Exercise 1

Code:

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;

int main() {

    // init vars to store dimensions of cylinder and cube
    double const PI = 3.14159;
    int radius;
    int height;
    double side;
    double volume;

    // get cylinder dimensions from user
    cout << fixed << showpoint << setprecision(2);
    cout << "Enter the radius of the cylinder: " << endl;
    cin >> radius;
    cout << "Enter the height of the cylinder: " << endl;
    cin >> height;

    // calculate volume of cylinder
    volume = PI * radius * radius * height;
    // calculate side length of cube with equivalent volune
    side = std::cbrt(volume);

    // output side length to user
    cout << "The side length of a cube with the same volume as your cylidner: " << side << endl;

    return 0;
}
```

Output:

```
Microsoft Visual Studio Debug Console
Enter the radius of the cylinder:
12
Enter the height of the cylinder:
12
The side length of a cube with the same volume as your cylidner: 17.58
```

Exercise 2

Code:

```
(Global Scope)
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;

int main() {

    // init vars to store info about tree count and space requirements
    double const PI = 3.14159;
    int len;
    int rad;
    int space;
    int num_trees;
    double space_used;

    // set output to 2 sig figs
    cout << fixed << showpoint << setprecision(2);

    // get yard dimension from user
    cout << "Enter the length of the yard: " << endl;
    cin >> len;

    // get tree radius from user
    cout << "Enter the radius of a fully grown tree: " << endl;
    cin >> rad;

    // get tree space-between from user
    cout << "Enter the required space between fully grown trees: " << endl;
    cin >> space;

    // calculate the number of trees that can fit in yard
    num_trees = static_cast<int> ((len + space) / (2 * rad + space));
    // calculate total space used by trees
    space_used = PI * rad * rad * num_trees;

    // output number of trees and soace occcupied to user
    cout << "The number of trees that can be planted: " << num_trees << endl;
    cout << "The amount of space occupied by trees: " << space_used << endl;
    return 0;
}
```

Output:

```
Microsoft Visual Studio Debug Console
Enter the length of the yard:
12
Enter the radius of a fully grown tree:
1
Enter the required space between fully grown trees:
1
The number of trees that can be planted: 4
The amount of space occupied by trees: 12.57
```

Exercise 3

Code:

```cpp
#include <iostream>

using namespace std;

int main() {

    // init vars for populaation and growth rate for 2 towns, and years
    double pop_a;
    double pop_b;
    double growth_a;
    double growth_b;
    int years = 0;

    // get population and grwoth rate info for 2 towns from user
    cout << "Enter the current population of town A: " << endl;
    cin >> pop_a;
    cout << "Enter the current population of town B (must be greater than town A's): " << endl;
    cin >> pop_b;
    cout << "Enter the growth rate of town A: " << endl;
    cin >> growth_a;
    cout << "Enter the growth rate of town B (must be less than town A's.)" << endl;
    cin >> growth_b;

    // convert given growth rate into a decimal representing a percentage
    growth_a = growth_a / 100 + 1;
    growth_b = growth_b / 100 + 1;

    // simulate pop growth for both towns until pop a >= pop b and keep track of the years
    while (pop_a < pop_b) {
        pop_a *= growth_a;
        pop_b *= growth_b;
        years++;
    }

    // output the years until pop a >= pop b and the populations of each town at that time
    cout << "After " << years << " year(s) the population of town A will be greater than or equal to the population of town B." << endl
    cout << "After " << years << " years, population of town A: " << static_cast<int> (pop_a) << endl;
    cout << "After " << years << " years, population of town B: " << static_cast<int> (pop_b) << endl;

    return 0;
}
```

Output:

```
Microsoft Visual Studio Debug Console
Enter the current population of town A:
5000
Enter the current population of town B (must be greater than town A's):
8000
Enter the growth rate of town A:
10
Enter the growth rate of town B (must be less than town A's.)
1
After 6 year(s) the population of town A will be greater than or equal to the population of town B.
After 6 years, population of town A: 8857
After 6 years, population of town B: 8492
```

Exercise 4

Code:

```cpp
#include <iostream>

using namespace std;

int main() {

    // init constant list of primes
    const int PRIMES[11] = {2,3,5,7,11,13,17,19,23,29,31};
    // init number to be tested
    int num;
    bool isPrime = true;

    // get number from user
    cout << "Enter a positive integer between 1 and 1000 (inclusive): " << endl;
    cin >> num;

    // check if number can be divided by a prime in our list. if so, prints out the prime and sets isPrime to false
    for (int i = 0; i < sizeof(PRIMES)/sizeof(int); i++) {
        if (num % PRIMES[i] == 0) {
            isPrime = false;
            cout << PRIMES[i] << endl;
        }
    }

    // outputs whether or not the number is prime.
    if (isPrime) {
        cout << num << " is prime." << endl;
    }
    else {
        cout << num << " is not prime. All of the numbers above divide your number." << endl;
    }

    return 0;
}
```

Output:

```
Microsoft Visual Studio Debug Console
Enter a positive integer between 1 and 1000 (inclusive):
431
431 is prime.
```

Exercise 5

Code:

```cpp
#include <iostream>

using namespace std;

int main() {

    // init constants for various royalty rates
    double const RATE1 = .125;
    double const RATE2 = .1;
    double const RATE3 = .14;
    int const ROYALTY1 = 25000;

    // init vars for info on books sold
    int num_sold;
    double net_price;
    double royalty2;
    double royalty3;

    // get price and number of books sold from user
    cout << "Enter the net price of the novel: " << endl;
    cin >> net_price;
    cout << "Enter the number of copies sold: " << endl;
    cin >> num_sold;

    // calculate royalties for option 2
    royalty2 = RATE1 * num_sold * net_price;

    // calculate royalties for option 3
    if (num_sold > 4000) {
        royalty3 = (RATE2 * 4000 * net_price) + (RATE3 * (num_sold - 4000) * net_price);
    }
    else {
        royalty3 = RATE2 * num_sold * net_price;
    }

    // output all royalty options to user
    cout << "O1:  " << ROYALTY1 << ", O2: " << royalty2 << ", O3: " << royalty3 << endl;

    // compare and output most profitable royalty option to user
    if (ROYALTY1 > royalty2 && ROYALTY1 > royalty3) {
        cout << "Best: O1" << endl;
    }
    else if (royalty2 > ROYALTY1 && royalty2 > royalty3) {
        cout << "Best: O2" << endl;
    }
    else {
        cout << "Best: O3" << endl;
    }

    return 0;
}
```

Output:

```
Microsoft Visual Studio Debug Console
Enter the net price of the novel:
30
Enter the number of copies sold:
5000
O1:  25000, O2: 18750, O3: 16200
Best: O1
```