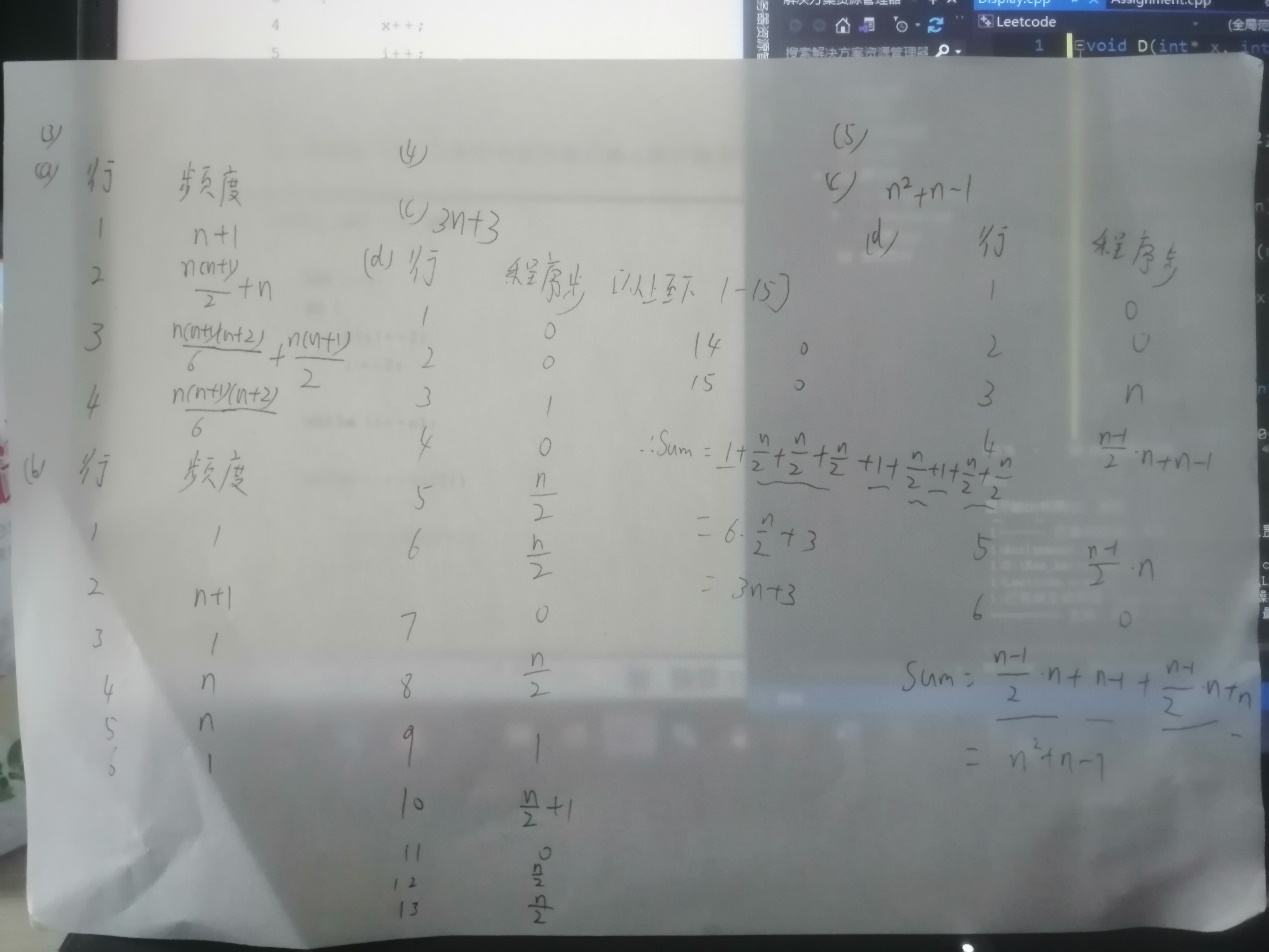
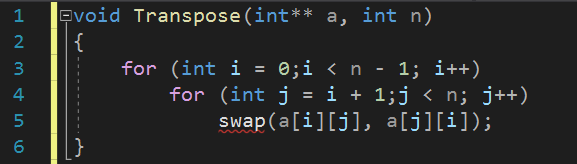
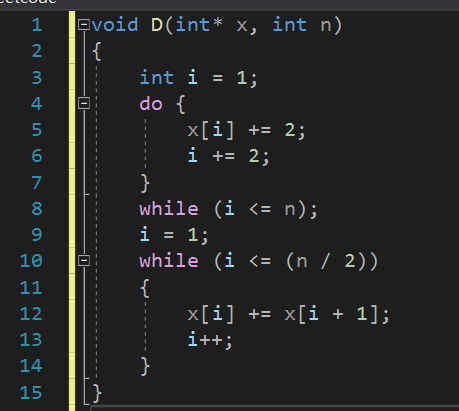
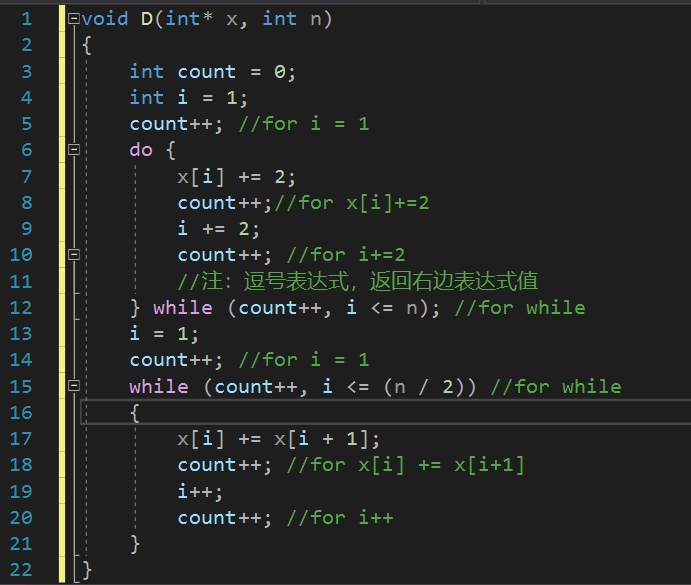
第三题3(a)(b) 第四题(c)(d) 第五题(c)(d)如下：[(a)(b)见下方]



行数参考:



4(a) 图片加代码



void D(int\* x, int n)

{

int count = 0;

int i = 1;

count++; //for i = 1

do {

x[i] += 2;

count++;//for x[i]+=2

i += 2;

count++; //for i+=2

//注：逗号表达式，返回右边表达式值

} while (count++, i <= n); //for while

i = 1;

count++; //for i = 1

while (count++, i <= (n / 2)) //for while

{

x[i] += x[i + 1];

count++; //for x[i] += x[i+1]

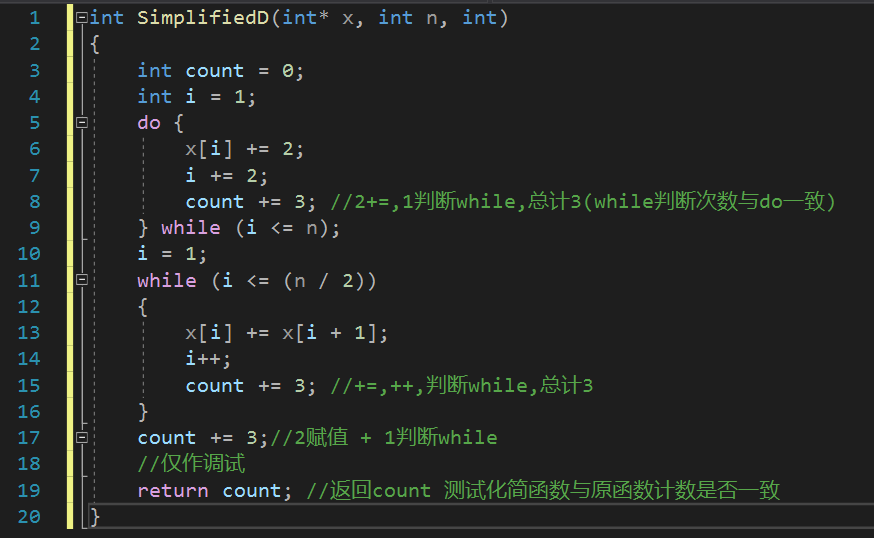
i++;

count++; //for i++

}

}

4(b)



int SimplifiedD(int\* x, int n, int)

{

int count = 0;

int i = 1;

do {

x[i] += 2;

i += 2;

count += 3; //2+=,1判断while,总计3(while判断次数与do一致)

} while (i <= n);

i = 1;

while (i <= (n / 2))

{

x[i] += x[i + 1];

i++;

count += 3; //+=,++,判断while,总计3

}

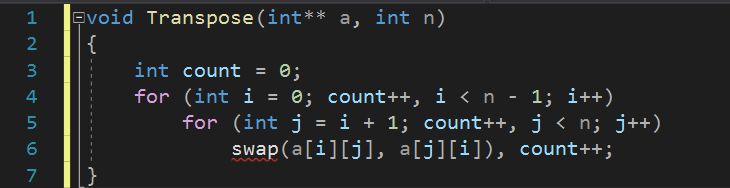
count += 3;//2赋值 + 1判断while

//仅作调试

return count; //返回count 测试化简函数与原函数计数是否一致

}

5(a)



void Transpose(int\*\* a, int n)

{

int count = 0;

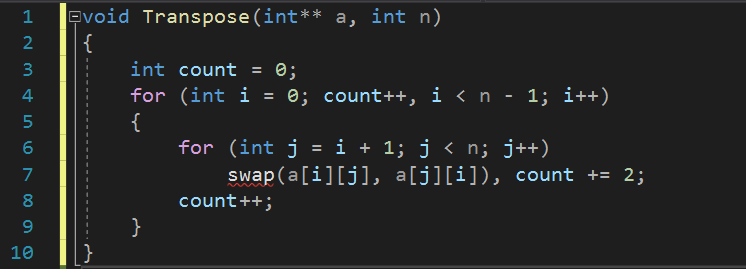
for (int i = 0; count++, i < n - 1; i++)

for (int j = i + 1; count++, j < n; j++)

swap(a[i][j], a[j][i]), count++;

}

5(b)



void Transpose(int\*\* a, int n)

{

int count = 0;

for (int i = 0; count++, i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

swap(a[i][j], a[j][i]), count += 2;

count++;

}

}

Complement：

以下赋测试程序，用于测试原程序与化简后程序计数是否相同，修改宏常量SIZE用于测试不同的n。若化简后程序与原程序同名，则其多一个int占位符用于标示。

#include<iostream>

using namespace std;

void D(int\* x, int n)

{

int count = 0;

int i = 1;

count++; //for i = 1

do {

x[i] += 2;

count++;//for x[i]+=2

i += 2;

count++; //for i+=2

}

while (count++,i <= n); //for while

i = 1;

count++; //for i = 1

while (count++,i <= (n / 2)) //for while

{

x[i] += x[i + 1];

count++; //for x[i] += x[i+1]

i++;

count++; //for i++

}

}

void SimplifiedD(int\* x, int n)

{

int count = 0;

int i = 1;

do {

x[i] += 2;

i += 2;

count += 3;

} while (i <= n); //for while

i = 1;

while (i <= (n / 2))

{

x[i] += x[i + 1];

i++;

count += 3;

}

count += 3;

}

/\*

\* Test Version for above

\*/

int D(int\* x, int n,int)

{

int count = 0;

int i = 1;

count++; //for i = 1

do {

x[i] += 2;

count++;//for x[i]+=2

i += 2;

count++; //for i+=2

} while (count++, i <= n); //for while

i = 1;

count++; //for i = 1

while (count++, i <= (n / 2)) //for while

{

x[i] += x[i + 1];

count++; //for x[i] += x[i+1]

i++;

count++; //for i++

}

return count;

}

int SimplifiedD(int\* x, int n,int)

{

int count = 0;

int i = 1;

do {

x[i] += 2;

i += 2;

count += 3;

} while (i <= n);

i = 1;

while (i <= (n / 2))

{

x[i] += x[i + 1];

i++;

count += 3;

}

count += 3;

return count;

}

int Transpose(int\*\* a, int n)

{

int counti, countj, counts;

int count;

count = counti = countj = counts = 0;

for (int i = 0; counti++,i < n - 1; i++)

{

for (int j = i + 1;countj++,j < n; j++)

swap(a[i][j], a[j][i]),counts++;

}

return count = (counti+countj+counts);

}

int Transpose(int\*\* a, int n,int)

{

int count = 0;

for (int i = 0; count++, i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

swap(a[i][j], a[j][i]), count += 2;

count++;

}

return count;

}

int main()

{

int n;

flag:

while (cin >> n)

{

//拒绝小于0非法操作

//拒绝过大数据量 免得申请失败

if (n < 0 || n > 1500) goto flag;

cout << "测试T5: " << endl;

int\*\* b = new int\* [n];

for (int i = 0; i < n; i++)

b[i] = new int[n];

cout << boolalpha; //manipulator,用于输出bool值

cout << (Transpose(b, n) == Transpose(b, n, 0)) ? true : false;

cout << endl << "测试T4: " << endl;

int size = n + 1;

int\* Array = new int[size];

//从1开始,范围1-n,传进去n,故多申请一位给索引0

D(Array, size - 1, 0) == SimplifiedD(Array, size - 1, 0) ? cout << true : cout << false;

cout << endl << endl;

}

}