# Assignment Description

For this assignment please complete the following exercises.  In addition to the requirements stated in these exercises for any classes that you create you should also create a class diagram using UML and a use case diagram.

Write a program that converts a number entered in Roman numerals to decimal form. Your program should consist of a class, romanType. An object of romanType should do the following:

1. Store the number as a Roman numeral.
2. Convert and store the number into decimal form.
3. Print the number as a Roman numeral or decimal number as requested by the user. (Write two separate functions—one to print the number as a Roman numeral and the other to print the number as a decimal number.)

The decimal values of the Roman numerals are:

* M 1000
* D 500
* C 100
* L 50
* X 10
* V   5
* I   1

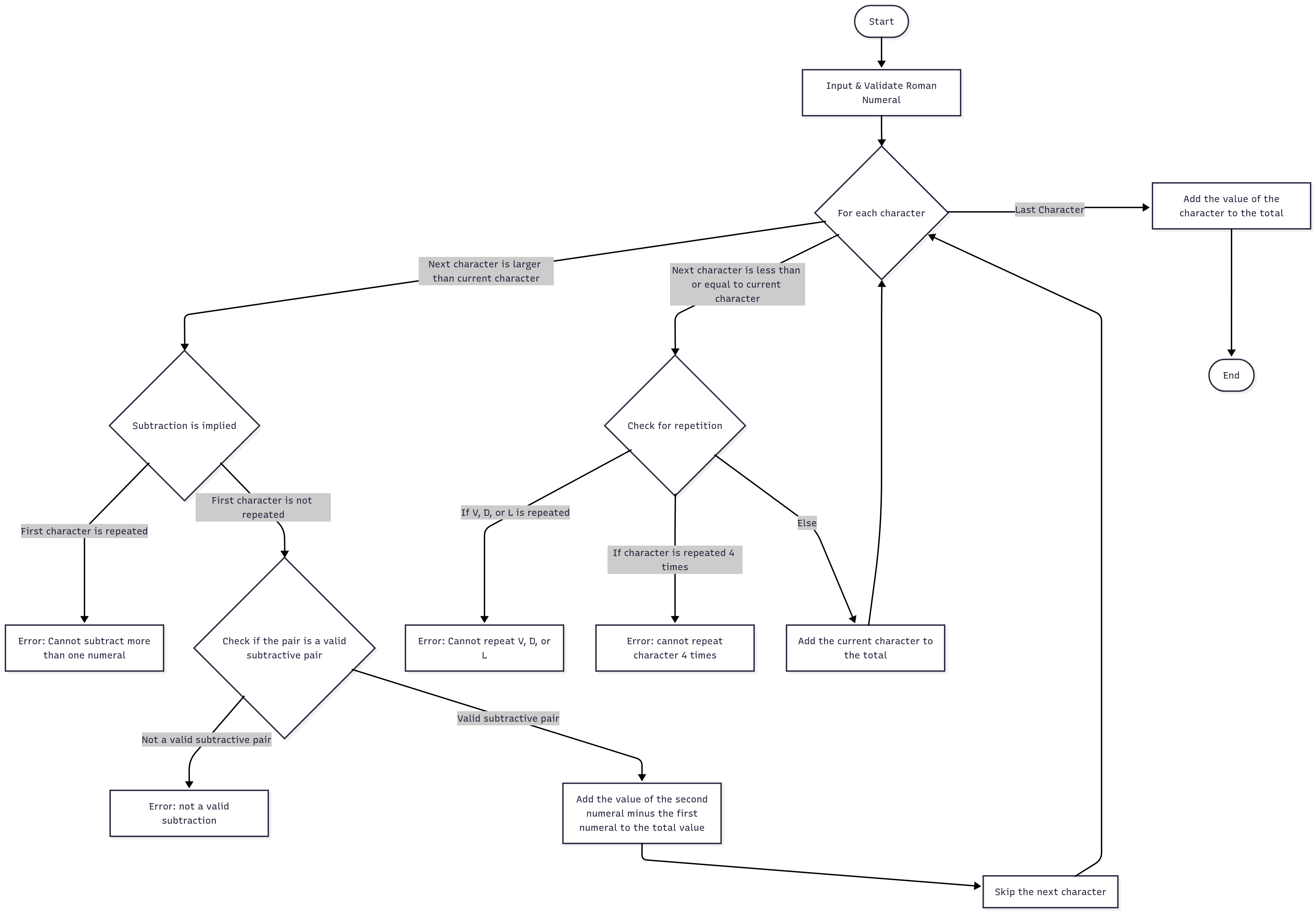
Remember, a larger numeral preceding a smaller numeral means addition, so LX is 60. A smaller numeral preceding a larger numeral means subtrac- tion, so XL is 40. Any place in a decimal number, such as the 1s place, the 10s place, and so on, requires from zero to four Roman numerals.

* Test your program using the following Roman numerals: MCXIV, CCCLIX, and MDCLXVI.

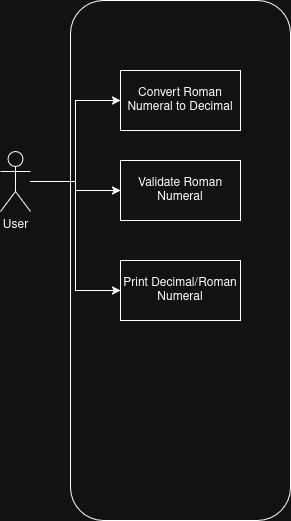
# 1 Readme Documentation

This program will take the input of a roman numeral. In the constructor for the romanType class, it will convert the given roman numeral to a decimal number. If the numeral is invalid it will print an error and the stored decimal will be 0. Otherwise, it will print the numeral in decimal.

# 2 Flowchart Screen Shots



# 3 UML and Use Case Diagrams



# 4 Source Code of All files (.h, .cpp)

#include <iostream>

#include <string>

#include <sstream>

#include <exception>

#include <bits/stdc++.h>

#include <vector>

class romanType{

private:

std::string numeral = "";

int decimal = 0;

std::unordered\_map<char, int> numeralValues = {

{'M', 1000},

{'D', 500},

{'C', 100},

{'L', 50},

{'X', 10},

{'V', 5},

{'I', 1}

};

public:

romanType(std::string numeral){

try{

decimal = numeralToDecimal(numeral);

}

catch(std::exception &e){

std::cout << e.what() << std::endl;

}

}

int numeralToDecimal(std::string numeral){

int decimal = 0;

const std::string validRomanNumerals = "MDCLXVI";

const std::vector<std::string> validSubtractionPairs = {"IV", "IX", "XL", "XC", "CD", "CM"};

// Make sure all characters are roman numerals

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

if(validRomanNumerals.find(numeral[i]) == std::string::npos)

throw std::runtime\_error("Invalid character in roman numeral");

}

int repetitionCounter = 1;

// Check that the order is correct and convert to decimal

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

char currentNumeral = numeral[i];

int currentNumeralValue = numeralValues[currentNumeral];

int lastNumeralIndex = numeral.size() - 1;

if(i == lastNumeralIndex){

decimal += currentNumeralValue;

continue;

}

char nextNumeral = numeral[i + 1];

int nextNumeralValue = numeralValues[nextNumeral];

// In case of subtraction

if(currentNumeralValue < nextNumeralValue){

if(repetitionCounter > 1) throw std::runtime\_error("Cannot subtract more than one numeral");

// Check if the numeral pair is a valid subtractive pair

std::string numeralPair(1, currentNumeral);

numeralPair += nextNumeral;

bool found = false;

for(auto it = validSubtractionPairs.begin(); it != validSubtractionPairs.end(); it++){

if(numeralPair == \*it) found = true;

}

if(!found) throw std::runtime\_error("Invalid subtractive pair");

decimal += (nextNumeralValue - currentNumeralValue);

// Adds two rather than one to i to skip the next character in the subtractive pair

i++;

}

else{

decimal += currentNumeralValue;

}

if(currentNumeral == nextNumeral){

repetitionCounter++;

if(currentNumeral == 'V') throw std::runtime\_error("Cannot repeat V");

if(currentNumeral == 'L') throw std::runtime\_error("Cannot repeat L");

if(currentNumeral == 'D') throw std::runtime\_error("Cannot repeat D");

if(repetitionCounter > 3) throw std::runtime\_error("Cannot repeat a numeral more than 3 times");

}

else{

repetitionCounter = 1;

}

}

return decimal;

}

void validateNumeral(std::string numeral){

}

void printNumeral(){

if(numeral == ""){

std::cout << "No numeral stored." << std::endl;

}

else{

std::cout << numeral << std::endl;

}

}

void printDecimal(){

if(decimal == 0){

std::cout << "No decimal number stored." << std::endl;

}

else{

std:: cout << decimal << std::endl;

}

}

};

int main(){

std::string numeral;

std::cout << "Please enter a roman numeral: ";

std::cin >> numeral;

romanType number(numeral);

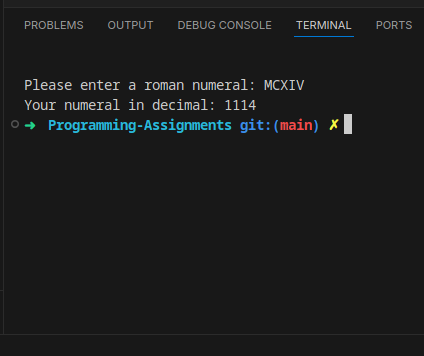
std::cout << "Your numeral in decimal: ";

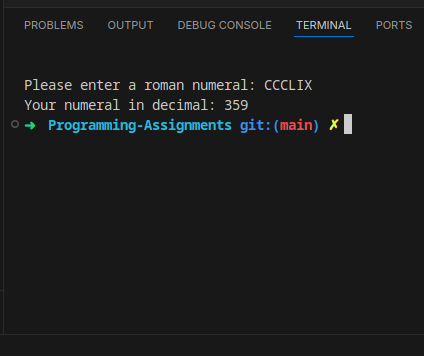
number.printDecimal();

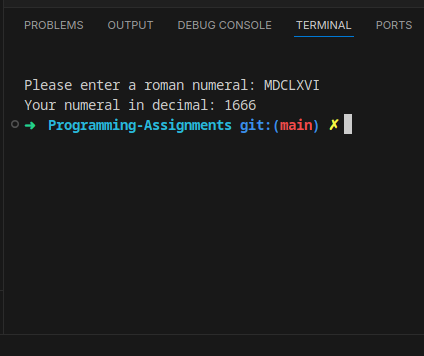
return 0;

}

# 5 Three Use Case Screen Shots







# (Optional) GitHub URL