Gordeev Mikhail

1. Make the substitution $1 + \frac{1}{u} = e^t$.

$$u = \frac{1}{e^t - 1}$$

$$(2u+1)\ln\left(1+\frac{1}{u}\right) - 2 = t\left(\frac{2}{e^t-1}+1\right) - 2 = \frac{e^t(t-2)+t+2}{e^t-1}$$

$u > 0 \Rightarrow t > 0 \Rightarrow e^t - 1 > 0$.
Now consider $e^t(t-2)+t+2$. Taylor series expansion of exponential function is

$$e^t = 1 + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \dots$$

Use the expansion in the equation.

$$(1 + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \dots)(t-2) + t + 2 =$$

$$t - 2 + t^2 - 2t + \frac{t^3}{2} - t^2 + t + 2 + \dots + \frac{t^n}{(n-1)!} - \frac{t^n}{n!/2} + \dots$$

$$(n-1)! < n!/2 \Rightarrow \frac{t^n}{(n-1)!} - \frac{t^n}{n!/2} > 0$$

The expression $(2u+1)\ln\left(1+\frac{1}{u}\right) - 2$ is positive for every $u > 0$.

4. X – symmetric positive definite matrix $\Rightarrow X = PDP^{-1}$, where D – diagonal matrix with positive elements.
$X^{-n} = PD^{-n}P^{-1}$, $(X^n + X^{2n})^{-1} = P(D^n + D^{2n})^{-1}P^{-1}$.
Trace of all expression is equal to trace of the inner diagonal matrix D'. For all $i \in [1..m]$:

$$d'_i = d_i^{-n} - (d_i^n + d_i^{2n})^{-1} = \frac{1}{d_i^n} - \frac{1}{d_i^n + d_i^{2n}} = \frac{d_i^{2n} + d_i^n - d_i^n}{d_i^{2n} + d_i^{3n}} = \frac{1}{1 + d_i^n}$$

Trace $= \frac{1}{1+d_1^n} + \frac{1}{1+d_2^n} + \dots + \frac{1}{1+d_m^n}$.

$$\begin{cases} d_i < 1 & \Rightarrow d'_i \xrightarrow[n\to\infty]{} 1 \\ d_i > 1 & \Rightarrow d'_i \xrightarrow[n\to\infty]{} 0 \\ d_i = 1 & \Rightarrow d'_i = \frac{1}{2} \end{cases}$$

The answer can be any number $k/2$, where k is integer in range $[0..2m]$.

6. Operators `+` and `<<` have to return non void result. Operator `+` must be constant and return sum of its operands. Operator `<<` shuld use template stream, be a *friend* function and return stream. If use ostream then need to write `std::ostream`.

```
class Complex
{
public:
  Complex(double real, double imaginary = 0)
  : _real(real),_imaginary(imaginary) {}

  Complex operator+(Complex other) const
  {
    Complex temp = *this;
    temp._real += other._real;
    temp._imaginary += other._imaginary;

    return temp;
  }

  template <typename T>
  friend T& ::operator<<(T& os, Complex x)
  {
    os<<"("<<x._real<<","<<x._imaginary<<")";

    return os;
  }
```

```
  Complex operator++()
  {
    ++_real;
    return *this;
  }

  Complex operator++(int)
  {
    Complex temp = *this;
    ++_real;
    return temp;
  }
private:
  double _real, _imaginary;
};
```

7.

```
bool is_balanced(BSTree *tree) {
  std::stack<std::pair<int, BSTree*>> st;
  st.push(std::make_pair(0,tree));
  int dmax = 0, dmin = tree->count();

  while (!st.empty()) {
    auto p = st.top();
    st.pop();
    int d = p.first;
    BSTree *t = p.second;
    if (t->is_leaf()) {
      dmin = std::min(dmin, d);
      dmax = std::max(dmax, d);
    } else {
      st.push(std::make_pair(d+1, t->left()));
      st.push(std::make_pair(d+1, t->right()));
    }
  }
```

```
    return abs(dmin - dmax) <= 1;
}
```