Question 1:

Build a code to handle a situation where the user enters a non-integer value for either the numerator or denominator. If the user enters a non-integer value, the program should catch this error and display the message "Invalid input: Please enter valid integer values for numerator and denominator." Additionally, make sure to update the input and output formats accordingly.

Exception Type

`runtime_error& e`

Input Format

- Enter an integer numerator: [integer value]
- Enter an integer denominator: [integer value]

Output Format

- If the user enters valid integer values for both numerator and denominator and the denominator is not zero, display the result of division.
- If the user enters a non-integer value for either the numerator or denominator, display the message: "Invalid input: Please enter valid integer values for numerator and denominator."
- If the user enters a denominator of zero, display the message: "Error: Division by zero is not allowed."
- After handling the input and errors, always display: "Program finished execution."

Title for Question 1: Arithmetic Exception

Solution:

```
#include <iostream>
#include <stdexcept>

using namespace std;

int main() {
    int numerator, denominator;

    // cout << "Enter numerator: ";
    cin >> numerator;

    // cout << "Enter denominator: ";
    cin >> denominator;

    try {
        if (denominator == 0) {
            throw runtime_error("Division by zero is not allowed.");
        }
        int result = numerator / denominator;
        cout << "Result of division: " << result << endl;
    } catch (const runtime_error& e) {
        cout << "Error: " << e.what() << endl;
    }

    cout << "Program finished execution." << endl; // This line added

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	10 2	Result of division: 5 Program finished execution.
2	50 0	Error: Division by zero is not allowed. Program finished execution.
3	75 3	Result of division: 25 Program finished execution.
4	125 0	Error: Division by zero is not allowed. Program finished execution.
5	81 9	Result of division: 9 Program finished execution.
6	100 0	Error: Division by zero is not allowed. Program finished execution.

White List:

Black List:

Question 2:

Suppose you have a modified version of the code that allows the user to enter multiple names and ages in a loop until they decide to stop. Each name and age pair is separated by a space, and the user can enter as many pairs as they want. The program will then display the names and ages entered. If any input error occurs, such as entering a non-numeric age or an age outside the valid range, it should be handled appropriately. Write a modified version of the code that accomplishes this task, and provide a sample input sequence along with the expected output for a scenario where the user enters three name and age pairs correctly and one pair with an invalid age.

Exception Type

```
runtime_error& e
out_of_range& e
```

Input Format

- The program expects the user to input the following:
- A string representing the name.
- An integer representing the age.

Output Format

- The program will produce the following outputs:
- If the name and age are valid (name is any string, age is an integer between 0 and 120):
- It will print the name and age in the format: "{name} age is {age}".
- Example output: "John age is 30"
- If the input is invalid:
- If the age is not a valid integer or is out of range, it will print an error message in the format: "Invalid input. Please enter a numeric value for age." or "Invalid age. Please enter a valid age between 0 and 120."
- Example error output for an invalid age: "Invalid input. Please enter a numeric value for age."

Title for Question 2: Input Mismatch Exception

Solution:

```
#include <iostream>
#include <stdexcept>
#include <limits>
using namespace std;
int main() {
    string name;
    int age;

    //std::cout << "Please enter your age: ";
    try {
       cin >> name;
        if (!(std::cin >> age)) {
            throw std::runtime_error("Invalid input. Please enter a numer
        }

        if (age < 0 || age > 120) {
            throw std::out_of_range("Invalid age. Please enter a valid ag
        }

        std::cout << name  << " age is " <<age << std::endl;
    } catch (const std::runtime_error& e) {
        std::cerr << e.what() << std::endl;
    } catch (const std::out_of_range& e) {
        std::cerr << e.what() << std::endl;
    }
    // Clear any remaining input from the stream
    std::cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');


    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	Alice 25	Alice age is 25

S.No	Inputs	Outputs
2	Bob -5	Invalid age. Please enter a valid age between 0 and 120.
3	Carol 150	Invalid age. Please enter a valid age between 0 and 120.
4	Dave Abc	Invalid input. Please enter a numeric value for age.
5	Eve 120	Eve age is 120
6	“” 30	“” age is 30

White List:

Black List:

Question 3:

Imagine you have an application that allows users to register their information, including a Register Number and a Mobile Number. Users input their data as follows:

- First, the program prompts the user to enter their Register Number.
- Then, the program prompts the user to enter their Mobile Number.
- You have been tasked with ensuring that the input validation works correctly for various scenarios.

Exception Type:

```
IllegalArgumentException& e
NumberFormatException& e
NoSuchElementException& e
```

Input Format:

- The program expects the user to input the Register Number and Mobile Number.

Output Format:

- If the provided Register Number and Mobile Number are valid, the program will output "Valid."
- If any of the exceptions are thrown during input validation, it will print an error message corresponding to the specific exception type.
- For IllegalArgumentException, it will output "IllegalArgumentException: Invalid input format."
- For NumberFormatException, it will output "NumberFormatException: Mobile number should contain only digits."
- For NoSuchElementException, it will output "NoSuchElementException: Register number should contain only digits and alphabets."

Title for Question 3: Number Format & Illegal Argument Exception

Solution:

```
#include <iostream>
#include <string>
#include <stdexcept>

// Custom exception classes
class IllegalArgumentException : public std::exception {
public:
    const char* what() const throw() {
        return "IllegalArgumentException: Invalid input format.";
    }
};

class NumberFormatException : public std::exception {
public:
    const char* what() const throw() {
        return "NumberFormatException: Mobile number should contain only
    }
};

class NoSuchElementException : public std::exception {
public:
    const char* what() const throw() {
        return "NoSuchElementException: Register number should contain on
    }
};

// Function to validate the Register Number
bool isValidRegisterNumber(const std::string& regNumber) {
    if (regNumber.length() != 9)
        throw IllegalArgumentException();

    for (char c : regNumber) {
        if (!std::isalnum(c))
            throw NoSuchElementException();
    }

    return true;
}

// Function to validate the Mobile Number
bool isValidMobileNumber(const std::string& mobileNumber) {
    if (mobileNumber.length() != 10)
        throw IllegalArgumentException();

    for (char c : mobileNumber) {
        if (!std::isdigit(c))
            throw NumberFormatException();
    }

    return true;
}

int main() {
    std::string regNumber, mobileNumber;

    try {
      //  std::cout << "Enter Register Number: ";
        std::cin >> regNumber;
        isValidRegisterNumber(regNumber);
```

```
       // std::cout << "Enter Mobile Number: ";
        std::cin >> mobileNumber;
        isValidMobileNumber(mobileNumber);

        std::cout << "Valid" << std::endl;
    } catch (const IllegalArgumentException& e) {
        std::cout<< e.what() << std::endl;
    } catch (const NumberFormatException& e) {
        std::cout<< e.what() << std::endl;
    } catch (const NoSuchElementException& e) {
        std::cout<< e.what() << std::endl;
    }

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	ABC123456 1234567890	Valid
2	AB@C12345 1234567890	NoSuchElementException: Register number should contain only digits and alphabets.
3	727410 1234567890	IllegalArgumentException: Invalid input format.
4	ABC123456 1234X67890	NumberFormatException: Mobile number should contain only digits.
5	ABC12345 12345	IllegalArgumentException: Invalid input format.
6	CSE123456 9876543210	Valid

White List:

Black List:

Question 4:

Write a code, we have a base class Shape with two pure virtual functions area() and perimeter(). The derived class Circle inherits from Shape and provides concrete implementations for these functions. The program prompts the user for input and calculates the area and perimeter of a circle based on the user-provided radius.

Virtual Function Header:

```
virtual double area() const = 0;
virtual double perimeter() const = 0;
```

Input Format:

- The program expects the user to input the radius of a circle as a floating-point number.

Output Format:

- The program calculates and displays the area and perimeter of the circle, each followed by a newline character.

Title for Question 4: Pure virtual functions area() and perimeter()

Solution:

```
#include <iostream>
#include<cmath>
class Shape {
public:
virtual double area() const = 0; // Pure virtual function
virtual double perimeter() const = 0; // Pure virtual function
};

class Circle : public Shape {
private:
double radius;
public:
Circle(double r) : radius(r) {}
double area() const override {
  return M_PI * radius * radius;
}
double perimeter() const override {
 return 2 * M_PI * radius;
}
};
int main() {
double userRadius;
// Prompt the user for the radius
//std::cout << "Enter the radius of the circle: ";
std::cin >> userRadius;
// Create a Circle object with the user-provided radius
Circle circle(userRadius);
// Calculate and display the area and perimeter of the circle
std::cout << "Area of the circle: " << circle.area() << std::endl;
std::cout << "Perimeter of the circle: " << circle.perimeter() << std::en
return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5.0	Area of the circle: 78.5398 Perimeter of the circle: 31.4159
2	-19.6349	Area of the circle: 1211.18 Perimeter of the circle: -123.37
3	2.5	Area of the circle: 19.635 Perimeter of the circle: 15.708
4	-3.0	Area of the circle: 28.2743 Perimeter of the circle: -18.8496

S.No	Inputs	Outputs
5	7.25	Area of the circle: 165.13 Perimeter of the circle: 45.5531
6	100.5	Area of the circle: 31730.9 Perimeter of the circle: 631.46

White List: virtual double area() const = 0;,virtual double perimeter() const = 0;

Black List:

Question 5:

Write a program that includes a base class with a virtual function and two derived classes that override the virtual function. Allow the user to choose which derived class's function to call at runtime.

Virtual Function Header:

```
virtual void printInfo() const
```

Input Format:

- The user is prompted to choose a derived class by entering a number (1 for Derived1, 2 for Derived2, 0 to exit).

Output Format:

- The program prints the message indicating which class's function was called based on the user's choice.

Title for Question 5: Override virtual function

Solution:

```
#include <iostream>
// Base class with a virtual function
class Base {
public:
virtual void printInfo() const {
 std::cout << "Base class's function called." << std::endl;
}
};
// First derived class
class Derived1 : public Base {
public:
void printInfo() const override {
 std::cout << "Derived1 class's function called." << std::endl;
}
};
// Second derived class
```

```
class Derived2 : public Base {
public:
void printInfo() const override {
 std::cout << "Derived2 class's function called." << std::endl;
}
};

int main() {
int choice;
while (true) {
 //std::cout << "Choose a derived class (1 for Derived1, 2 for Derived2,
 std::cin >> choice;
 if (choice == 0) {
 break; // Exit the program
 }
 Base* obj = nullptr;
 switch (choice) {
 case 1:
 obj = new Derived1();
 break;
 case 2:
 obj = new Derived2();
 break;
 default:
 std::cout << "Invalid choice. Try again." << std::endl;
 continue; // Ask for input again
}
 // Call the virtual function of the selected derived class
 obj->printInfo();
 // Clean up allocated memory
 delete obj;
}
return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	1 0	Derived1 class's function called.
2	3 0	Invalid choice. Try again.
3	2 1 0	Derived2 class's function called. Derived1 class's function called.
4	2 0	Derived2 class's function called.
5	5 1 0	Invalid choice. Try again. Derived1 class's function called.
6	6 0	Invalid choice. Try again.

White List: virtual void printInfo() const

Black List:

Question 6:

Implement a C++ program that validates email addresses using regular expressions. Utilize the regex library and prompt the user to enter an email address.In the case of an invalid email, use a try-catch block to handle exceptions and display a corresponding error message.

Input Format:

- The program prompts the user to enter an email address.

Output Format:

- If the user enters a valid email address, the program outputs: "Valid email address: [emailAddress]."
- If the user enters an invalid email address, the program uses a try-catch block to catch a runtime_error exception and displays the error message: "Invalid email address. Please enter a valid email."

Title for Question 6: Valid Email Exception

Solution:

```
#include <iostream>
#include <string>
#include <regex>
#include <stdexcept>
using namespace std; // Add this line to use the std namespace
int main() {
string emailAddress;
//cout << "Enter an email address: ";
cin >> emailAddress;
try {
// Define a regular expression pattern to validate email addresses
regex pattern(R"(\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\b)");
if (regex_match(emailAddress, pattern)) {
cout << "Valid email address: " << emailAddress << endl;
} else {
throw runtime_error("Invalid email address. Please enter a valid email.")
}
} catch (const runtime_error& e) {
cout << e.what() << endl;
}
return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	email@address.org	Valid email address: email@address.org
2	Invalid.email@com	Invalid email address. Please enter a valid email.
3	john.doe@example.com	Valid email address: john.doe@example.com
4	invalid_email	Invalid email address. Please enter a valid email.
5	example@email.co.uk	Valid email address: example@email.co.uk
6	user@.com	Invalid email address. Please enter a valid email.

White List:

Black List: