ASSIGNMENT 3

***QUESTION 1***

CODE

#include <iostream>

#include <string>

using namespace std;

class Medicine

{

private:

    string name;

    string formula;

    int rPrice;

    string builtDate;

    string expDate;

public:

    Medicine(string name, string formula, int rPrice, string builtDate, string expDate)

    {

        this->name = name;

        this->formula = formula;

        this->rPrice = rPrice;

        this->builtDate = builtDate;

        this->expDate = expDate;

    }

    virtual void display()

    {

        cout << "Medicine Name: " << name << endl;

        cout << "Medicine Formula: " << formula << endl;

        cout << "Retail Price: " << rPrice << endl;

        cout << "Built Date: " << builtDate << endl;

        cout << "Expiry Date: " << expDate << endl;

    }

    bool operator==(Medicine \*other)

    {

        for (int i = expDate.length() - 1; i >= expDate.length() - 4; --i)

        {

            if (other->expDate[i] != expDate[i])

            {

                return false;

            }

        }

        return true;

    }

    string

    GetName() const

    {

        return name;

    }

    void SetName(string name)

    {

        this->name = name;

    }

    string GetFormula()

    {

        return formula;

    }

    void SetFormula(string formula)

    {

        this->formula = formula;

    }

    int GetRPrice() const

    {

        return rPrice;

    }

    void SetRPrice(int rPrice)

    {

        this->rPrice = rPrice;

    }

    string GetBuiltDate() const

    {

        return builtDate;

    }

    void SetBuiltDate(string builtDate)

    {

        this->builtDate = builtDate;

    }

    string GetExpDate() const

    {

        return expDate;

    }

    void SetExpDate(string expDate)

    {

        this->expDate = expDate;

    }

};

class Tablet : public Medicine

{

private:

    double sucroseLevel; // value from 0 to 1

public:

    Tablet(string name, string formula, int rPrice, string builtDate, string expDate, double \_sucroseLevel) : Medicine(name, formula, rPrice, builtDate, expDate), sucroseLevel(\_sucroseLevel) {}

    void display()

    {

        cout << "Medicine Name: " << GetName()

             << endl;

        cout << "Medicine Formula: " << GetFormula()

             << endl;

        cout << "Retail Price: " << GetRPrice()

             << endl;

        cout << "Built Date: " << GetBuiltDate()

             << endl;

        cout << "Expiry Date: " << GetExpDate()

             << endl;

        cout << "Sucrose Level: " << sucroseLevel << endl;

    }

    double GetSucroseLevel() const

    {

        return sucroseLevel;

    }

    void SetSucroseLevel(double sucroseLevel)

    {

        this->sucroseLevel = sucroseLevel;

    }

};

class Capsule : public Medicine

{

private:

    int absorption; // 1 to 100 percentage

public:

    int GetAbsorption() const

    {

        return absorption;

    }

    void SetAbsorption(int absorption)

    {

        this->absorption = absorption;

    }

    Capsule(string name, string formula, int rPrice, string builtDate, string expDate, int \_absorption) : Medicine(name, formula, rPrice, builtDate, expDate), absorption(\_absorption) {}

    void display()

    {

        cout << "Medicine Name: " << GetName()

             << endl;

        cout << "Medicine Formula: " << GetFormula()

             << endl;

        cout << "Retail Price: " << GetRPrice()

             << endl;

        cout << "Built Date: " << GetBuiltDate()

             << endl;

        cout << "Expiry Date: " << GetExpDate()

             << endl;

        cout << "Absorption Percentage: " << absorption << endl;

    }

};

class Syrup : public Medicine

{

private:

    int quantity; // in ml

public:

    Syrup(string name, string formula, int rPrice, string builtDate, string expDate, int \_quantity) : Medicine(name, formula, rPrice, builtDate, expDate), quantity(\_quantity) {}

    void display()

    {

        cout << "Medicine Name: " << GetName()

             << endl;

        cout << "Medicine Formula: " << GetFormula()

             << endl;

        cout << "Retail Price: " << GetRPrice()

             << endl;

        cout << "Built Date: " << GetBuiltDate()

             << endl;

        cout << "Expiry Date: " << GetExpDate()

             << endl;

        cout << "Quantity: " << quantity << endl;

    };

    int GetQuantity() const

    {

        return quantity;

    }

    void SetQuantity(int quantity)

    {

        this->quantity = quantity;

    }

};

class Shop

{

private:

    Medicine \*medicine;

    int totalMedicine;

public:

    Shop()

    {

        Medicine med1("med", "ethane", 25, "15 04 2022", "15 04 2024");

        Tablet tab1("Panadol", "methane", 65, "15 04 2023", "15 04 2024", 0.65);

        Capsule cap1("Panadol", "butane", 90, "15 04 2020", "15 04 2022", 43);

        Syrup syr1("syrup 1", "octane", 200, "15 04 2021", "15 04 2026", 100);

        Medicine \*medicines[] = {&med1, &tab1, &cap1, &syr1};

        totalMedicine = 4;

    }

    friend class Counter;

    friend class Pharmicist;

};

class Pharmicist

{

private:

    Medicine \*medicine;

    int totalMedicine;

public:

    Pharmicist(Shop &s) : medicine(s.medicine), totalMedicine(s.totalMedicine) {}

    bool Search\_Medicine(string \_formula, string &Name, int &ind)

    {

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

        {

            cout << "loop called " << i+1 << endl;

            medicine[i].display();

            string temp=medicine[i].GetFormula();

            if (\_formula == temp){

                cout << "condition true " << i << endl;

                Name = medicine[i].GetName();

                ind = i;

                cout << "condition true " << i << endl;

                return true;

            }

            cout << "loop called " << i << endl;

        }

        return false;

    };

};

class Counter

{

private:

    Medicine \*medicine;

    int totalMedicine;

    int totalRevenue;

    string medicineName;

    int index;

public:

    Counter(Shop &s) : medicine(s.medicine), totalMedicine(s.totalMedicine) {}

    void removeMedcineFromList()

    {

        for (int i = index; i < totalMedicine; ++i)

        {

            medicine[i] = medicine[i + 1];

        }

        --totalMedicine;

    }

    void Search\_Medicine(Pharmicist &p)

    {

        string \_formula;

        cout << "Enter the formula of the medicine: ";

        cin >> \_formula;

        if (p.Search\_Medicine(\_formula, medicineName, index) == 1)

        {

            cout << "Medicine Available" << endl;

            medicine[index].display();

            removeMedcineFromList();

            updateRevenue();

        }

        else

        {

            cout << "Medicine not Available" << endl;

        }

    }

    void updateRevenue()

    {

        int temp = 0;

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

        {

            temp += medicine[i].GetRPrice();

        }

        totalRevenue = temp;

    }

};

QUESTION 2

CODE

#include <iostream>

using namespace std;

template <class T>

class Pet{

    private:

    T name;

    int age;

    public:

    Pet(T name, int age) : name(name), age(age){}

    virtual void makeSound() = 0;

};

template <class T>

class Cat : public Pet<T> {

    public:

    Cat(T name, int age) : Pet<T>(name, age) {}

    void makeSound() override{

        cout << "Meow" << endl;

    }

};

template <class T>

class Dog : public Pet<T> {

    private:

    public:

    Dog(T name, int age ) : Pet<T>(name, age) {}

    void makeSound() override {

        cout << "Woof" << endl;

    }

};

template <class T>

class Bird : public Pet<T> {

    private:

    public:

    Bird(T name, int age ) : Pet<T>(name, age) {}

    void makeSound() override {

        cout << "Chirp" << endl;

    }

};

int main(){

    cout << "23k-0800 Muhammad Mufeez" << endl;

    Cat<string> cat("cat", 5);

    Dog<string> dog("dog", 12);

    Bird<string> bird("bird", 2);

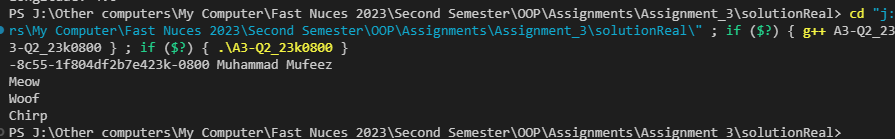
    cat.makeSound();

    dog.makeSound();

    bird.makeSound();

}

**OUTPUT**



***QUESTION 3***

CODE

#include <iostream>

using namespace std;

template <class T>

class Matrix

{

protected:

    T \*\*arr;

    int rows;

    int cols;

public:

    Matrix();

    Matrix(int \_rows, int \_cols)

    {

        this->rows = \_rows;

        this->cols = \_cols;

        arr = new T \*[\_rows];

        for (int i = 0; i < \_rows; ++i)

        {

            arr[i] = new T[\_cols]();

        }

        for(int i=0;i<rows;++i){

            for(int j=0;j<cols;++j){

                arr[i][j]=0;

            }

        }

    }

    T getElement(int i, int j)

    {

        while (i >= rows || i < 0)

        {

            cout << "Enter Correct Row Number" << endl;

            cin >> i;

        }

        while (j >= cols || j < 0)

        {

            cout << "Enter Correct Column Number" << endl;

            cin >> j;

        }

        return arr[i][j];

    }

    void setElement(int i, int j, T val)

    {

        while (i >= rows || i < 0)

        {

            cout << "Enter Correct Row Number" << endl;

            cin >> i;

        }

        while (j >= cols || j < 0)

        {

            cout << "Enter Correct Column Number" << endl;

            cin >> j;

        }

        arr[i][j] = val;

    }

    void setMatrix()

    {

        T temp;

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

        {

            cout << "Row: " << i+1 << endl;

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

            {

                cin >> arr[i][j];

        cin.ignore();

            }

        }

    };

    virtual void display()

    {

        cout << "Displaying General Matrix" << endl;

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

        {

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

            {

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

            }

            cout << endl;

        }

    }

    ~Matrix()

    {

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

        {

            delete[] arr[i];

        };

        delete[] arr;

    }

    Matrix<T> operator+(Matrix<T> &other)

    {

        // considering that rows and columns of both matrices are same

        Matrix<T> temp(rows, cols);

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

        {

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

            {

                temp.arr[i][j] = arr[i][j] + other.arr[i][j];

            }

        }

        return temp;

    };

    Matrix<T> operator-(Matrix<T> &other)

    {

        // considering that rows and columns of both matrices are same

        Matrix<T> temp(rows, cols);

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

        {

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

            {

                temp.arr[i][j] = arr[i][j] - other.arr[i][j];

            }

        }

        return temp;

    };

    Matrix<T> operator\*(Matrix<T> &other)

    {

        Matrix<T> temp(rows, other.cols);

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

            for (int j = 0; j < other.cols; ++j)

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

                {

                    temp.arr[i][j] += arr[i][k] \* other.arr[k][j];

                }

        return temp;

    };

};

class IntMatrix : public Matrix<int>

{

public:

    IntMatrix() {}

    IntMatrix(int rows, int cols) : Matrix<int>::Matrix(rows, cols) {}

    void display()

    {

        cout << "Displaying Integer Matrix" << endl;

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

        {

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

            {

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

            }

            cout << endl;

        }

    }

};

class DoubleMatrix : public Matrix<double>

{

public:

    DoubleMatrix();

    DoubleMatrix(int rows, int cols) : Matrix<double>::Matrix(rows, cols) {}

    void display()

    {

        cout << "Displaying Double Matrix" << endl;

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

        {

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

            {

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

            }

            cout << endl;

        }

    }

};

int main()

{

    cout << "23k-0800 Muhammad Mufeez" << endl;

    cout<<"int matrix class"<<endl;

    IntMatrix m1(2, 3);

    DoubleMatrix m2(3, 2);

    m1.setMatrix();

    cout<<"double matrix class"<<endl;

    m2.setMatrix();

    m1.display();

    m2.display();

    Matrix<int> mt(3, 3);

    Matrix<int> mt1(3, 3);

    mt.setMatrix();

    mt.display();

    mt1.setMatrix();

    mt1.display();

    Matrix<int> mt2 = mt + mt1;

    cout<<"added"<<endl;

    mt2.display();

    cout<<"subtracted"<<endl;

    Matrix<int> mt3 = mt - mt1;

    mt3.display();

    cout<<"multiplied"<<endl;

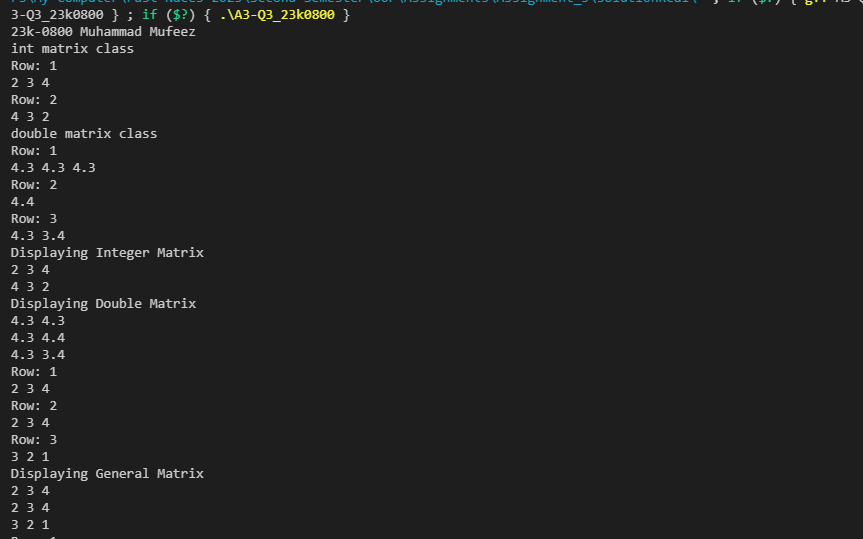
    Matrix<int> mt4 = mt \* mt1;

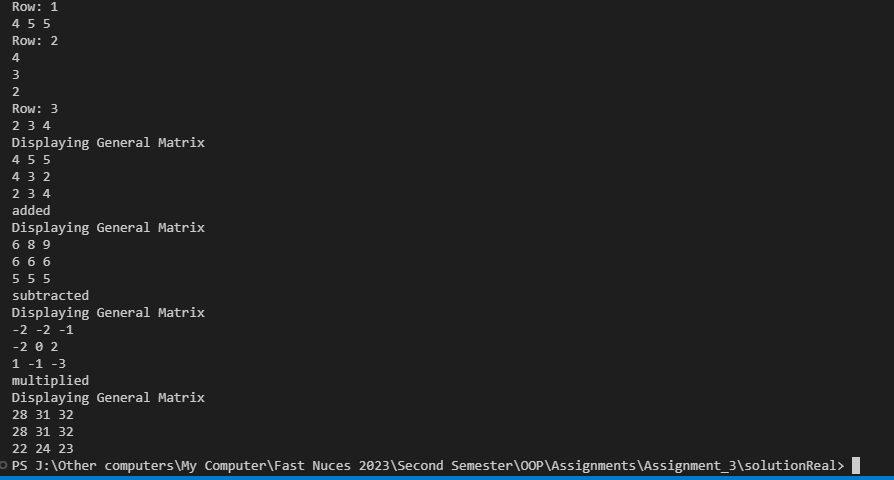
    mt4.display();

    return 0;

}

**OUTPUT**





***QUESTION 4***

CODE

#include <iostream>

#include<cmath>

using namespace std;

class Drone

{

private:

    float latitude;

    float longitude;

    float altitude;

    float speed;

public:

    Drone(float latitude, float longitude, float altitude, float speed)

        : latitude(latitude), longitude(longitude), altitude(altitude), speed(speed)

    {

    }

    float getLatitude() const { return latitude; }

    float getLongitude() const { return longitude; }

    float getAltitude() const { return altitude; }

    float getSpeed() const { return speed; }

    void setLatitude(float latitude) { this->latitude = latitude; }

    void setLongitude(float longitude) { this->longitude = longitude; }

    void setAltitude(float altitude) { this->altitude = altitude; }

    virtual void adjustAltitude(float meters) = 0;

    virtual void setSpeed(float speed){this->speed=speed;};

};

class Flyable : virtual public Drone

{

public:

Flyable(float latitude, float longitude, float altitude, float speed):Drone(latitude, longitude, altitude, speed){}

    virtual void takeoff() = 0;

    virtual void land() = 0;

    virtual void navigateTo(float latitude, float longitude, float altitude) = 0;

};

class Scannable : virtual public Drone

{

public:

Scannable(float latitude, float longitude, float altitude, float speed):Drone(latitude, longitude, altitude, speed){}

    virtual void scanArea(float radius) = 0;

};

class ReconDrone : public Flyable, public Scannable

{

private:

    int cameraResol;

    int maxFlightTime;

public:

    ReconDrone(float latitude, float longitude, float altitude, float speed, int cameraResol)

        : Flyable(latitude, longitude, altitude, speed),Scannable(latitude, longitude, altitude, speed),Drone(latitude, longitude, altitude, speed), cameraResol(cameraResol), maxFlightTime(0)

    {

    }

    void setCameraResol(int cameraResol) { this->cameraResol = cameraResol; }

    void adjustAltitude(float meters) override

    {

        setAltitude(getAltitude()+meters);

    }

    void takeoff() override

    {

        cout<<"Taking off "<<endl;

    }

   void navigateTo(float latitude, float longitude,float altitude) override

    {

        float a=pow(latitude-getLatitude(),2)+pow(longitude-getLongitude(),2)+pow(altitude-getAltitude(),2);

        float distance=sqrt(a);

        cout<<"Distance is: "<<distance<<endl;

        cout<<"Estimated time is: "<<distance/getSpeed()<<endl;

        scanArea(distance);

    }

    void land()

    {

        cout<<"Landing now "<<endl;

    }

    void scanArea(float radius) override

    {

        int objects;

        objects=3;

        // objects=0;

        int i;

        try

        {

            if(objects==0)

            {

                throw objects;

            }

            for(i=1;i<=objects;i++)

            {

                cout<<"Object number: "<<i+1<<endl;

                setLatitude(i+1.6);

                cout<<"Latitude: "<<getLatitude()<<endl;

                setLongitude(i+1.6);

                cout<<"Longitude: "<<getLongitude()<<endl;

            }

        }

        catch(int objects)

        {

            cout<<"failure."<<endl;

        }

    };

};

int main()

{

    cout << "23k-0800 Muhammad Mufeez" << endl;

    ReconDrone r1(34.4,43.4,434.3,43,540);

    r1.navigateTo(55,55,55);

    return 0;

};

**OUTPUT**

