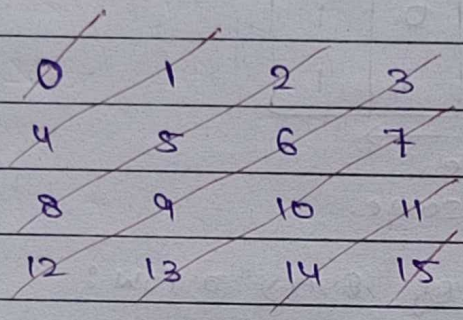
Page No.:
Date: / /

~~Dai~~ Lecture -18

## 2D-Array and vector

* **Diagonal print:**




Code:

```
col = 0;
while (col < 4) {
    i = 0, j = col;
    while (j >= 0) {
        cout << arr[i][j];
        i++;
        j--;
    }
```

Output: 0 1 4 2 5 8 3 6 9 12

```
row = 1;
while (row < 5) {
    i = row, j = 3;
    while (i <= 4) {
        cout << arr[i][j];
        i++;
        j--;
    }
}
```

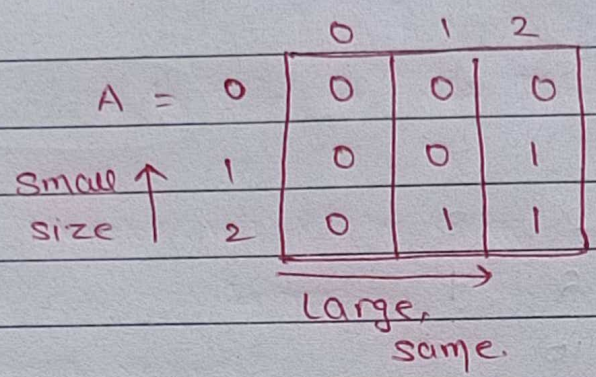
Output: 7 10 13 11 14 15

Teacher's Signature.....

Page No.
Date : / /

## * Count zeroes in a sorted Matrix :

N = 3




Find last zero of every row :

0 0 (0) → last

Count = Count + j + 1

= 3

0 (0) 0 1

↓

last Count = Count + j + 1

= 3 + 1 + 1

= 5

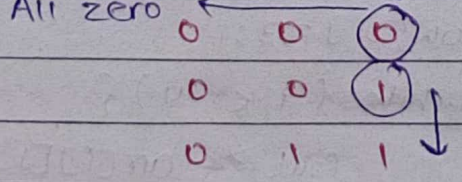
(0) 1 1

↓

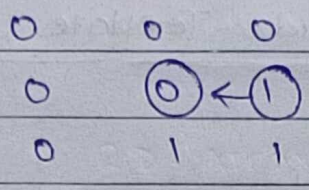
last Count = 5 + 0 + 1

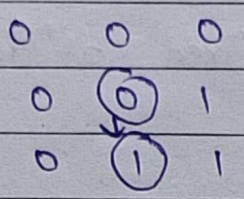
= 6

## * Start from top right :



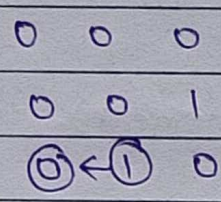

Teacher's Signature.....





When 1 found
↳ Move left.

When 0 found
↳ Move down.



count = j+1;

Till when:

1) i >= row
2) col >= 0

Code:

```
int count = 0, row = A.size(), col = A[0].size();
    int i = 0; j = col - 1;
While ( i < row && j >= 0) {
        While ( j >= 0 && A[i][j] == 1)
            j--;
        count += j+1;
        i++;
    }
    return count;
}
};
```

Page No.:
Date: / /

# STL (Standard Template Library)

## Dynamic Array (vector)

↳ Size in not fixed at initial.

```
vector<int> vector_name;
```

↳ arr

Insert value :

↳ arr.push_back (Element);

Delete value:

↳ arr.pop_back();

↳ pop last element

Size :

↳ arr.size();

For accessing first element:

↳ arr[0] or arr.front();

For accessing last element:

↳ arr[n-1] or arr.back();

For deleting all element of arr / vector:

↳ arr.clear();

vector size is always : { $2^n$

0, 1, 2, 4, 8, 16, 32 ...

Teacher's Signature

It double array size while pushing element.

* Create a vector and insert 10 element:

```
#include <vector>
  vector <int> v;
   for (int i= 1;  i<=10;  i++ ){
         v.push_back(i);
    }
         i< v.size();
   for (int  i= 0;  i<10;  i++ ){
          cout<< v[i]<<" ";
    }
```

Sort (v.begin(), v.end(), greater <int>()); O(NlogN);

cout << v.capacity() << endl; → $2^n$

cout << v.size() << endl; → actual size

v.pop_back() → delete last element

→ vector <int> arr(size);

→ vector <int> arr (size, initialise);

vector < int> arr (5, 2);

↳ 2 2 2 2 2

↳ on this capacity is 5.

$2^n$ → Not applicable

If we add one element then it double 5 to 10.

Teacher's Signature.....

Page No.:
Date: / /

## 2D - Array / Vector :

```
vector<vector<int>> arr(4, vector<int>(3));
```

array name | rows | column

```
rows   = arr.size();
column = arr[0].size();
```

Total Element = rows x column;

```
vector<vector<int>> arr(3, vector<int>(3));
    for(int i=0; i<3; i++)
        for(int j=0; j<3; j++)
            cin >> arr[i][j];

// sort(arr.begin(), arr.end());

    for(int i=0; i<3; i++)
        sort(arr[i].begin(), arr[i].end());

    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
```

Teacher's Signature .....

Page No.:
Date: / /

```
char arr[10];
  for(int i=0; i<10; i++){
        cin >> arr[i];
  }

cin >> arr;
```

Teacher's Signature.....