EXPERIMENT NO 3

Date of Completion: - 13th January 2022

Name: - Tambe Manish Mangesh

Roll No: - 221080

Gr No: - 22120186, **Batch: -** 'DSY'

AIM: - Create a database using array of structures and perform following operations on it: i. Add record ii. Display Database iii. Search record (binary search) iv. Delete record.

ALGORITHM: -

- **To Initialize arrays and variables in constructor**
  Constructor name: - SparseMatrix()
  1. Allocate memory for globally declared simple matrix 'arr' dynamically.
  2. Allocate memory for globally declared structure 'sm' dynamically.
  3. Allocate memory for globally declared structure 'sm1' dynamically.
  4. Allocate memory for globally declared structure 'result' dynamically.
  5. Assign user entered values for number of rows and columns to first element of structure 'sm' and 'sm1'.
  6. Allocate memory for globally declared array iTotal & iIndex with 'sm' structure's column count and from iIndex 'sm' structure's row count.
  7. Initialize globally declared variable 'k' with value 1;
  8. Initialize iRow and iCol with user entered values.

- **To accept regular matrix elements**
  Function name: - AcceptArray()
  1. Initialize 'iElements' variable to 0;
  2. Initialize two loop variables i.e. 'i' and 'j' to 0 itself in the loop.
  3. Check whether 'i' < iRow & 'j' < iCol.
  4. If yes then accept the value from user and increment the loop counters 'i' & 'j';
  5. If no then exit from the loop.

- **To display regular matrix elements.**
  Function name: - DisplayMatrix()
  1. Initialize two loop variables i.e., ‘i’ and ‘j’ to 0 itself in the loop.
  2. Check whether ‘i’ < iRow & ‘j’ < iCol.
  3. If yes then display the value and increment the loop counters ‘i’ & ‘j’;
  4. If no then exit from the loop.

- **To covert regular matrix in sparse matrix.**
  Function name: - ConvertConven()
  1. Initialize ‘temp’ variable to 0.
  2. Initialize two loop variables i.e., ‘i’ and ‘j’ to 0 itself in the loop.
  3. Check whether element pointed by counters are 0 or not.
  4. If not then insert the value of ‘i’ in the ‘k’ th structure element ‘iRow’ & inert the value pointed by ‘j’ in the ‘k’ th structure element ‘iCol’ and insert the value pointed by ‘i’ and ‘j’ in array arr to ‘k’ th structure element. and increment ‘k’ counter.
  5. if not increment inner loop counter i.e., ‘i’;

- **To display sparse matrix**
  Function name: - DisplayConven()
  1. Initialize two loop variables i.e., ‘i’ and ‘j’ to 0 itself in the loop.
  2. Check whether ‘i’ < iRow & ‘j’ < iCol.
  3. If yes then display the value and increment the loop counters ‘i’ & ‘j’;
  4. If no then exit from the loop.

- **Calculate simple transpose of sparse matrix**
  Function name: - CalculateSimTrans()
  1. Initialize ‘temp’ variable to 0.
  2. Initialize 0th value of ‘sm1’ with 0th value of ‘sm’.
  3. Initialize loop variables i.e., ‘i’ to 0 itself in the loop.
  4. Check whether loop counter is < ‘k’ or not.
  5. If not then insert values in structure’s ‘iRow’, ‘iCol’ & ‘iValue’ pointed by loop variable ‘i’.
  6. Increment loop counter ‘i’.

- **Display simple transpose of sparse matrix**
  Function name: - DisplaySimpleTrans()
  1. Initialize two loop variables i.e., ‘i’ and ‘j’ to 0 itself in the loop.
  2. Check whether ‘i’ < iRow & ‘j’ < iCol.
  3. If yes then display the value and increment the loop counters ‘i’ & ‘j’;
  4. If no then exit from the loop.

- **Swapping of elements from simple transpose of sparse matrix.**
  Function name: - CalculateSimSwapTrans()
  1. Initialize temp variable to 0th value.
  2. Initialize two loop variables i.e., ‘i’ & ‘j’ to ‘1’ respectively.
  3. Check whether loop counter ‘i’ is < ‘k - 1’ & ‘j’ < ‘k-1-i’ or not.
  4. If not then check whether value of ‘iRow’ from ‘sm1’ structure is less than value of ‘iRow’ from ‘sm1’ structure pointed by ‘j’ and ‘j+1’ respectively
  5. If yes then swap the elements present in ‘j’ and ‘j+1’ position from ‘sm1’ structure respectively.
  6. If not then increment the inner loop counter.

- **Displaying swapped simple transpose matrix.**
  Function name: - DisplaySimSwapTrans()
  1. Initialize two loop variables i.e., ‘i’ and ‘j’ to 0 itself in the loop.
  2. Check whether ‘i’ < iRow & ‘j’ < iCol.
  3. If yes then display the value and increment the loop counters ‘i’ & ‘j’;
  4. If no then exit from the loop.

- **Converting simple sparse matrix to fast transpose matrix**

  Function name: - ConvertFastTran()

  1. Create an initialize ‘Col’ & ‘Loc’ variables with 0.
  2. Create one loop and initialize loop variable ‘i’ to 0 in loop itself.
  3. Check whether loop counter is less than ‘sm[0].iCol’
  4. If yes then place 0 value in the position where loop counter points & increment loop counter.
  5. If no then exit from the loop.
  6. Create one loop and initialize loop variable ‘i’ to 1 in loop itself.
  7. Check whether loop counter is less than or equal to sm[0].iValue
  8. If yes then insert value from ‘iCol’ from structure pointed by counter in ‘Col’
  9. And then increment the value present at Col location of iTotal array and increment loop counter.
  10. If no then exit from loop.
  11. Initialize 0th index of ‘Iindex’ array with 0.
  12. Create one loop and initialize loop variable ‘i’ to 1 in loop itself.
  13. Check whether loop counter is less than or equal to sm[0].iValue
  14. If yes then perform iIndex[i] = iIndex[i-1] + iTotal[i-1] and increment loop counter.
  15. If no then exit from the loop.
  16. Initialize all the 0th position values of the resultant structure i.e., ‘result’ with 0th position values of ‘sm’ structure.

17. Create one loop and initialize loop variable ‘i’ to 1 in loop itself.
18. Check whether loop counter is less than or equal to sm[0].iValue
19. If yes then insert the value present at ‘iCol’ of ‘i’ th index of structure in ‘Col’ variable. & Insert value from ‘iIndex’ pointed by value of ‘Col’. Then insert the value in ‘result’ structure pointed by ‘Loc’ variable from ‘sm’ structure pointed by loop counter.
20. Increment value present at ‘iIndex’ pointed by ‘Col’ variable. & Increment counter
21. And if no then exit from the loop.

- **Displaying fast transpose sparse matrix**
  Function Name: - DisplayFastTrans()
  1. Initialize loop variables i.e., ‘i’ to 0 itself in the loop.
  2. Check whether ‘i’ <= sm[0].iValue.
  3. If yes then display the value and increment the loop counter ‘i’;
  4. If no then exit from the loop.

Code: -

```
#include<iostream>

#include<stdlib.h>

using namespace std;

struct SMATRIX {

    int iRow;

    int iCol;

    int iValue;

};

typedef struct SMATRIX sm;

class SparseMatrix

{

    private: int iRow;

    private: int iCol;

    private: int k;

    private: int **arr;

    private: int *iTotal;
```

```
private: int *iIndex;

private: SMATRIX* sm;

private: SMATRIX* sm1;

private: SMATRIX* result;

public: SparseMatrix(int iRows, int iCols)

{

    iRow = iRows;

    iCol = iCols;

    arr = new int*[iRow];

    for(int i = 0; i < iRow; i++)

    {

        arr[i] = new int[iCol];

    }

    sm = new SMATRIX[iRow*iCol];

    sm1 = new SMATRIX[iRow*iCol];

    sm[0].iRow = iRow;

    sm[0].iCol = iCol;

    sm1[0].iRow = iRow;

    sm1[0].iCol = iCol;

    iTotal = new int[sm[0].iCol];

    iIndex = new int[(sm[0].iCol)+1];

    result = new SMATRIX[iRow * iCol];

    k = 1;

}

public: ~SparseMatrix() // O(n)

{

    int i = 0;

    for(int i = 0; i < iRow; i++)
```

```
        {
            delete[] arr[i];
        }
        delete[] arr;
        delete[] sm;
        delete[] sm1;
        delete[] result;
        delete[] iTotal;
        delete[] iIndex;
    }
    public: void AcceptArray()
    {
        int iElements = 0;
        for(int i = 0 ; i < iRow; i++)
        {
            for(int j = 0; j < iCol; j++)
            {
                cout<<"Enter The element at "<<i<<" row & "<<j<<" column
=\n";
                cin>>iElements;
                arr[i][j] = iElements;
            }
        }
    }
    public: void DisplayMatrix()
    {
        cout<<"Entered elements are = \n";  // O(n *m)
        for(int i = 0 ; i < iRow; i++)
        {
            for(int j = 0; j < iCol; j++)
            {
                cout<<arr[i][j]<<"\t";
            }
            cout<<"\n";
        }
        cout<<"\n";
    }
    public: void ConvertConven()
    {
        int temp = 0;
        for(int i = 0; i < iRow; i++)
        {
            for(int j = 0; j < iCol; j++)
            {
                if(arr[i][j] != 0)
                {
                    sm[k].iRow = i;
                    sm[k].iCol = j;
                    sm[k].iValue = arr[i][j];
                    k++;
                }
            }
        }

        temp = k;
        sm[0].iValue = temp-1;
```

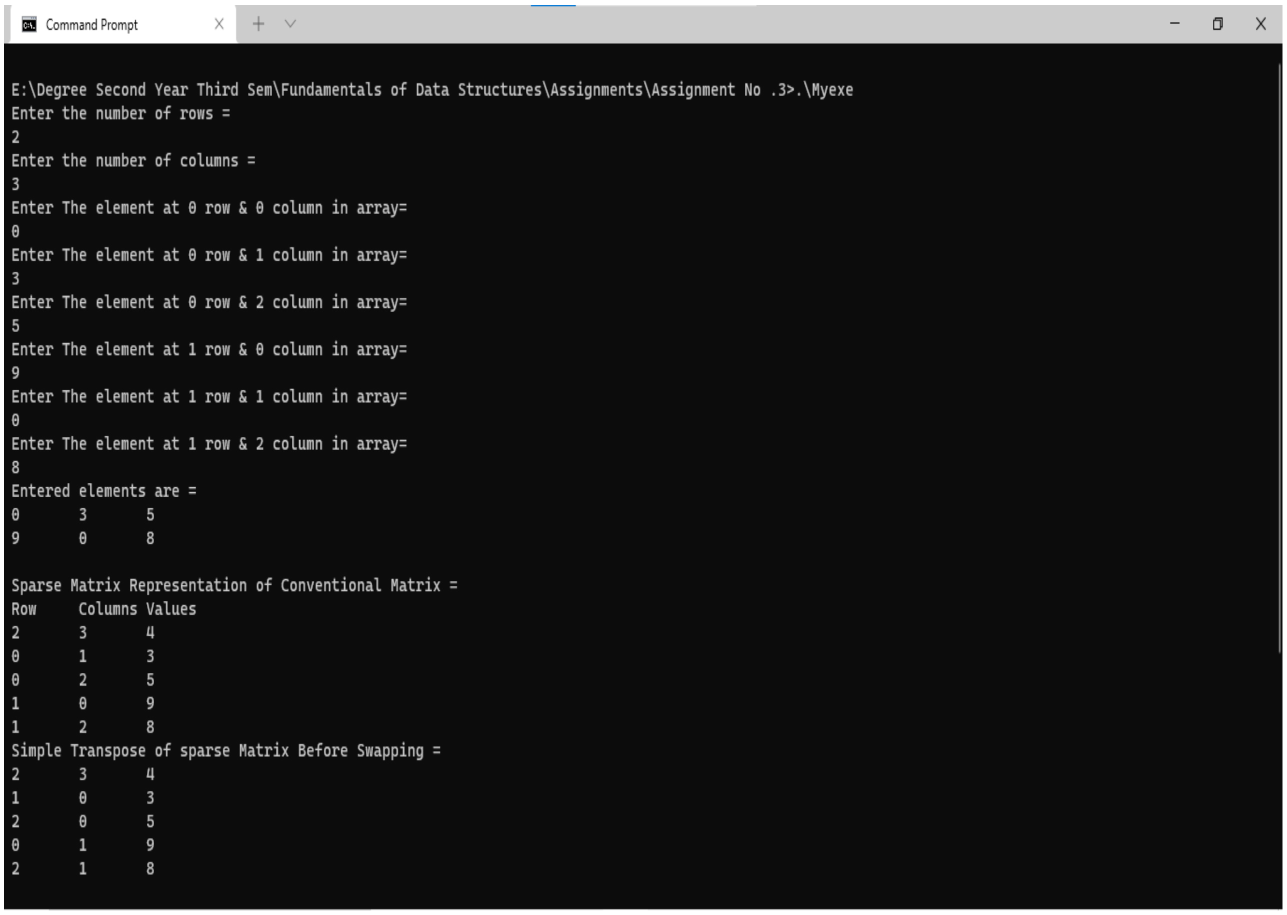
```
}
public: void DisplayConven()
{
 cout<<"Sparce Matrix Representation of Conventional Matrix = \n";
    cout<<"Row\tColumns\tValues\n";
    for(int i = 0; i < k; i++)
    {
        cout<<sm[i].iRow<<"\t";
        cout<<sm[i].iCol<<"\t";
        cout<<sm[i].iValue<<"\t\n";
    }
}
public: void CalculateSimTrans()
{
    int temp = 0;
    sm1[0].iValue = sm[0].iValue;
    for(int i = 1; i < k; i++)
    {
        sm1[i].iRow = sm[i].iCol;
        sm1[i].iCol = sm[i].iRow;
        sm1[i].iValue = sm[i].iValue;
    }
}

public: void DisplaySimpleTrans()
{
    cout<<"Simple Transpose Matrix Before Swapping = \n";
    for(int i = 0; i < k; i++)
```

```
        {
            cout<<sm1[i].iRow<<"\t";
            cout<<sm1[i].iCol<<"\t";
            cout<<sm1[i].iValue<<"\t\n";
        }
    }
    public: void CalculateSimSwapTrans()
    {
        int temp = 0;
        for(int i = 1; i < k - 1; i++)
        {
            for(int j = 1; j < k-1-i; j++)
            {
                if(sm1[j].iRow > sm1[j+1].iRow)
                {
                    iRow = sm1[j].iRow;
                    iCol = sm1[j].iCol;
                    temp = sm1[j].iValue;
                    sm1[j].iRow = sm1[j+1].iRow;
                    sm1[j].iCol = sm1[j+1].iCol;
                    sm1[j].iValue = sm1[j+1].iValue;
                    sm1[j+1].iRow = iRow;
                    sm1[j+1].iCol = iCol;
                    sm1[j+1].iValue = temp;
                }
            }
        }
    }
```

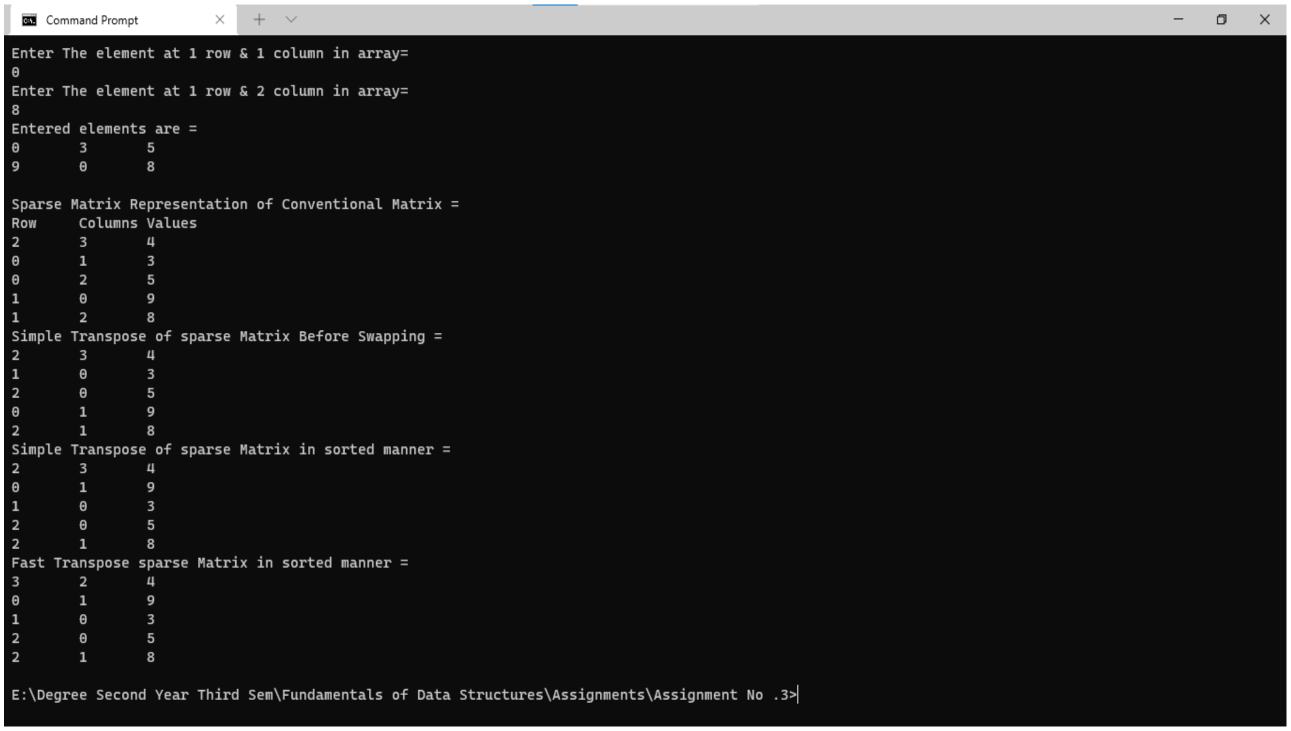
```
public: void DisplaySimSwapTrans()
{
    cout<<"Simple Transpose Matrix in sorted manner = \n";
    for(int i = 0; i < k; i++)
    {
        cout<<sm1[i].iRow<<"\t";
        cout<<sm1[i].iCol<<"\t";
        cout<<sm1[i].iValue<<"\t\n";
    }
}
public: void ConvertFastTran()
{
    int Col = 0;
    int Loc = 0;
    for(int i = 0; i < sm[0].iCol; i++)
    {
        iTotal[i] = 0;
    }
    for(int i = 1; i <= sm[0].iValue; i++)
    {
        Col = sm[i].iCol;
        iTotal[Col]++;
    }
    iIndex[0] = 1;
    for(int i = 1; i <= sm[0].iCol; i++)
    {
        iIndex[i] = iIndex[i-1] + iTotal[i-1];
    }
```

```
        result[0].iRow = sm[0].iCol;
        result[0].iCol = sm[0].iRow;
        result[0].iValue = sm[0].iValue;
        for(int i = 1; i <= sm[0].iValue; i++)
        {
            Col = sm[i].iCol;
            Loc = iIndex[Col];
            result[Loc].iCol = sm[i].iRow;
            result[Loc].iRow = sm[i].iCol;
            result[Loc].iValue = sm[i].iValue;
            iIndex[Col]++;
        }
    }
    public: void DisplayFastTras()
    {
        cout<<"Fast Transpose Matrix in sorted manner = \n";
        for(int i = 0; i <= sm[0].iValue; i++)
        {
            cout<<result[i].iRow<<"\t";
            cout<<result[i].iCol<<"\t";
            cout<<result[i].iValue<<"\t\n";
        }
    }
};
int main()
{
    int iRow = 0;
    int iCol = 0;
    cout<<"Enter the number of rows = \n";
    cin>>iRow;
    cout<<"Enter the number of rows = \n";
    cin>>iCol;
    SparseMatrix sobj(iRow, iCol);
    sobj.AcceptArray();
    sobj.DisplayMatrix();
    sobj.ConvertConven();
    sobj.DisplayConven();
    sobj.CalculateSimTrans();
    sobj.DisplaySimpleTrans();
    sobj.CalculateSimSwapTrans();
    sobj.DisplaySimSwapTrans();
    sobj.ConvertFastTran();
    sobj.DisplayFastTras();
    sobj.~SparseMatrix();
    return 0;
}
```

Output: -



```
E:\Degree Second Year Third Sem\Fundamentals of Data Structures\Assignments\Assignment No .3>.\Myexe
Enter the number of rows =
2
Enter the number of columns =
3
Enter The element at 0 row & 0 column in array=
0
Enter The element at 0 row & 1 column in array=
3
Enter The element at 0 row & 2 column in array=
5
Enter The element at 1 row & 0 column in array=
9
Enter The element at 1 row & 1 column in array=
0
Enter The element at 1 row & 2 column in array=
8
Entered elements are =
0       3       5
9       0       8

Sparse Matrix Representation of Conventional Matrix =
Row     Columns Values
2       3       4
0       1       3
0       2       5
1       0       9
1       2       8
Simple Transpose of sparse Matrix Before Swapping =
2       3       4
1       0       3
2       0       5
0       1       9
2       1       8
```



```
Enter The element at 1 row & 1 column in array=
0
Enter The element at 1 row & 2 column in array=
8
Entered elements are =
0       3       5
9       0       8

Sparse Matrix Representation of Conventional Matrix =
Row     Columns Values
2       3       4
0       1       3
0       2       5
1       0       9
1       2       8
Simple Transpose of sparse Matrix Before Swapping =
2       3       4
1       0       3
2       0       5
0       1       9
2       1       8
Simple Transpose of sparse Matrix in sorted manner =
2       3       4
0       1       9
1       0       3
2       0       5
2       1       8
Fast Transpose sparse Matrix in sorted manner =
3       2       4
0       1       9
1       0       3
2       0       5
2       1       8

E:\Degree Second Year Third Sem\Fundamentals of Data Structures\Assignments\Assignment No .3>
```

Time-Complexity of functions present in the above program: -

1. **Constructor: -** SparseMatrix()
   Time – Complexity: - O(n)

2. **Destructor: -** ~ SparseMatrix()
   Time – Complexity: - O(n)

3. **Function: -** AcceptArray()
   Time – Complexity: - O(n* m)

4. **Function: -** DisplayMatrix()
   Time - Complexity: - O(n * m)

5. **Function: -** ConvertConven()
   Time - Complexity: - O(n * m)

6. **Function: -** DisplayConven()
   Time - Complexity: - O(n)

7. **Function: -** CalculateSimTrans()
   Time - Complexity: - O(n)

8. **Function: -** DisplaySimpleTrans()
   Time - Complexity: - O(n)

9. **Function: -** CalculateSimSwapTrans()
   Time - Complexity: - O(n * m)

10. **Function: -** DisplaySimSwapTrans()
    Time - Complexity: - O(n)

11. **Function: -** ConvertFastTran()

    Time - Complexity: - O(n + m)

12. **Function: -** DisplayFastTras()

    Time - Complexity: - O(n)

(**Note: -** Where n = number of rows & m = number of columns)