## **calculator**

### Task 1 1

Input the radius of the circle.

float radius;  
input\_number(radius, "Enter the radius of the circle: ");

Calculate the area and circumference of the circle.

float area = M\_PI \* radius \* radius;  
float circumference = 2 \* M\_PI \* radius;  
std::cout << "Area of a circle with radius " << radius << " = " << area << std::endl;  
std::cout << "Circumference of a circle with radius " << radius << " = " << circumference  
<< std::endl;

### Task 1 2

Input the principal amount, rate of interest, and time period.

float principal, rate, time;  
input\_number(principal, "Enter the principal amount: ");  
input\_number(rate, "Enter the rate of interest (%): ");  
input\_number(time, "Enter the time period (years): ");

Calculate the simple interest.

float simple\_interest = (principal \* rate \* time) / 100;  
std::cout << "Simple Interest on a principal amount of " << principal << " at a rate of "  
<< rate << "% for " << time << " years = " << simple\_interest << std::endl;

### Task 2 1

False, they are stored as binary numbers.

return 0;  
}

### Task 2 2

Device.

### Task 2 3

.cpp extension.

### Task 2 4

Translate source code into machine readable code (or exectuable program) that the computer can run.

### Task 2 5

8 bits.

### Task 2 6

Hexadecimal (base 16).

### Task 2 7

.

### Task 2 8

True.

### Task 2 9

cout is used without the namespace std, and line 5 is missing a semicolon at the end of the line.

### Task 2 10

Missing the insertion operator on line 8, correct line is cin >> num1 >> num2;

## **trig\_calculator**

### Task 3 0

Start with a verbose number input function to re-use.

void input\_number(double& number, const std::string& input\_message) {  
while (true) {  
std::cout << input\_message;  
std::cin >> number;  
  
if (std::cin.fail()) {  
std::cerr << "Invalid input. Please enter a valid number." << std::endl;  
std::cin.clear();  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
} else {  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
return;  
}  
}  
}

Implement the main program loop with the atanh and acosh functions.

int main() {  
while (true) {  
double choice;  
std::cout << "Select an operation:\n1. Inverse Hyperbolic Tangent\n"  
<< "2. Inverse Hyperbolic Cosine\n0. Exit\n";  
input\_number(choice, "Enter your choice: ");  
  
if (choice == 0) {  
break;  
}  
  
double x;  
switch (static\_cast<int>(choice)) {  
case 1:  
input\_number(x, "Enter a value for x (between -1 and 1): ");  
std::cout << "The inverse hyperbolic tangent of " << x << " is " << atanh(x)  
<< std::endl;  
break;  
case 2:  
input\_number(x, "Enter a value for x (greater than or equal to 1): ");  
std::cout << "The inverse hyperbolic cosine of " << x << " is " << acosh(x)  
<< std::endl;  
break;  
default:  
std::cerr << "Error: Invalid operation number." << std::endl;  
break;  
}  
}  
  
return 0;  
}

### Task 3 1

Implement the atanh function, with error checking.

double atanh(double x) {  
if (x < -1 || x > 1) {  
std::cerr << "\nError: The input value must be between -1 and 1." << std::endl;  
return NAN;  
}  
return 0.5 \* log((1 + x) / (1 - x));  
}

### Task 3 2

Implement the acosh function, with error checking.

double acosh(double x) {  
if (x < 1) {  
std::cerr << "\nError: The input value must be greater than or equal to 1." << std::endl;  
return NAN;  
}  
return log(x + sqrt(x \* x - 1));  
}

### Task 4 1

bba.

### Task 4 2

TTTT, although note for the third statement, technically the return statement can be implicit, but it’s always required even if it’s implicitly there.

## **quadratic\_calculator**

### Task 5 0

Include the necessary libraries.

#include <cmath>  
#include <iostream>  
#include <limits>  
#include <tuple>

Start with a verbose number input function to re-use.

void input\_number(double& number, const std::string& input\_message) {  
while (true) {  
std::cout << input\_message;  
std::cin >> number;  
  
if (std::cin.fail()) {  
std::cerr << "Invalid input. Please enter a valid number." << std::endl;  
std::cin.clear();  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
} else {  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
return;  
}  
}  
}

Calculate the quadratic’s solution.

std::tuple<double, double> solve\_quadratic(const double a, const double b, const double c) {  
double discriminant = pow(b, 2) - 4 \* a \* c;  
if (discriminant < 0) {  
return std::make\_tuple(NAN, NAN);  
}  
  
double x1 = (-b + sqrt(discriminant)) / (2 \* a);  
double x2 = (-b - sqrt(discriminant)) / (2 \* a);  
  
return std::make\_tuple(x1, x2);  
}

Implement the main program loop.

int main() {  
while (true) {  
double a, b, c;  
input\_number(a, "Enter the coefficient of x^2: ");  
input\_number(b, "Enter the coefficient of x: ");  
input\_number(c, "Enter the constant term: ");  
  
std::tuple<double, double> x = solve\_quadratic(a, b, c);  
  
std::cout << "The quadratic equation " << a << "x^2 + " << b << "x + " << c << " = 0 has ";  
  
if (std::isnan(std::get<0>(x))) {  
std::cout << "no real solutions." << std::endl;  
} else if (std::get<0>(x) == std::get<1>(x)) {  
std::cout << "one solution: x = " << std::get<0>(x)  
<< std::endl;  
} else {  
std::cout << "two solutions: x = " << std::get<0>(x)  
<< " and x = " << std::get<1>(x) << std::endl;  
}  
  
char response;  
std::cout << "Would you like to solve another quadratic equation? (y/n): ";  
std::cin >> response;  
  
if (response != 'y') {  
break;  
}  
}  
  
return 0;  
}

## **variable\_manipulation**

### Task 6 0

Include the necessary libraries.

#include <cmath>  
#include <iostream>

### Task 6 1

Declare the variables with their respective data types.

int x = 25;  
int y = 18;  
double pay\_rate = 12.50;  
int first\_number = 10;  
int temporary\_number = first\_number;

### Task 6 2

Swap the values of x and y using a temporary variable.

int temporary\_x = x;  
x = y;  
y = temporary\_x;

### Task 6 3

Output the values of x and y and evaluate the expression.

std::cout << "The value of x is: " << x << std::endl;  
std::cout << "The value of y is: " << y << std::endl;  
double evaluated\_expression = (x + 12) / (y - 18 + pow(x, 3)); // Unsure if correct expression  
std::cout << "Substituting these values into the expression (x + 12) / (y - 18 + x^3) gives: "  
<< evaluated\_expression << std::endl;