**Program Development Process**

**Step 1: Requirements specification**

Read the problem to be solved.

State the problem clearly / Understand the problem.

Describe the problem to be solved in your own words here:

The purpose of the program is to create a custom string class that takes in a vector of STL strings. Step 1 asks to test the functions created and Step 2 asks to use the binary cosine coefficient to compare an input to 2 documents using the StringSet class.

**Step 2: Analyze**

Describe the data flow and to identify the inputs, outputs, and constants.

Identify what the output is first, and then figure out what input data you need.

This list will eventually lead to the list of variables and constants to be defined.

Inputs Outputs Constants

String StringSet Size None

Query StringSet Output

Search Term Search Results

Doc1 and Doc2 Binary Cosine Coefficient

**Step 3: Program design**

Describe the process for obtaining the output from the input.

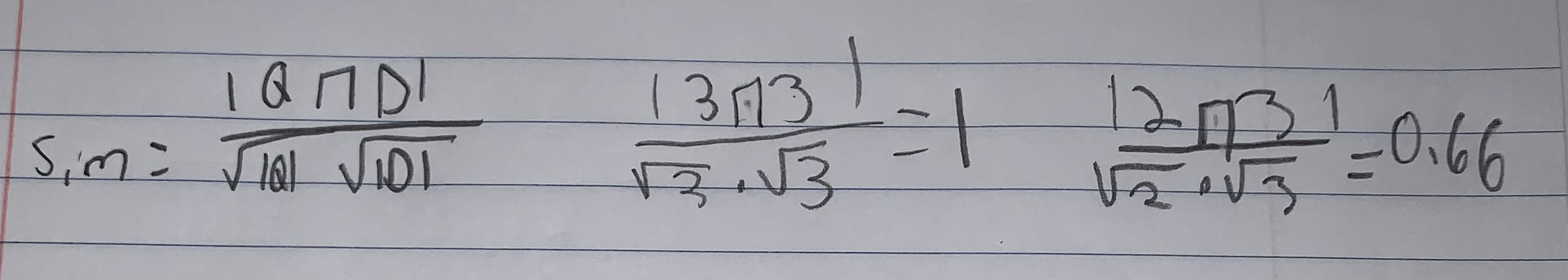
Note the steps you are performing.

This will lead to the C++ statements you write.

Your algorithm design is described here.

Show your manual calculations (hand work) here.

To find the binary cosine coefficients, you need to take the intersection size of the document and the query and divide it by the product of the square root of the document size and the square root of the query size.



**Step 4: Implementation in C++**

Also known as coding

Develop a C++ solution using your work from step 3.

Write the declarations first and then write the C++ statements.

Enter and debug program on the computer.

Show your source code here:

#include <iostream>

#include <vector>

using namespace std;

class StringSet

{

private:

vector<string> data;

public:

StringSet(); //Default Constructor

~StringSet(); //Destructor

StringSet(string arr[]); //Takes as an input parameter an array of strings for the initial values in the set

void add(string s); //Adds a string to the set

void remove(string s); //Removes a string from the set

void clear(); //clear the entire set to empty

int search(string s); //Search for a string

int size(); //Returns size of strings in set

void output(); //Outputs all strings in the set

StringSet operator+ (StringSet p); //Overloads the + operator

friend StringSet operator\*(StringSet& set1, StringSet& set2); //Overloads the \* operator

friend ostream& operator <<(ostream& output, StringSet& p); //Overloads the << operator

friend istream& operator >>(istream& input, StringSet& p); //Overloads the >> operator

bool operator == (StringSet& p); //Overloads the == operator

};

StringSet::StringSet() //Default Constructor

{

}

StringSet::~StringSet() //Destructor

{

data.clear();

}

StringSet::StringSet(string arr[]) //Takes as an input parameter an array of strings for the initial values in the set

{

for (int i = 0; i < sizeof(arr); i++)

data.push\_back(arr[i]);

}

void StringSet::add(string str) //Adds a string to the set

{

data.push\_back(str);

}

void StringSet::remove(string str) //Removes a string from the set

{

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

if (data[i] == str)

data.erase(data.begin() + i);

}

void StringSet::clear() //clear the entire set to empty

{

data.clear();

}

int StringSet::search(string str) //Search for a string

{

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

if (data[i] == str)

return i;

return -1;

}

int StringSet::size() //Returns size of strings in set

{

return data.size();

}

void StringSet::output() //Outputs all strings in the set

{

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

cout << data[i] << endl;

}

StringSet StringSet::operator+(StringSet p) //Overloads the + operator

{

StringSet temp;

temp.data.resize(data.size() + p.data.size());

for (int i = 0; i < temp.data.size(); i++)

if (i < data.size())

temp.data[i] = data[i];

else

temp.data[i] = p.data[i - data.size()];

return temp;

}

StringSet operator\*(StringSet& set1, StringSet& set2) //Overloads the \* operator

{

StringSet temp;

for (int i = 0; i < set1.data.size(); i++)

for (int x = 0; x < set2.data.size(); x++)

if (set1.data[i] == set2.data[x])

temp.add(set1.data[i]);

return temp;

}

ostream& operator <<(ostream& output, StringSet& p) //Overloads the << operator

{

for (int i = 0; i < p.data.size(); i++)

output << p.data[i] << endl;

return output;

}

istream& operator>>(istream& input, StringSet& p) //Overloads the >> operator

{

string s;

input >> s;

p.data.push\_back(s);

return input;

}

bool StringSet::operator == (StringSet& p) //Overloads the == operator

{

if (data.size() == p.data.size())

{

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

if (data[i] != p.data[i])

return false;

}

else

return false;

return true;

}

**Step 5: Testing**

Test your program with sample data set to make sure the output is correct.

Should test multiple data sets including the boundary cases.

Summary and analyze your result.

Show the output screen shots here.

