
 for i=2 to n
  c=a+b
  a=b
  b=c
 end for
 print b
end if
Stop
```

*Input

① n=4 ② n=3

*output

① 5 ② 3

④ mxn grid

Aim.
To find the number of unique paths for a robo to move from the top-left corner to the botto right corner of an mxn grid, when it can move only right or down

*pseudocode

```
Start
Read m,n
Declare dp[m][n]
for i=0 To m-1
  dp[i][0]=1
End for
for j=0 to n-1
  dp[0][j]=1
end for
for i=1 to m-1
for j=1 to n-1
  dp[i][j]=dp[i-1][j]+dp[i][j-1]
end for
end for
print dp[m-1][n-1]
Stop
```

Input

m=7 n=3

output

28

15 Lowercase String

Aim:

To find and return the start and end indices of all Large groups in a given Lowercase String

*pseudocode

```
Start
 Read string s
   n=Length of s
   i=0
  while i<n
  start=i
 while i<n AND s[i]==s[start]
   i=i+1
 END while
   end=i-1
 If(end-start+1)>=3
  print(start,end)
  ENDIF
 END WHILE
```

*INPUT

s="abbxxxxzzy"

*output

[[3,6]]

16 Game of Life

Aim: Generate next state of grid

*pseudocode

```
Start
Read board[m][n]
 Create newBoard[m][n]
foreach cell (i,j) in board
  Live neighbors=0
  foreach neighbor of (i,j)
  If neighbor is inside grid AND board[neighbor]==
   LiveNeighbors++
  ENDIf
 END FOR
If board[i][j]==1
if LiveNeighbors <2 OR Liveneighbors>3
  newboard[i][j]=0
Else
   newboard[i][j]=1
End If
Else
 If Liveneighbors=3
    newBoard[i][j]=1
  else
  NextBoard[i][j]=0
```

```
END IF
END IF
END FOR
print newBoard
STOP
```

*input:

```
board=[[0,1,0],
       [0,0,1],
       [1,1,1],
       [0,0,0]]
```

*output:

```
[[0,0,0],
 [1,0,1],
 [0,1,1],
 [0,1,0]]
```

(17) Stack glasses in a pyramid

Aim: To determine how full a specific glass is in a champagne tower after pouring a given numbe of cups, where excess champagne flows equ to the glasses below

*Pseudocode

```
Start
Read poured, query_row, query_glass
DECLARE dp[101][101] = {0}
 dp[0][0] = poured
for i = 0 to query_row
  for j = 0 to i
  excess = (dp[i][j] - 1)/2
  If excess > 0
  dp[i+1][j] += excess
  dp[i+1][j+1] += excess
    ENDIF
 ENDFOR
ENDFOR
 If dp[query_row][query_glass] >= 1
   print 1.00000
else
   print dp[query_row][query_glass]
END IF
STOP
```

* Input

```
poured=1
query_row=1
query_glass=1
```

*output

```
0.00000
```