1.Next permutations : 18.02.2024

```
#include <algorithm>
#include <iostream>
using namespace std;
 int main()
{

    int arr[] = { 1, 2, 3 };
    sort(arr, arr + 3);
    cout << "The 3! possible permutations with       3 elements:\n";
do {
cout << arr[0] << " " << arr[1] << " " << arr[2] << "\n";

    } while (next_permutation(arr, arr + 3));
      cout << "After loop: " << arr[0] << ' '

         << arr[1] << ' ' << arr[2] << '\n';


    return 0;
}
```

Output:

```
/tmp/kte4iYXTbM.o
The 3! possible permutations with 3 elements
    :
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
After loop: 1 2 3
```

2.Sieve of eranthosis: 19.02.2024

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>


void SieveOfEratosthenes(int n)
{
    bool prime[n + 1];
  memset(prime, true, sizeof(prime));
```

```
    for (int p = 2; p * p <= n; p++) {


        if (prime[p] == true) {
  for (int i = p * p; i <= n; i += p)
 prime[i] = false;

        }

    }

    for (int p = 2; p <= n; p++)

        if (prime[p])

            printf("%d ",p);
}
int main()
{

    int n = 30;

    printf("Following are the prime numbers smaller than or equal to %d \n", n);

    SieveOfEratosthenes(n);

    return 0;
}
```

Output:

```
/tmp/lKHCurLRKd.o
Following are the prime numbers smaller than
    or equal to 30
2 3 5 7 11 13 17 19 23 29
```

3.Search in matrix: 20.02.24

```
#include <bits/stdc++.h>
using namespace std;
vector<int> linearSearch(vector<vector<int>> arr, int target)
{

  for (int i = 0; i < arr.size(); i++) {
```

```
  for (int j = 0; j < arr[i].size(); j++) {
  if (arr[i][j] == target) {
   return {i, j};

      }

    }

  }

  return {-1, -1};
}
int main()
{
   vector<vector<int>> arr = { { 3, 12, 9 },

                               { 5, 2, 89 },

                               { 90, 45, 22 } };

  int target = 89;

  vector<int> ans = linearSearch(arr, target);

  cout << "Element found at index: [" << ans[0] << " " <<ans[1] <<"]";


  return 0;
}
```

Output:

```
/tmp/cEYzO6qbeH.o
Element found at index: [1 2]
```

4.Rotate 90 clockwise: 21.02.24

```
#include <stdio.h>
#define N 3
void rotate90Clockwise(int arr[N][N]) {
    for (int i = 0; i < N / 2; i++) {
        for (int j = i; j < N - i - 1; j++) {
            int temp = arr[i][j];
```

```
            arr[i][j] = arr[N - 1 - j][i];
            arr[N - 1 - j][i] = arr[N - 1 - i][N - 1 - j];
            arr[N - 1 - i][N - 1 - j] = arr[j][N - 1 - i];
            arr[j][N - 1 - i] = temp;
        }
    }
}

void printMatrix(int arr[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int matrix[N][N] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    printf("Original Matrix:\n");
    printMatrix(matrix);
    rotate90Clockwise(matrix);

    printf("\nMatrix after rotating 90 degrees       clockwise:\n");
    printMatrix(matrix);

    return 0;
}
```

Output:

```
/tmp/4OxkmFvFmu.o
Original Matrix:
1 2 3
4 5 6
7 8 9

Matrix after rotating 90 degrees clockwise:
7 4 1
8 5 2
9 6 3
```

Maximum num of first row: 22.02.24

```
#include <bits/stdc++.h>
using namespace std;
#define R 4
#define C 4
  int rowWithMax1s(bool mat[R][C]) {
    int rowIndex = -1 ;
   int maxCount = 0 ;
   for(int i = 0 ; i < R ; i++){
  int count = 0 ;
for(int j = 0 ; j < C ; j++ ){
 if(mat[i][j] == 1){

              count++ ;

           }

        }

        if(count > maxCount){
        maxCount = count ;
        rowIndex = i ;

        }

    }



    return rowIndex ;
}
 int main()
{

    bool mat[R][C] = { {0, 0, 0, 1},
                       {0, 1, 1, 1},
                        {1, 1, 1, 1},
                        {0, 0, 0, 0}};


    cout << "Index of row with maximum 1s is " << rowWithMax1s(mat);
```

```
    return 0;
}
```

Output :

```
/tmp/hDSK1R4ltc.o
Index of row with maximum 1s is 2
```

Left rotate Matrix by 2 time : 23.02.24

```
#include <stdio.h>

void leftRotateMatrix(int mat[][3], int rows, int cols, int k) {
    k = k % cols; // Adjust k in case it's greater than the number of columns

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {

            int newCol = (j + k) % cols;

                        printf("%d ", mat[i][newCol]);
        }
        printf("\n");
    }
}

int main() {
    int rows = 3, cols = 3;
    int matrix[3][3] = {{1, 2, 3},
                        {4, 5, 6},
                        {7, 8, 9}};
    int k = 2; // Number of rotations

    printf("Original Matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
```

```
    printf("\nMatrix after left rotation %d times:\n", k);
    leftRotateMatrix(matrix, rows, cols, k);

    return 0;
}
```

Output:

```
/tmp/hDSK1R4ltc.o
Original Matrix:
1 2 3
4 5 6
7 8 9

Matrix after left rotation 2 times:
3 1 2
6 4 5
9 7 8
```

```
#include <stdio.h>
void printDiagonalPattern(int mat[][3], int rows, int cols) {
    for (int k = 0; k < rows + cols - 1; k++) {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (i + j == k) {
                    printf("%d ", mat[i][j]);
                }
            }
        }
    }
}

int main() {
    int rows = 3, cols = 3;
    int matrix[3][3] = {{1, 2, 3},
                        {4, 5, 6},
                        {7, 8, 9}};

    printf("Original Matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
```

```
    }

    printf("\nMatrix in Diagonal Pattern:\n");
    printDiagonalPattern(matrix, rows, cols);

    return 0;
}
```

Output:

```
/tmp/AjNjAPABxh.o
Original Matrix:
1 2 3
4 5 6
7 8 9

Matrix in Diagonal Pattern:
1 2 4 3 5 7 6 8 9
```

:

7. Set

```
#include <iostream>
#include <vector>

using namespace std;

void setZeroes(vector<vector<int>>& matrix) {
    int m = matrix.size();
    int n = matrix[0].size();

    vector<int> rows(m, 0);
    vector<int> cols(n, 0);
        for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (matrix[i][j] == 0) {
                rows[i] = 1;
                cols[j] = 1;
            }
        }
    }

        for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (rows[i] || cols[j]) {
                matrix[i][j] = 0;
            }
        }
```

```
    }
}

int main() {
        vector<vector<int>> matrix = {{1, 2, 3}, {4, 0, 6}, {7, 8, 9}};



    setZeroes(matrix);
    for (const auto& row : matrix) {
        for (int value : row) {
            cout << value << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
/tmp/LSnd49QkwH.o
1 0 3
0 0 0
7 0 9
```