/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Exercise 2.1. Consider the quadratic polynomial q(x) = x^2 − 111.11x + 1.2121.

1. Find the roots of q using the two forms of the quadratic formula. After every operation (+, −, ×, /, sqrt) truncate

the result to five significant decimal digits.

2. For a quadratic polynomial ax^2 + bx + c, show that the roots x1 and x2 satisfy x1x2 = c/a. Use this latter

relationship to evaluate the root having smaller absolute value.

3. Write a C++ program that reads constants a, b, c from the command line, applies the appropriate formula to

calculate the roots, and outputs the roots.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream> // cin and cout are here

#include <cmath> // cos and M\_PI are here

#include <cstdlib> // strtold is here

using namespace std;

int main(int argc, char\* argv[])

{

char\* next;

double a = 0;

double b = 0;

double c = 0;

double x1 = 0;

double x2 = 0;

if(argc < 4)

{

cout << "User input should be coeff of quadratic eqaution (a, b & c)" << endl;

return 1;

}

else if((b\*b-4\*a\*c) < 0)

{

cout << "No real roots for the given quadratic equation" << endl;

return 1;

}

a = strtold(argv[1], &next);

b = strtold(argv[2], &next);

c = strtold(argv[3], &next);

if(b < 0)

{

x1 = (-b + sqrt(b\*b-4\*a\*c))/(2\*a);

x2 = c/(a\*x1);

}

else

{

x1 = (-b - sqrt(b\*b-4\*a\*c))/(2\*a);

x2 = c/(a\*x1);

}

cout << "Roots for given quadratic equation are as follows:" << endl;

cout << "x1: "<< x1 << endl;

cout << "x2: "<< x2 << endl;

return 0;

}

**OUTPUT:**

homework2 $./exercise2\_1 1 111.11 1.2121

Roots for given quadratic equation are as follows:

x1: -111.099

x2: -0.0109101

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Exercise 2.2. Write a C++ program that does the following:

1. Reads an integer n from the command line.

2. Calculates sum=1/ (i\*i). Accumulates the result in a variable declared as float up = 0.0.

3. Calculates the sum of the same terms in reverse order.Accumulates the result in a variable declared as float down = 0.0.

4. Prints n and prints up and down with 12 decimal places.

Run the program with argument 5000 and comment on the result.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream> // cin and cout are here

#include <cstdlib> // strtold is here

#include <iomanip>

using namespace std;

int main(int argc, char\* argv[])

{

if(argc < 2)

{

cout << "Please enter a number" << endl;

return 1;

}

char\* next;

float up = 0.0;

float down = 0.0;

int i = 0;

int n = strtold(argv[1], &next);

for(i=1; i<=n ; i++){

float x = i;

up += 1/(x\*x);

down += 1/((n-x+1)\*(n-x+1));

}

/\* set the precision for upto 12 decimal places \*/

cout << fixed;

cout << setprecision(12);

cout << "Input n: " << up << endl;

cout << "Result up: " << up << endl;

cout << "Result down: " << down << endl;

return 0;

}

**OUTPUT:**

homework2 $./exercise2\_2 5000

Input n: 1.644725322723

Result up: 1.644725322723

Result down: 1.644734025002

Here the results are different starting from 5th decimal place. Since we are adding the numbers in the reverse order in second case; it will try to eliminate the problem of adding numbers of very different magnitude as addition of first (n-1) terms will build up a number close to nth term. So, Result down will be more accurate computationally.