**DATA SUTRUCTURES AND ALGORITHMS**

**TASK 1:**

**CODE:**

#include<iostream>

#include<cstdlib>

#include<ctime>

using namespace std;

class Grayscale {

int num;

int \*\*ptr = nullptr;

public:

Grayscale()

{

ptr = new int\*[5];

for (int i = 0; i < 5; i++)

{

ptr[i] = new int[5];

}

}

void initializeGrayScale()

{

srand(time(0));

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

ptr[i][j] = rand() % 256 + 1;

}

}

}

void displayGS()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

cout << ptr[i][j] << " ";

}

cout << endl;

}

}

void transposeGS()

{

int \*\*transpose = nullptr;

transpose = new int\*[5];

for (int i = 0; i < 5; i++)

{

transpose[i] = new int[5];

for (int j = 0; j < 5; j++)

{

transpose[i][j]=ptr[i][j];

}

}

//rotating

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

ptr[j][4 - i] = transpose[i][j];

}

}

}

//destructor

~Grayscale()

{

for (int i = 0; i < 5; i++)

{

delete[] ptr[i];

}

delete[] ptr;

}

};

class RGB {

int\*\*\* ptrRGB = nullptr;

public:

//RGB constructor

RGB()

{

ptrRGB = new int\*\*[5];

for (int i = 0; i < 5; i++)

{

ptrRGB[i] = new int\*[5];

for (int j = 0; j < 5; j++)

{

ptrRGB[i][j] = new int[3];

}

}

}

//initializing RGB array

void initializeRGB()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

ptrRGB[i][j][0] = rand() % 256 + 1;

ptrRGB[i][j][1] = rand() % 256 + 1;

ptrRGB[i][j][2] = rand() % 256 + 1;

}

}

}

//displaying RGB

void displayRGB()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

cout <<"("<< ptrRGB[i][j][0] << ","<<ptrRGB[i][j][1]<<","<<ptrRGB[i][j][2]<<"), ";

}

cout << endl;

}

}

void transposeRGB()

{

int \*\*\*transpose = nullptr;

transpose = new int\*\*[5];

for (int i = 0; i < 5; i++)

{

transpose[i] = new int\*[5];

for (int j = 0; j < 5; j++)

{

transpose[i][j] = new int[3];

transpose[i][j][0] = ptrRGB[i][j][0];

transpose[i][j][1] = ptrRGB[i][j][1];

transpose[i][j][2] = ptrRGB[i][j][2];

}

}

//rotating

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

ptrRGB[j][4 - i][0] = transpose[i][j][0];

ptrRGB[j][4 - i][1] = transpose[i][j][1];

ptrRGB[j][4 - i][2] = transpose[i][j][2];

}

}

}

//destructor

~RGB()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

delete[] ptrRGB[i][j];

}

delete[] ptrRGB[i];

}

delete[] ptrRGB;

}

};

int main()

{

Grayscale img1;

//initializing image GS

img1.initializeGrayScale();

//displaying before rortate

cout << "Displaying GRAYSCALE image before rotate:\n";

img1.displayGS();

//rotating GS

img1.transposeGS();

//display after Rotate

cout << "\nDisplaying GRAYSCALE image after rotate:\n";

img1.displayGS();

RGB img2;

//initialinzing RGB image

img2.initializeRGB();

//displaying before rortate

cout << "\n\nDisplaying RGB image before rotate:\n";

img2.displayRGB();

//rotating RGB

img2.transposeRGB();

//display after Rotate

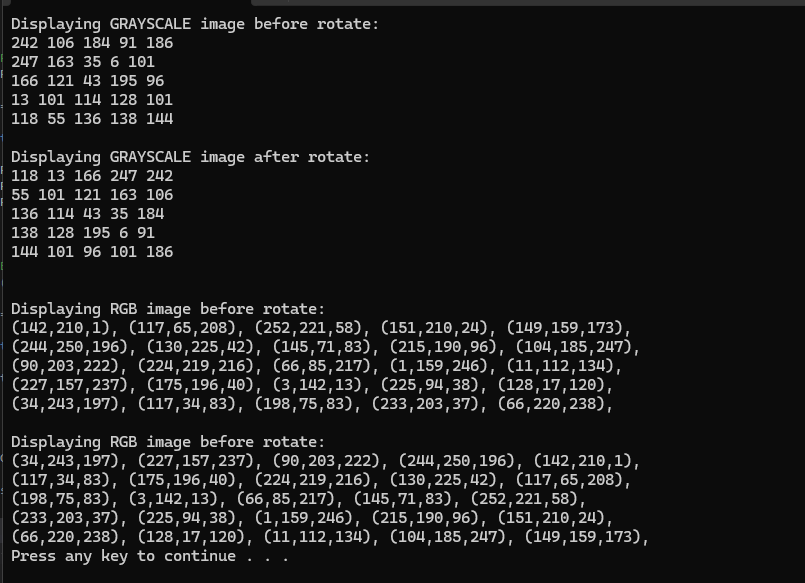
cout << "\nDisplaying RGB image before rotate:\n";

img2.displayRGB();

system("pause");

}

**OUTPUT:**

****

**Task 2:**

**Code:**

#include<iostream>

#include<string>

using namespace std;

class Rubix {

char \*\*\*ptr = nullptr;

//char arr[6];

public:

//constructor

Rubix()

{

ptr = new char\*\*[6];

//cout << "plz Enter character of Each colour on U want on Cube: ";

for (int i = 0; i < 6; i++)

{

//cin >> arr[i];

ptr[i] = new char\*[3];

for (int j = 0; j < 3; j++)

{

ptr[i][j] = new char[3];

}

}

}

//initializing Rubix

void initialize()

{

for (int i = 0; i < 6; i++)

{

cout << "\nEnter single character of color name in face " << i+1 << ": \n";

for (int j = 0; j < 3; j++)

{

cin >> ptr[i][j][0];

cin >> ptr[i][j][1];

cin >> ptr[i][j][2];

}

}

}

//display

void display ()

{

for (int i = 0; i < 6; i++)

{

cout << "\n\nFace " << i+1 << ": \n";

for (int j = 0; j < 3; j++)

{

cout <<""<< ptr[i][j][0] << " "<<ptr[i][j][1]<<" "<<ptr[i][j][2]<<" ";

cout << endl;

}

cout << endl;

}

}

//analyze cube

void analyze()

{

int count[6];

int c = 0;

for (int i = 0; i < 6; i++)

{

c = 1;

for (int j = 0; j < 3; j++)

{

for (int k = 0; k < 2; k++)

{

if (ptr[i][j][k] != ptr[i][j][k + 1])

c++;

}

}

count[i] = c;

}

for (int i = 0; i < 6; i++)

{

int flag = 0;

for (int j = 0; j < 6; j++)

{

if (count[i] <= count[j])

flag++;

}

if (flag == 5)

cout << "\nface " << i + 1 << " is best among all faces.";

for (int j = 0; j < 6; j++)

{

flag = 0;

if (count[i] >= count[j])

flag++;

}

if (flag == 5)

cout << "\nface " << i + 1 << " is worst among all faces.";

}

}

//destructor

~Rubix()

{

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 3; j++)

{

delete[] ptr[i][j];

}

delete[] ptr[i];

}

delete[] ptr;

}

};

int main()

{

Rubix img;

img.initialize();

//displaying user input

system("cls");

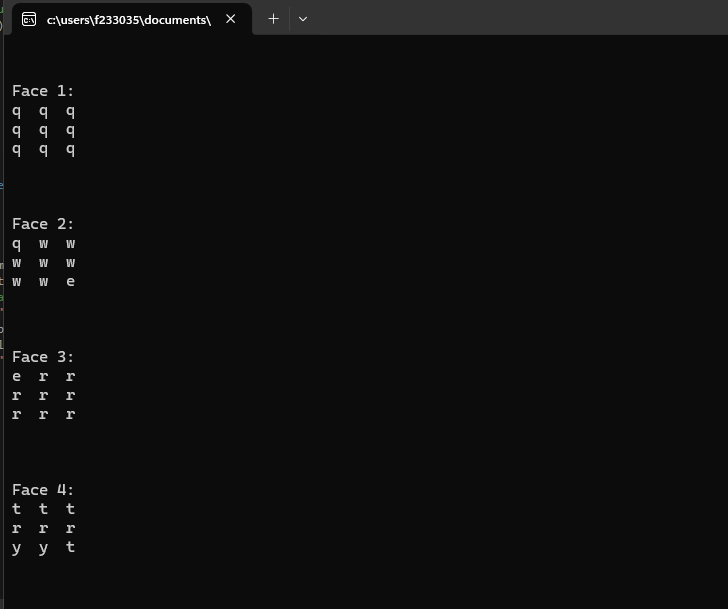
img.display();

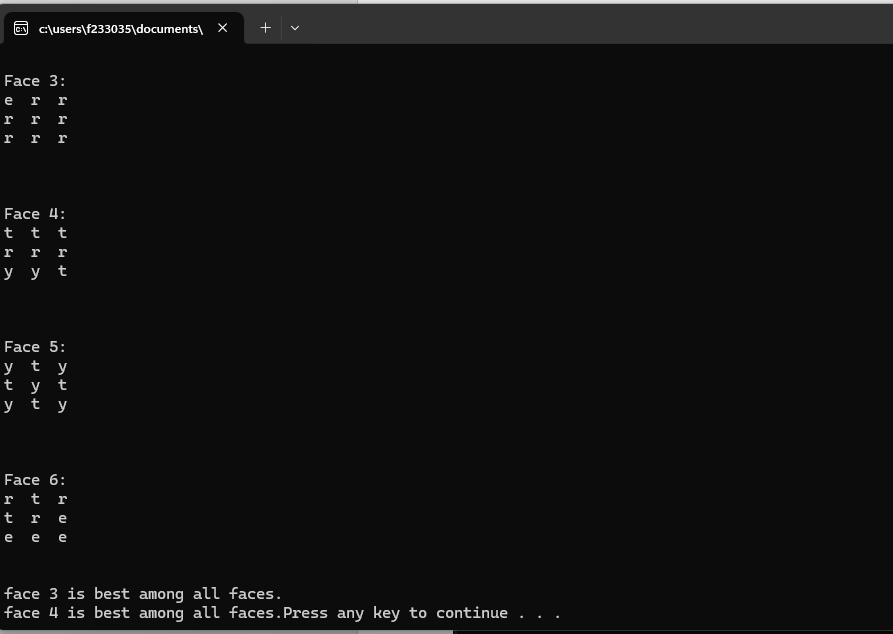
img.analyze();

system("pause");

}

**Output:**

****

****