SHEET 3 SOLUTIONS

SHEHAB MAHMOUD SALAH | 2100320

GENERAL NOTE: to **copy** code **from** this PDF document, **copy each block of code** separately to **not** lose the code's formatting, alternatively you can find all the **source code** to **PROGRAMMING EXERCISES** on my **GitHub**: https://bit.ly/CSE131Sheets **happy compiling** 😁

1. a)

```
f(x) = g(x) + sqrt(h(x))
f(2) = g(2) + sqrt(h(2))

g(x) = 4 * h(x)
g(2) = 4 * h(2)

h(x) = x * x + k(x) - 1
h(2) = 2 * 2 + k(2) - 1
h(2) = 4 + k(2) - 1

k(x) = 2 * (x + 1)
k(2) = 2 * (2 + 1)
k(2) = 2 * 3
k(2) = 6
```

Now, substitute the value of k(2) into the expression for h(2):

```
h(2) = 4 + 6 - 1
h(2) = 9
```

Next, substitute the value of h(2) into the expression for g(2):

```
g(2) = 4 * 9
g(2) = 36
```

Finally, substitute the values of g(2) and h(2) into the expression for f(2):

```
f(2) = 36 + sqrt(9)
f(2) = 36 + 3
f(2) = 39
```

∴ x1 = 39

b)

```
double x2 = g(h(2));
```

We have already found the value of h(2) earlier:

```
h(2) = 9
```

Now, let's find the value of g(h(2)):

```
g(x) = 4 * h(x)
g(h(2)) = 4 * h(9)
g(h(2)) = 4 * 100
```

∴ x2 = 400

c)

```
double x3 = k(g(2) + h(2));
```

We already have the values for g(2) and h(2):

```
g(2) = 36
h(2) = 9
```

Now, let's find the value of k(g(2) + h(2)):

```
k(x) = 2 * (x + 1)
k(g(2) + h(2)) = k(36 + 9)
k(45) = 2 * (45 + 1)
k(45) = 2 * 46
k(45) = 92
```

∴ x3 = 92

d)

```
double x4 = f(0) + f(1) + f(2);
```

We already have the value for f(2):

```
f(2) = 39
```

Now let's find the values for f(0) and f(1):

```
f(0) = g(0) + sqrt(h(0))
h(0) = 0 * 0 + k(0) - 1
h(0) = k(0) - 1

k(0) = 2 * (0 + 1)
k(0) = 2

h(0) = 2 - 1
h(0) = 1

g(0) = 4 * h(0)
g(0) = 4 * 1
g(0) = 4

f(0) = 4 + sqrt(1)
f(0) = 4 + 1
f(0) = 5

f(1) = g(1) + sqrt(h(1))
h(1) = 1 * 1 + k(1) - 1
h(1) = 1 + k(1) - 1

k(1) = 2 * (1 + 1)
k(1) = 2 * 2
k(1) = 4
```

```
h(1) = 1 + 4 - 1
h(1) = 4

g(1) = 4 * h(1)
g(1) = 4 * 4
g(1) = 16

f(1) = 16 + sqrt(4)
f(1) = 16 + 2
f(1) = 18
```

Now we can find the value of x4:

```
x4 = f(0) + f(1) + f(2)
x4 = 5 + 18 + 39
```

∴ x4 = 62

e)

```
double x5 = f(-1) + g(-1) + h(-1) + k(-1);

k(-1)
k(x) = 2 * (x + 1)
k(-1) = 2 * (-1 + 1)
k(-1) = 2 * 0
k(-1) = 0

h(-1)
h(x) = x * x + k(x) - 1
h(-1) = (-1) * (-1) + k(-1) - 1
h(-1) = 1 + 0 - 1
h(-1) = 0

g(-1)
g(x) = 4 * h(x)
g(-1) = 4 * h(-1)
g(-1) = 4 * 0
g(-1) = 0

f(-1)
f(x) = g(x) + sqrt(h(x))
f(-1) = g(-1) + sqrt(h(-1))
f(-1) = 0 + sqrt(0)
f(-1) = 0
```

Now let's find the value of x5:

```
x5 = f(-1) + g(-1) + h(-1) + k(-1)
x5 = 0 + 0 + 0 + 0
```

∴ x5 = 0

2. (Exercise(1) on GitHub)

```cpp
#include <iostream>
#include <cmath>
using namespace std;

// Function to compute the perimeter of a circle
double perimeter(float r) {
    const double PI = 3.14159265358979323846;
    return 2 * PI * r;
}

// Function to compute the area of a circle
double area(float r) {
    const double PI = 3.14159265358979323846;
    return PI * r * r;
}

int main() {
    float radius;

    cout << "Enter the radius of the circle: ";
    cin >> radius;

    cout << "Perimeter: " << perimeter(radius) << endl;
    cout << "Area: " << area(radius) << endl;

    return 0;
}
```

3. (Exercise(2) on GitHub)

```cpp
#include <iostream>
using namespace std;

// Function to test whether a year is a leap year
bool leap_year(int year) {
    if (year % 4 == 0) {
        if (year % 100 == 0) {
            return year % 400 == 0;
        } else {
            return true;
        }
    } else {
        return false;
    }
}

int main() {
    int year;

    cout << "Enter a year: ";
    cin >> year;
```

```
    if (leap_year(year)) {
        cout << year << " is a leap year." << endl;
    } else {
        cout << year << " is not a leap year." << endl;
    }

    return 0;
}
```

4. The **false_swap2** function doesn't swap the contents of **x** and **y** because it's swapping the values of the **local copies** of the variables **a** and **b**. In C++, function arguments are passed by value by default, which means that the function receives a copy of the original variables rather than a **reference** to the original variables themselves. So, any changes made to a and b inside the function **do not affect** the original variables **x** and **y**.

   To fix this issue, you can pass the variables **by reference** using the **&** symbol, which allows the function to directly modify the original variables. Here's the modified code:

   (**Exercise(3)** on GitHub)

```
#include <iostream>
using namespace std;

// Function to swap the values of two integers
void true_swap(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}

int main()
{
    int x = 3;
    int y = 4;
    true_swap(x, y);
    cout << x << " " << y << "\n"; // This should now output "4 3"
    return 0;
}
```

5. (Exercise(4) on GitHub)

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    double x, y;
    double m_x, m_x_squared, m_5x, m_sqrt_x;

    for (x = 0; x <= 10; x += 0.1) {
        // Calculate m(x)
        m_x = 7 * pow(x, 3) - 5 * pow(x, 2) + 2 * x + 11;

        // Calculate m(x^2)
        m_x_squared = 7 * pow(pow(x, 2), 3) - 5 * pow(pow(x, 2), 2) +
2 * pow(x, 2) + 11;

        // Calculate m(5x)
        m_5x = 7 * pow(5 * x, 3) - 5 * pow(5 * x, 2) + 2 * (5 * x) +
11;

        // Calculate m(sqrt(x))^0.2
        m_sqrt_x = 7 * pow(pow(sqrt(x), 0.2), 3) - 5 *
pow(pow(sqrt(x), 0.2), 2) + 2 * pow(sqrt(x), 0.2) + 11;

        // Calculate y(x)
        y = (m_x_squared + m_5x) / m_sqrt_x;

        // Print the result
        cout << "y(" << x << ") = " << y << endl;
    }

    return 0;
}
```

6. (Exercise(5) on GitHub)

```cpp
#include <iostream>
#include <cmath>
using namespace std;

// Function to calculate m(x)
double m(double x) {
    return 7 * pow(x, 3) - 5 * pow(x, 2) + 2 * x + 11;
}

// Function to calculate y(x)
double y(double x) {
    double m_x_squared = m(pow(x, 2));
    double m_5x = m(5 * x);
    double m_sqrt_x = m(pow(sqrt(x), 0.2));
    return (m_x_squared + m_5x) / m_sqrt_x;
}
```

```
int main() {
    double x;

    for (x = 0; x <= 10; x += 0.1) {
        // Calculate and print the result
        cout << "y(" << x << ") = " << y(x) << endl;
    }

    return 0;
}
```

7. There's a slight inconsistency in the problem statement: the prototype of the function is `void isPrime(int x)`, which means it **does not return any value**, but the problem description asks to **return 1 if the number is prime and 0 if not**. I will provide a solution using a function that returns an `int` value instead of `void`. Here's the corrected prototype and implementation:

(**Exercise(6)** on GitHub)

```
#include <iostream>
#include <cmath>
using namespace std;

// Function to check if the given integer value is prime or not
int isPrime(int x) {
    if (x <= 1) {
        return 0;
    }

    for (int i = 2; i <= sqrt(x); i++) {
        if (x % i == 0) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int number;

    cout << "Enter an integer: ";
    cin >> number;

    if (isPrime(number)) {
        cout << number << " is prime." << endl;
    } else {
        cout << number << " is not prime." << endl;
    }

    return 0;
}
```

This concludes Sheet (3) Solutions, this document + source code to all programming exercises available on https://bit.ly/CSE131Sheets **.**