Date: / / Page no:

- **Cyclic Sort :-**

Cycle sort is an in-place, unstable sorting algorithm

* when given no from range → (1,N) we cycle sort.

- working :-

0	1	2	3	4
3	5	2	1	4

when the array is sorted, in that case all the no are going to be at their correct index.

So, after sorting :-

1	2	3	4	5
0	1	2	3	4

↑

so, index = (value - 1)

{Check - Swap - Move}

worst case ⇒ $O(n^2)$

why ?

0	1	2	3	4
(3)	5	2	1	4

i) check is 3 at correct index.
if not do → 3-1 = 2 (index = value - 1)

Date: / / Page no:

then swap it with index no 2.

0 1 2 3 4

∴ 2, 5, 3, 1, 4

ii) after swapping we know that 3 is at correct index. Now check for 2.

(2), 5, 3, 1, 4

iii) swap 2 with index 1 value.

0 1 2 3 4

(5), 2, 3, 1, 4

iv) Check for 5 and swap with index 4.

0 1 2 3 4

(4), 2, 3, 1, 5

v) Check for 4 and swap it with index 3. (value - 1).

0 1 2 3 4

1, 2, 3, 4, 5

vi) Now every element is at correct index so -

Sorted array :- | 1 | 2 | 3 | 4 | 5 |

Date: / / Page no.

- Code implementation :-

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        int[] arr = {3, 1, 4, 2, 5};
        cycle(arr);
        Sout(Arrays.toString(arr));
    }

    static void cycle(int[] arr)
    {
        int min = arr[0];
        for(int i=1; i<arr.length; i++)
        {
            if(arr[i] < min)
            {
                min = arr[i];
            }
        }
        i=0
        while(i < arr.length)
        {
            int correct = arr[i] - min;
            if(arr[i] != arr[Correct])
            {
                int temp = arr[i];
```

```
        arr[i] = arr[correct];
        arr[correct] = temp;
    }
    else
    {
        i++ i
    }
  }
 }
}
```

* This sorting applies on both ranges such as (0 to n) and (1 to n). So when the ranges start from 0 then our (value-1) logic will fail because in that case every element will be having the same value as index.

ex:

0	1	2	3	4	5
0	1	2	3	4	5

To handle this condition we have used for loop to find the min no and then we are decreasing it from arr[i].
=> {arr[i]-1} or {arr[i]-0}.

Q=> Find the missing no.
Q-> Find the duplicate element.
Q-> Find all the missing no.
Q-> Find all the duplicate.