- **Name: yuyang tian**
- **NU ID: 002297971**

1 Cylindrical container

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

int main() {
 double radius;
 double height;
 double volume;
 double side;

 cout << "input the radius of the base" << endl;
 cin >> radius;
 cout << "input the height of a cylindrical container" << endl;
 cin >> height;

 volume = M_PI * radius * radius * height;
 side = cbrt(volume);
 cout << "the side of the cube with the same volume is " << side << endl;
}
```

2 Plant tree in yard

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

int main() {
    double length;
    double radius;
    double space;


    cout << "input the length of the yard" << endl;
    cin >> length;
    cout << "input the radius of a fully grown tree" << endl;
    cin >> radius;
    cout << "input the required space between fully grown trees" << endl;
    cin >> space;

    double eachOccupied = 2*radius+space;
    int count = static_cast<int> (length/eachOccupied);
    double occupied = eachOccupied * count;
    cout << "the number of trees that can be planted in the yard is " << count
<< endl;
    cout << "the total space that will be occupied by the fully grown trees is "
<< occupied << endl;

    return 0;
}
```

3 Population growth

```
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    double populationA, growthRateA, populationB, growthRateB;

    cout << "input the initial population of town A: ";
    cin >> populationA;
    cout << "input the growth rate of town A (in decimal form): ";
    cin >> growthRateA;
    cout << "input the initial population of town B: ";
    cin >> populationB;
    cout << "input the growth rate of town B (in decimal form): ";
    cin >> growthRateB;

    // Calculate the number of years required
    double years = (log(populationB) - log(populationA)) / (log(1 + growthRateA)
- log(1 + growthRateB));

    // Check if years is a valid number (not NaN or infinity)
    if (!isnan(years) && !isinf(years) && years >= 0) {
        // Output the result
        cout << "It will take approximately " << years << " years for town A's
population to surpass or equal town B's population." << endl;
    } else {
        cout << "Invalid input. The populations and growth rates provided may
not lead to a crossover." << endl;
    }
    return 0;
}
```

4 Primes

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

int main() {
    int primes[11] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31};
    string res;
    int num;
    bool isPrime = true;
    cout << "enter a positive integer between 1 and 1000 (inclusive):" << endl;
    cin >> num;

    for (int prime : primes)
    {
        if(num % prime == 0) {
            isPrime = false;
            res += to_string(prime) + " ";
        }
    }
    if(isPrime) {
        cout << "this number is a prime." << endl;
    } else {
        cout << "this number is not a prime, and it can be divided by " << res
<< "."<< endl;
    }
    return 0;
}
```

5 Novel income

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

int main() {
    double netPrice;
    int numberOfCopy;
```

```cpp
    // Prompt the user to enter input
    cout << "Enter the net price of each copy of the novel: $";
    cin >> netPrice;
    cout << "Enter the estimated number of copies that will be sold: ";
    cin >> numberOfCopy;

    const double manuscriptRoyalty = 5000;
    const double publicationRoyalty = 20000;
    const double generalRate = 0.125;
    const double basicRate = 0.1;
    const double highRate = 0.14;

    double option1Royalty = manuscriptRoyalty + publicationRoyalty;
    double option2Royalty = numberOfCopy * netPrice * generalRate;
    double option3Royalty =
        (numberOfCopy > 4000) ? (4000 * netPrice * basicRate + (numberOfCopy -
4000) * netPrice * highRate) : (numberOfCopy * netPrice * basicRate);

    string bestOption;
    double highestRoyalty = option1Royalty;

    if (option2Royalty > highestRoyalty) {
        highestRoyalty = option2Royalty;
        bestOption = "Option 2";
    }

    if (option3Royalty > highestRoyalty) {
        bestOption = "Option 3";
    }

    cout << "Royalty for Option 1: $" << option1Royalty << endl;
    cout << "Royalty for Option 2: $" << option2Royalty << endl;
    cout << "Royalty for Option 3: $" << option3Royalty << endl;
    cout << "The best option for the author is " << bestOption << endl;

    return 0;
}
```