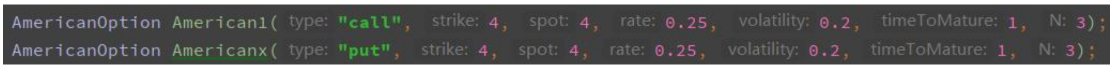
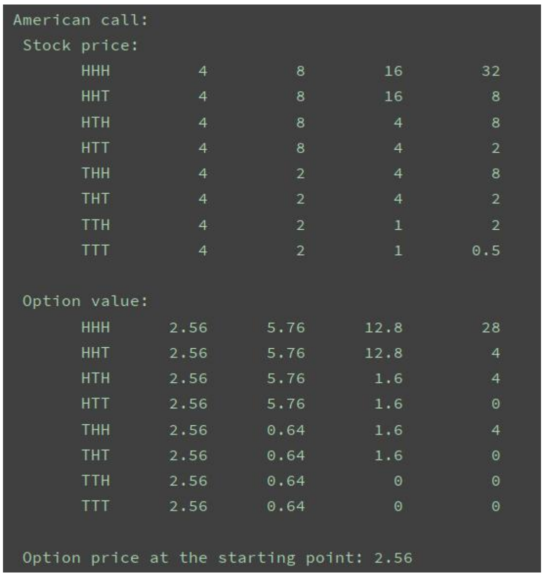
Problem 1

For this problem, I found that different methods in calculating u, d, p and q in a binomial tree model will lead to different answer. I tested my code with one homework I did for FE 543, Stochastic Calculus last semester.

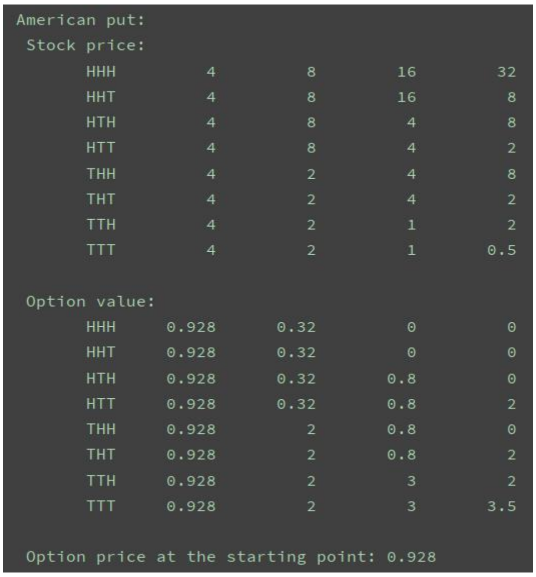


```
AmericanOption American1( type: "call",  strike: 4,  spot: 4,  rate: 0.25,  volatility: 0.2,  timeToMature: 1,  N: 3);
AmericanOption Americanx( type: "put",  strike: 4,  spot: 4,  rate: 0.25,  volatility: 0.2,  timeToMature: 1,  N: 3);
```

Above is the how I created my objects, where n stands for the terms I'll use to build my binomial tree model. If I take u = 2, d = 0.5 and get p = (1+r-d)/(u-d), q = 1-p, then below is what I have. For testing purposes, I use very simple numbers, and print out the process of price/value change.

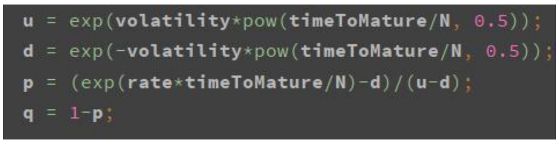


```
American call:
 Stock price:
      HHH         4          8         16         32
      HHT         4          8         16          8
      HTH         4          8          4          8
      HTT         4          8          4          2
      THH         4          2          4          8
      THT         4          2          4          2
      TTH         4          2          1          2
      TTT         4          2          1        0.5

 Option value:
      HHH      2.56       5.76       12.8         28
      HHT      2.56       5.76       12.8          4
      HTH      2.56       5.76        1.6          4
      HTT      2.56       5.76        1.6          0
      THH      2.56       0.64        1.6          4
      THT      2.56       0.64        1.6          0
      TTH      2.56       0.64          0          0
      TTT      2.56       0.64          0          0

 Option price at the starting point: 2.56
```



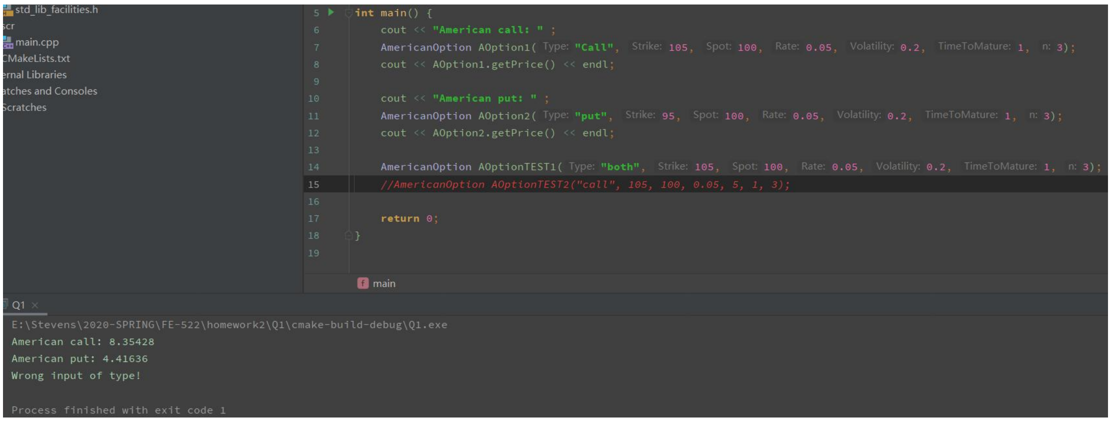
```
American put:
 Stock price:
      HHH         4          8         16         32
      HHT         4          8         16          8
      HTH         4          8          4          8
      HTT         4          8          4          2
      THH         4          2          4          8
      THT         4          2          4          2
      TTH         4          2          1          2
      TTT         4          2          1        0.5

 Option value:
      HHH     0.928       0.32          0          0
      HHT     0.928       0.32          0          0
      HTH     0.928       0.32        0.8          0
      HTT     0.928       0.32        0.8          2
      THH     0.928          2        0.8          0
      THT     0.928          2        0.8          2
      TTH     0.928          2          3          2
      TTT     0.928          2          3        3.5

 Option price at the starting point: 0.928
```

The results totally matched the answer for that homework question. In this case I'm sure my calculation is correct. Then, in order to make it more precise, I change the way to get u, d, p and q. Now the formula becomes:



```
u = exp(volatility*pow(timeToMature/N, 0.5));
d = exp(-volatility*pow(timeToMature/N, 0.5));
p = (exp(rate*timeToMature/N)-d)/(u-d);
q = 1-p;
```

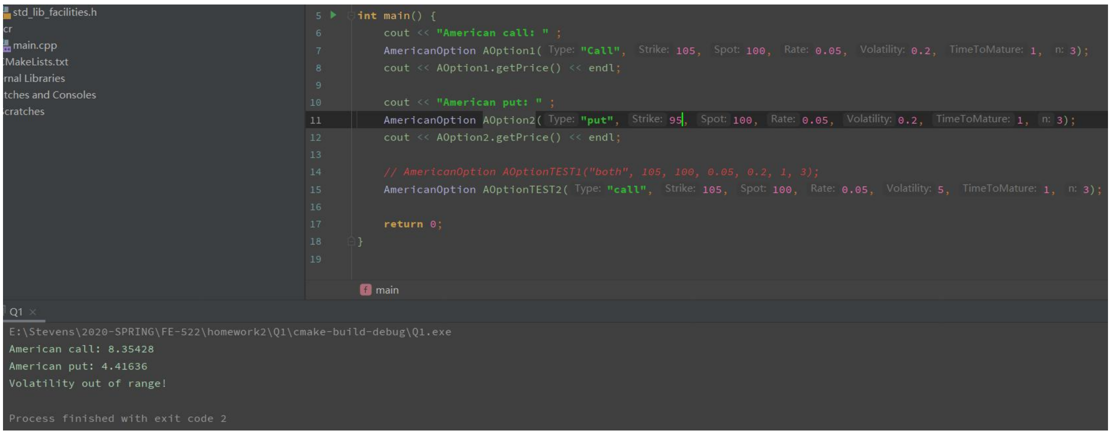
This time in order to make my result more concise, I comment the print matrix commands and only show the option price I got. Here is my objects and output. The exit code is 1 because of the illegal input of type. I also have a sentence to warn the mistake.



```
int main() {
    cout << "American call: " ;
    AmericanOption AOption1( Type: "Call",  Strike: 105,  Spot: 100,  Rate: 0.05,  Volatility: 0.2,  TimeToMature: 1,  n: 3);
    cout << AOption1.getPrice() << endl;

    cout << "American put: " ;
    AmericanOption AOption2( Type: "put",  Strike: 95,  Spot: 100,  Rate: 0.05,  Volatility: 0.2,  TimeToMature: 1,  n: 3);
    cout << AOption2.getPrice() << endl;

    AmericanOption AOptionTEST1( Type: "both",  Strike: 105,  Spot: 100,  Rate: 0.05,  Volatility: 0.2,  TimeToMature: 1,  n: 3);
    //AmericanOption AOptionTEST2("call", 105, 100, 0.05, 5, 1, 3);

    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Q1\cmake-build-debug\Q1.exe
American call: 8.35428
American put: 4.41636
Wrong input of type!

Process finished with exit code 1
```

And I won’t accept illegal input of volatility, too:

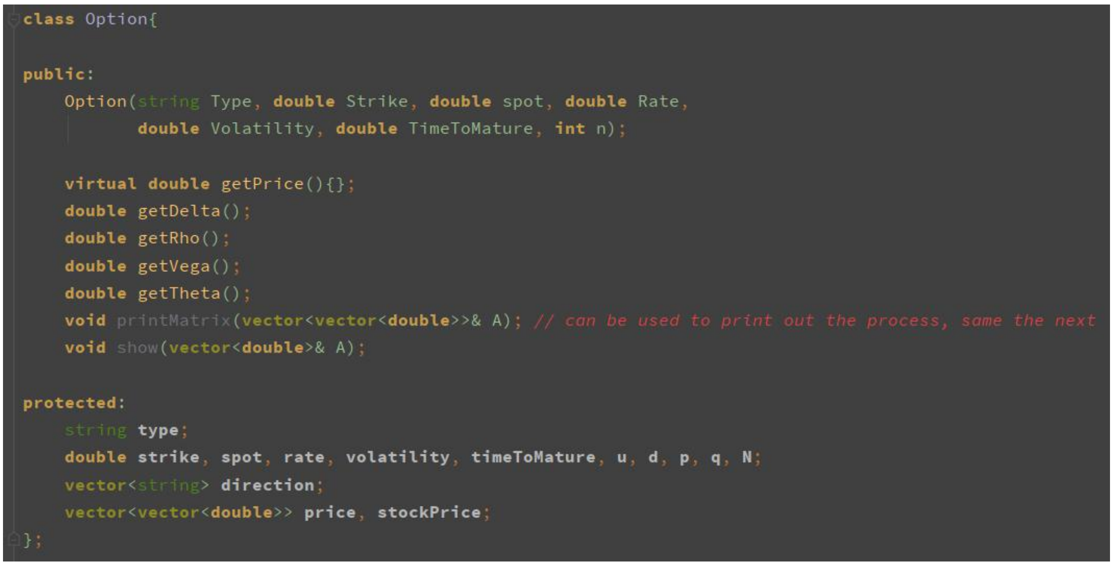


```
int main() {
    cout << "American call: " ;
    AmericanOption AOption1( Type: "Call",  Strike: 105,  Spot: 100,  Rate: 0.05,  Volatility: 0.2,  TimeToMature: 1,  n: 3);
    cout << AOption1.getPrice() << endl;

    cout << "American put: " ;
    AmericanOption AOption2( Type: "put",  Strike: 95,  Spot: 100,  Rate: 0.05,  Volatility: 0.2,  TimeToMature: 1,  n: 3);
    cout << AOption2.getPrice() << endl;

    // AmericanOption AOptionTEST1("both", 105, 100, 0.05, 0.2, 1, 3);
    AmericanOption AOptionTEST2( Type: "call",  Strike: 105,  Spot: 100,  Rate: 0.05,  Volatility: 5,  TimeToMature: 1,  n: 3);

    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Q1\cmake-build-debug\Q1.exe
American call: 8.35428
American put: 4.41636
Volatility out of range!

Process finished with exit code 2
```

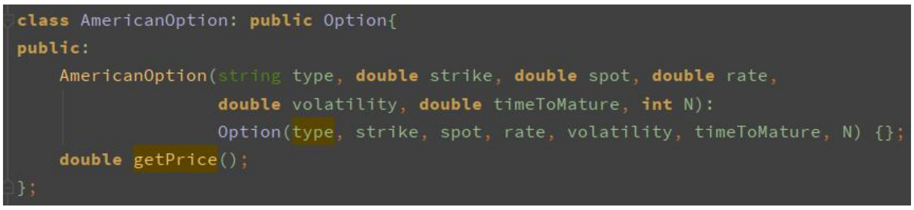
Problem 2

This is the structural of my Option class. I’ll use binomial tree for both European and American options.

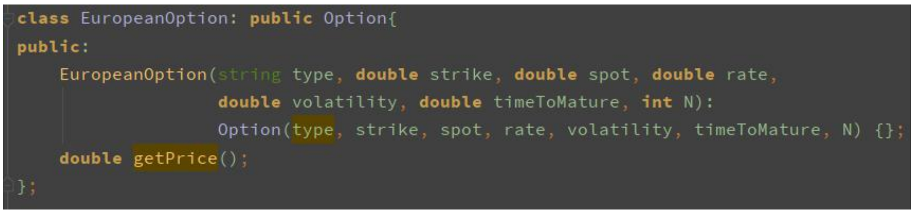


```
class Option{

public:
    Option(string Type, double Strike, double spot, double Rate,
           double Volatility, double TimeToMature, int n);

    virtual double getPrice(){};
    double getDelta();
    double getRho();
    double getVega();
    double getTheta();
    void printMatrix(vector<vector<double>>& A); // can be used to print out the process, same the next
    void show(vector<double>& A);

protected:
    string type;
    double strike, spot, rate, volatility, timeToMature, u, d, p, q, N;
    vector<string> direction;
    vector<vector<double>> price, stockPrice;
};
```

And here are the structural of my inherited classes:

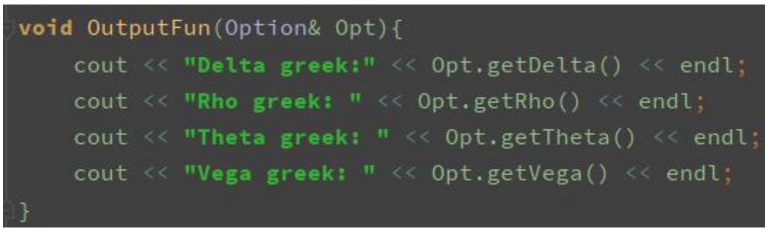


```
class AmericanOption: public Option{
public:
    AmericanOption(string type, double strike, double spot, double rate,
                   double volatility, double timeToMature, int N):
                   Option(type, strike, spot, rate, volatility, timeToMature, N) {};
    double getPrice();
};
```



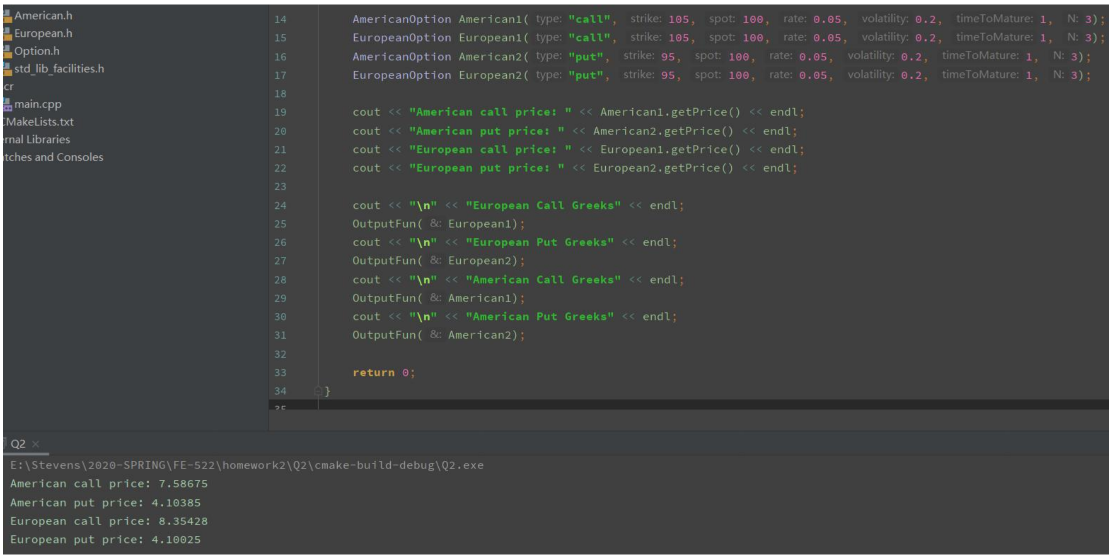
```
class EuropeanOption: public Option{
public:
    EuropeanOption(string type, double strike, double spot, double rate,
                   double volatility, double timeToMature, int N):
                   Option(type, strike, spot, rate, volatility, timeToMature, N) {};
    double getPrice();
};
```

This is my external function to print out Greeks.



```cpp
void OutputFun(Option& Opt){
    cout << "Delta greek:" << Opt.getDelta() << endl;
    cout << "Rho greek: " << Opt.getRho() << endl;
    cout << "Theta greek: " << Opt.getTheta() << endl;
    cout << "Vega greek: " << Opt.getVega() << endl;
}
```

Then I create four objects, standing for four pairs of type.

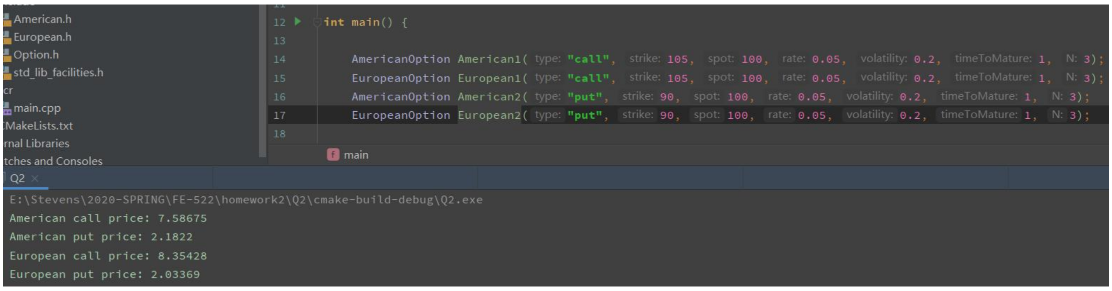


```cpp
    AmericanOption American1( type: "call",  strike: 105,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);
    EuropeanOption European1( type: "call",  strike: 105,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);
    AmericanOption American2( type: "put",  strike: 95,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);
    EuropeanOption European2( type: "put",  strike: 95,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);

    cout << "American call price: " << American1.getPrice() << endl;
    cout << "American put price: " << American2.getPrice() << endl;
    cout << "European call price: " << European1.getPrice() << endl;
    cout << "European put price: " << European2.getPrice() << endl;

    cout << "\n" << "European Call Greeks" << endl;
    OutputFun( &: European1);
    cout << "\n" << "European Put Greeks" << endl;
    OutputFun( &: European2);
    cout << "\n" << "American Call Greeks" << endl;
    OutputFun( &: American1);
    cout << "\n" << "American Put Greeks" << endl;
    OutputFun( &: American2);

    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Q2\cmake-build-debug\Q2.exe
American call price: 7.58675
American put price: 4.10385
European call price: 8.35428
European put price: 4.10025
```

We see here, American call and European call have different price, but two put options seem to share the same price.

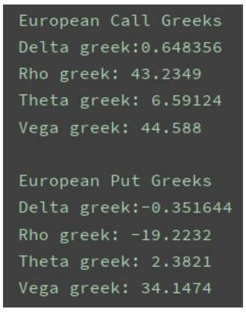
I think whether the prices of American and European options are different or not depends on how far away strike price is to spot price. If I change my input of put options' strike, we could get different prices for American and European type.



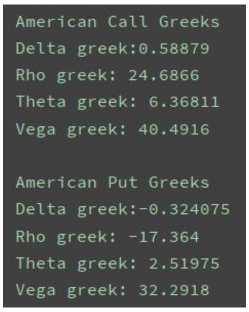
```cpp
int main() {

    AmericanOption American1( type: "call",  strike: 105,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);
    EuropeanOption European1( type: "call",  strike: 105,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);
    AmericanOption American2( type: "put",  strike: 90,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);
    EuropeanOption European2( type: "put",  strike: 90,  spot: 100,  rate: 0.05,  volatility: 0.2,  timeToMature: 1,  N: 3);

```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Q2\cmake-build-debug\Q2.exe
American call price: 7.58675
American put price: 2.1822
European call price: 8.35428
European put price: 2.03369
```

And here is my output is Greeks. For Delta, my bump is 0.1 and for other 3 I set bump = 0.01.



```
European Call Greeks
Delta greek:0.648356
Rho greek: 43.2349
Theta greek: 6.59124
Vega greek: 44.588

European Put Greeks
Delta greek:-0.351644
Rho greek: -19.2232
Theta greek: 2.3821
Vega greek: 34.1474
```



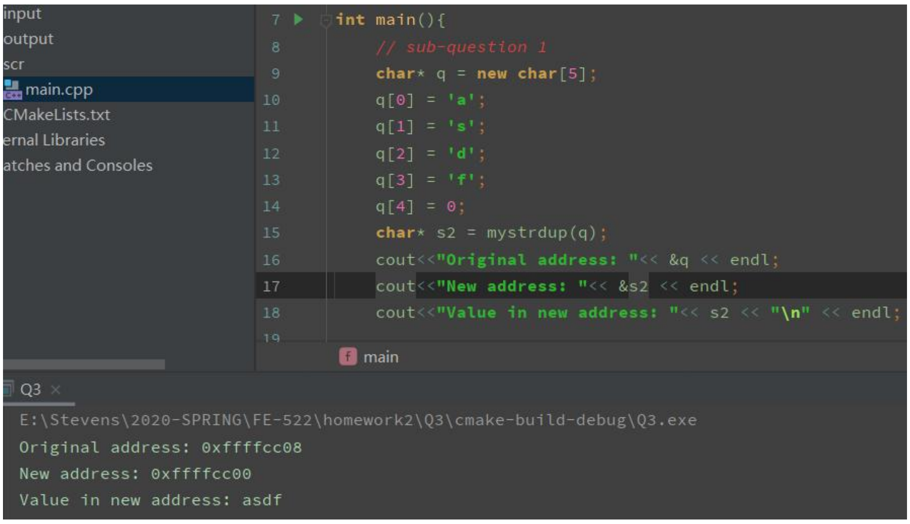
```
American Call Greeks
Delta greek:0.58879
Rho greek: 24.6866
Theta greek: 6.36811
Vega greek: 40.4916

American Put Greeks
Delta greek:-0.324075
Rho greek: -17.364
Theta greek: 2.51975
Vega greek: 32.2918
```

Compare these output we could see, corresponding Greeks of American and European call options are similar in magnitude. Same for put options. And for this pair of options, European option prices are more sensitive to parameters delta, rho, theta and vega.

Problem 3

Sub-questions 1:

If I understand correctly, we are asked to copy the value stored at the address of the input pointer, and assign this value to a new pointer pointing to a new address. In this case, I print out the address of input, the value stored with output pointer as well as the address of the output pointer.

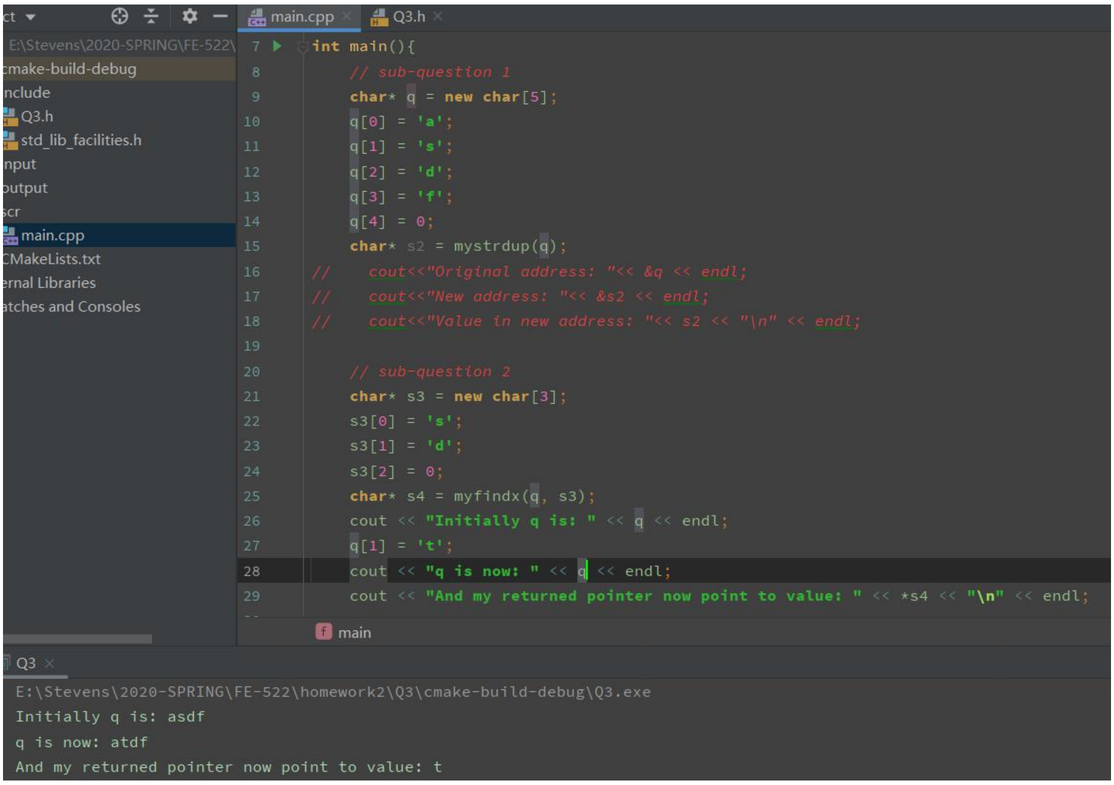


```
int main(){
    // sub-question 1
    char* q = new char[5];
    q[0] = 'a';
    q[1] = 's';
    q[2] = 'd';
    q[3] = 'f';
    q[4] = 0;
    char* s2 = mystrdup(q);
    cout<<"Original address: "<< &q << endl;
    cout<<"New address: "<< &s2 << endl;
    cout<<"Value in new address: "<< s2 << "\n" << endl;
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Q3\cmake-build-debug\Q3.exe
Original address: 0xffffcc08
New address: 0xffffcc00
Value in new address: asdf
```

Sub-questions 2:

To test my function for this problem, I assign the returned pointer to “s4”. Then I changed the value at the first occurrence from “s” to “t”, then print out what my “s4” points to.



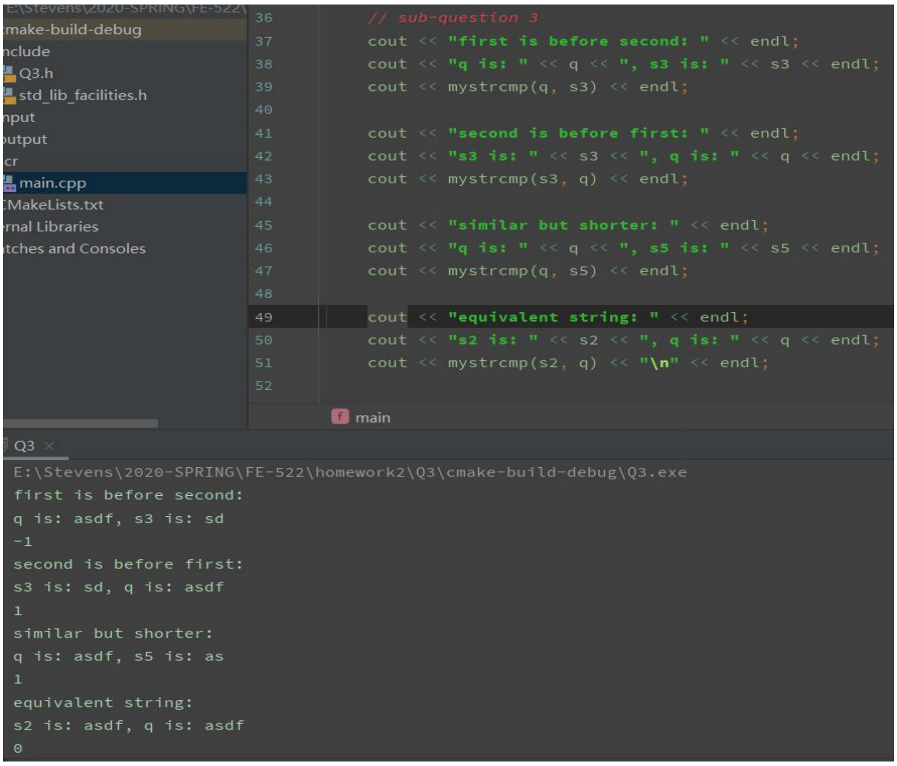
```
int main(){
    // sub-question 1
    char* q = new char[5];
    q[0] = 'a';
    q[1] = 's';
    q[2] = 'd';
    q[3] = 'f';
    q[4] = 0;
    char* s2 = mystrdup(q);
//    cout<<"Original address: "<< &q << endl;
//    cout<<"New address: "<< &s2 << endl;
//    cout<<"Value in new address: "<< s2 << "\n" << endl;

    // sub-question 2
    char* s3 = new char[3];
    s3[0] = 's';
    s3[1] = 'd';
    s3[2] = 0;
    char* s4 = myfindx(q, s3);
    cout << "Initially q is: " << q << endl;
    q[1] = 't';
    cout << "q is now: " << q << endl;
    cout << "And my returned pointer now point to value: " << *s4 << "\n" << endl;
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Q3\cmake-build-debug\Q3.exe
Initially q is: asdf
q is now: atdf
And my returned pointer now point to value: t
```

When I change the value at q[1], which is the first occurrence point, the value that my output pointer “s4” pointing to changes correspondingly. In this case I think I meet the requirement.

Sub-questions 3:

For this problem, I use ASCII code to judge lexicographical order.I compare ASCII code of corresponding characters in 2 strings one by one. Once there is a difference, I have a return. If they're exactly the same, finally I'll return a 0. I also include the condition that one of the string is the same as the first part of the other, and is shorter.



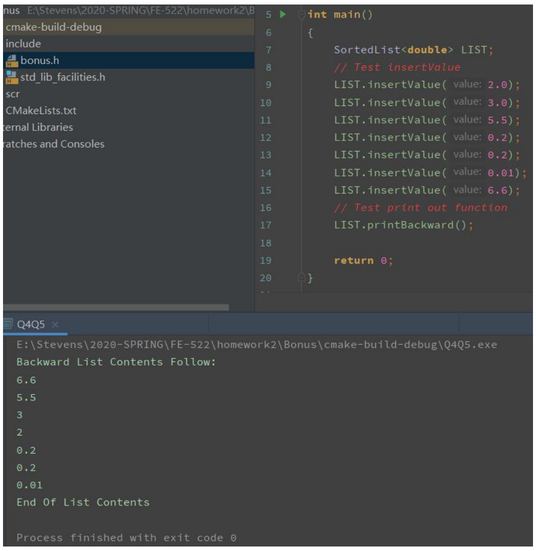
```
    // sub-question 3
    cout << "first is before second: " << endl;
    cout << "q is: " << q << ", s3 is: " << s3 << endl;
    cout << mystrcmp(q, s3) << endl;

    cout << "second is before first: " << endl;
    cout << "s3 is: " << s3 << ", q is: " << q << endl;
    cout << mystrcmp(s3, q) << endl;

    cout << "similar but shorter: " << endl;
    cout << "q is: " << q << ", s5 is: " << s5 << endl;
    cout << mystrcmp(q, s5) << endl;

    cout << "equivalent string: " << endl;
    cout << "s2 is: " << s2 << ", q is: " << q << endl;
    cout << mystrcmp(s2, q) << "\n" << endl;
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Q3\cmake-build-debug\Q3.exe
first is before second:
q is: asdf, s3 is: sd
-1
second is before first:
s3 is: sd, q is: asdf
1
similar but shorter:
q is: asdf, s5 is: as
1
equivalent string:
s2 is: asdf, q is: asdf
0
```

Bonus

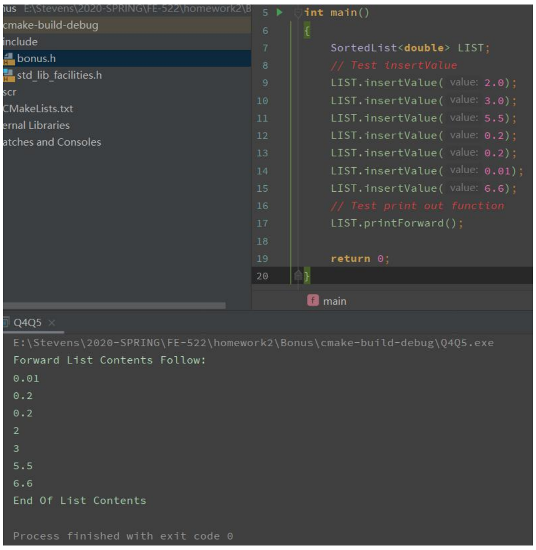
Basically, problem 4 is just a prat of Bonus. As long as I did bonus well, I did problem 4 well. Therefore I put Bonus first. To save time, you could skip the Problem 4 part.

Test insert and print functions:



```
int main()
{
    SortedList<double> LIST;
    // Test insertValue
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5.5);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.01);
    LIST.insertValue( value: 6.6);
    // Test print out function
    LIST.printBackward();

    return 0;
}
```

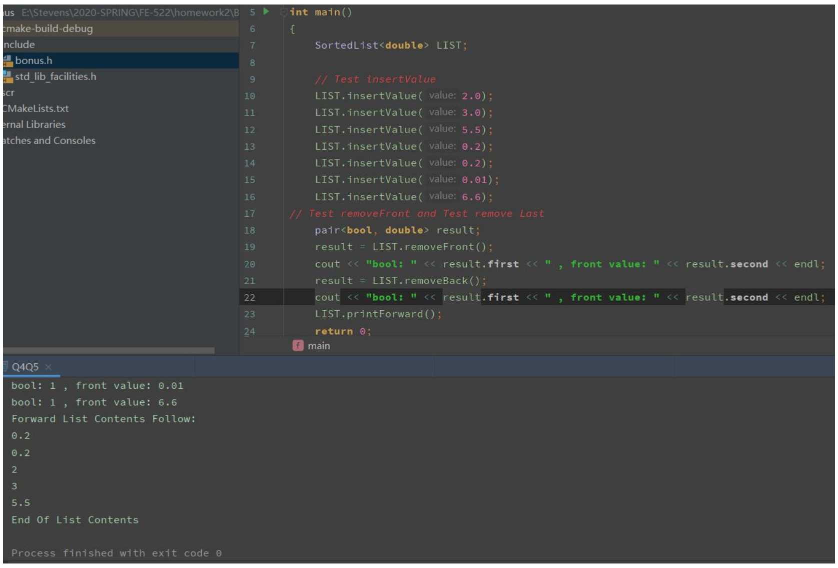
```
E:\Stevens\2020-SPRING\FE-522\homework2\Bonus\cmake-build-debug\Q4Q5.exe
Backward List Contents Follow:
6.6
5.5
3
2
0.2
0.2
0.01
End Of List Contents

Process finished with exit code 0
```



```
int main()
{
    SortedList<double> LIST;
    // Test insertValue
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5.5);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.01);
    LIST.insertValue( value: 6.6);
    // Test print out function
    LIST.printForward();

    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Bonus\cmake-build-debug\Q4Q5.exe
Forward List Contents Follow:
0.01
0.2
0.2
2
3
5.5
6.6
End Of List Contents

Process finished with exit code 0
```

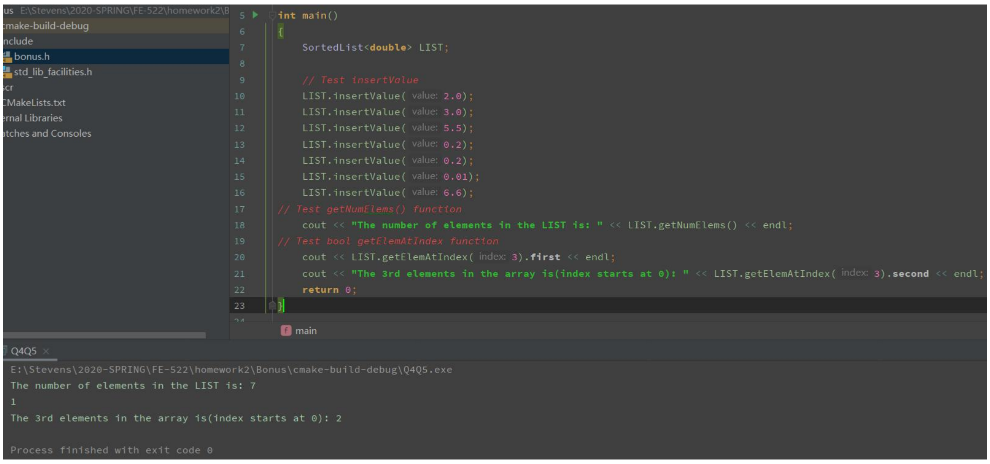
Then I test remove Front and Test remove Last functions. Now there are only 5 elements left.



```
int main()
{
    SortedList<double> LIST;

    // Test insertValue
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5.5);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.01);
    LIST.insertValue( value: 6.6);
// Test removeFront and Test remove Last
    pair<bool, double> result;
    result = LIST.removeFront();
    cout << "bool: " << result.first << " , front value: " << result.second << endl;
    result = LIST.removeBack();
    cout << "bool: " << result.first << " , front value: " << result.second << endl;
    LIST.printForward();
    return 0;
```

```
bool: 1 , front value: 0.01
bool: 1 , front value: 6.6
Forward List Contents Follow:
0.2
0.2
2
3
5.5
End Of List Contents

Process finished with exit code 0
```

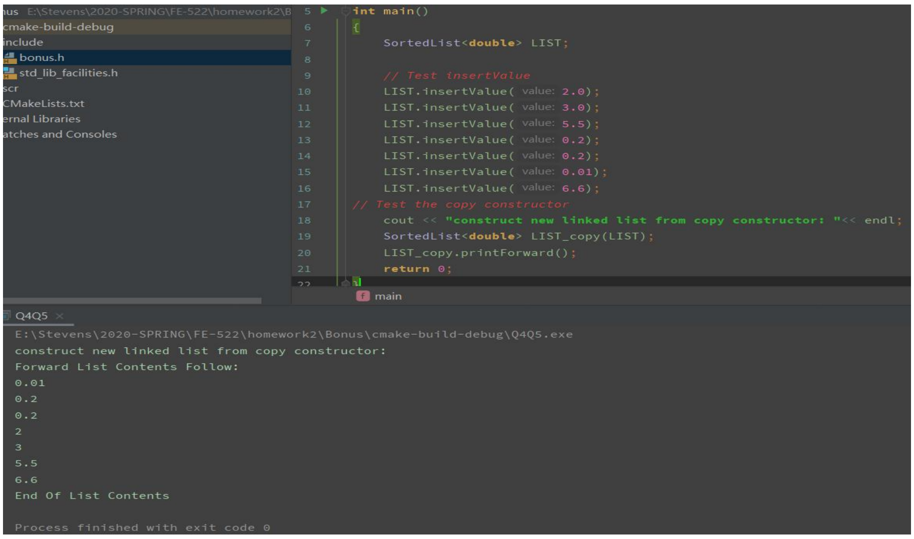
Here is the getNumElems function and the getElemAtIndex function:



```
int main()
{
    SortedList<double> LIST;

    // Test insertValue
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5.5);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.01);
    LIST.insertValue( value: 6.6);
// Test getNumElems() function
    cout << "The number of elements in the LIST is: " << LIST.getNumElems() << endl;
// Test bool getElemAtIndex function
    cout << LIST.getElemAtIndex( index: 3).first << endl;
    cout << "The 3rd elements in the array is(index starts at 0): " << LIST.getElemAtIndex( index: 3).second << endl;
    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Bonus\cmake-build-debug\Q4Q5.exe
The number of elements in the LIST is: 7
1
The 3rd elements in the array is(index starts at 0): 2

Process finished with exit code 0
```

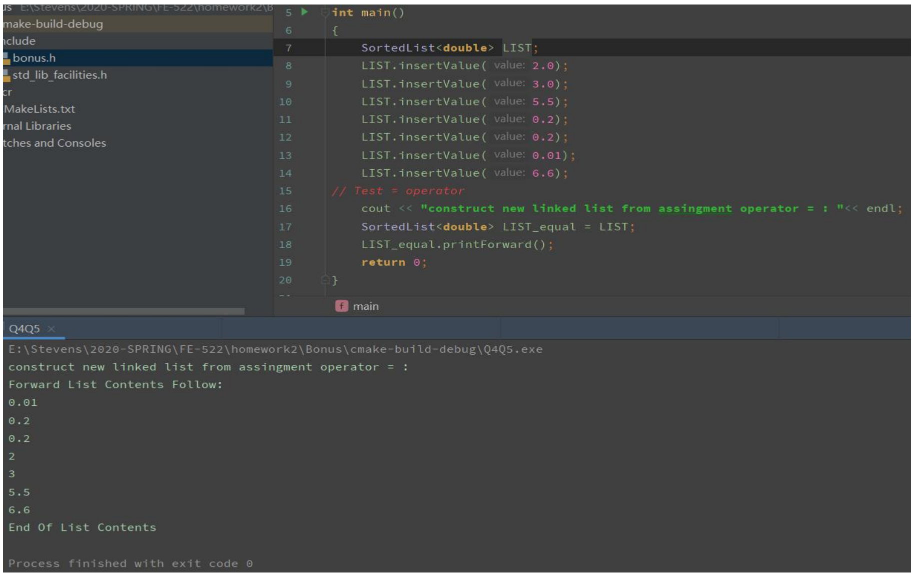
Test copy constructor:



```
int main()
{
    SortedList<double> LIST;

    // Test insertValue
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5.5);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.01);
    LIST.insertValue( value: 6.6);
// Test the copy constructor
    cout << "construct new linked list from copy constructor: "<< endl;
    SortedList<double> LIST_copy(LIST);
    LIST_copy.printForward();
    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Bonus\cmake-build-debug\Q4Q5.exe
construct new linked list from copy constructor:
Forward List Contents Follow:
0.01
0.2
0.2
2
3
5.5
6.6
End Of List Contents

Process finished with exit code 0
```

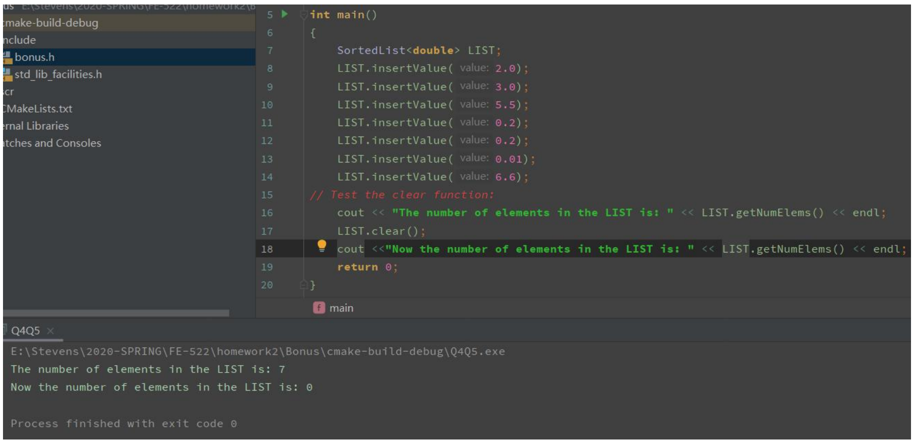
And the = operator:



```
int main()
{
    SortedList<double> LIST;
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5.5);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.01);
    LIST.insertValue( value: 6.6);
// Test = operator
    cout << "construct new linked list from assingment operator = : "<< endl;
    SortedList<double> LIST_equal = LIST;
    LIST_equal.printForward();
    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Bonus\cmake-build-debug\Q4Q5.exe
construct new linked list from assingment operator = :
Forward List Contents Follow:
0.01
0.2
0.2
2
3
5.5
6.6
End Of List Contents

Process finished with exit code 0
```

Finally it’s the clear function:

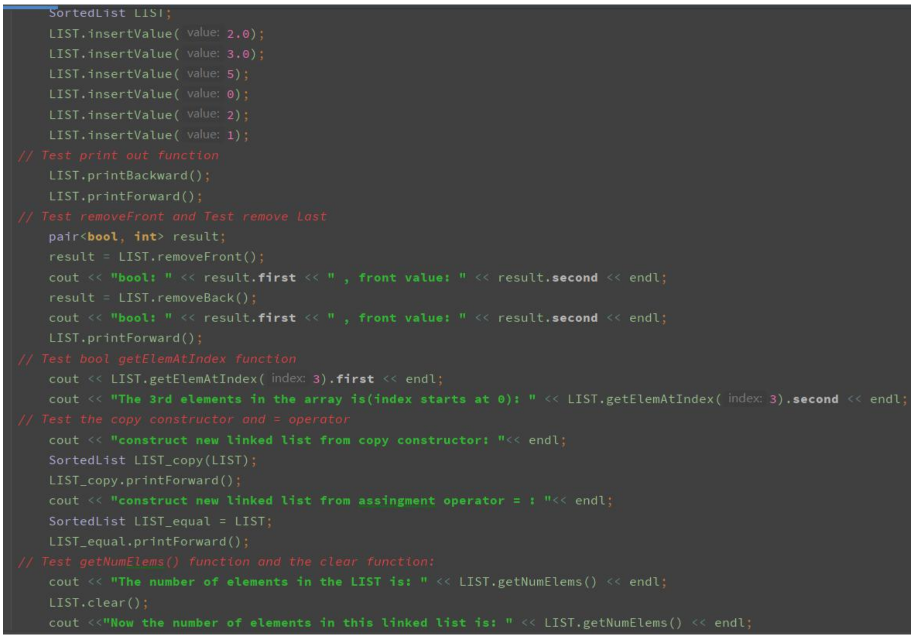


```
int main()
{
    SortedList<double> LIST;
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5.5);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.2);
    LIST.insertValue( value: 0.01);
    LIST.insertValue( value: 6.6);
// Test the clear function:
    cout << "The number of elements in the LIST is: " << LIST.getNumElems() << endl;
    LIST.clear();
    cout <<"Now the number of elements in the LIST is: " << LIST.getNumElems() << endl;
    return 0;
}
```

```
E:\Stevens\2020-SPRING\FE-522\homework2\Bonus\cmake-build-debug\Q4Q5.exe
The number of elements in the LIST is: 7
Now the number of elements in the LIST is: 0

Process finished with exit code 0
```

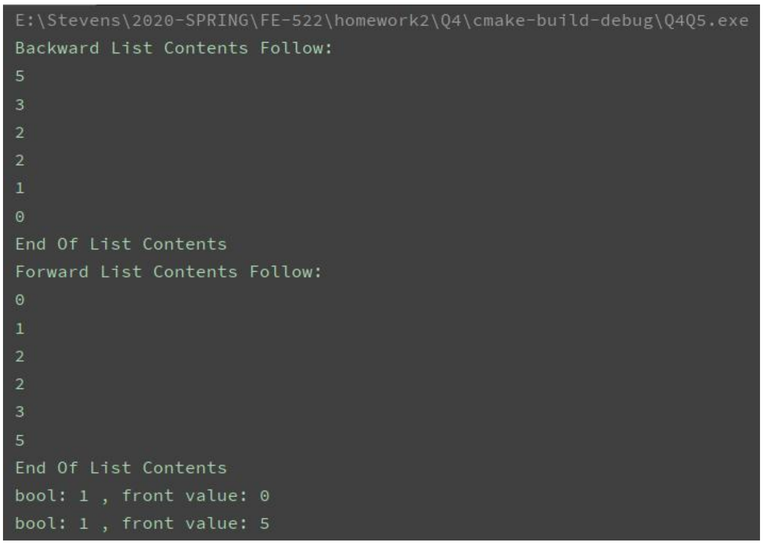
Problem 4

As long as I did bonus well, I must did this problem well. So I’ll just briefly report it:

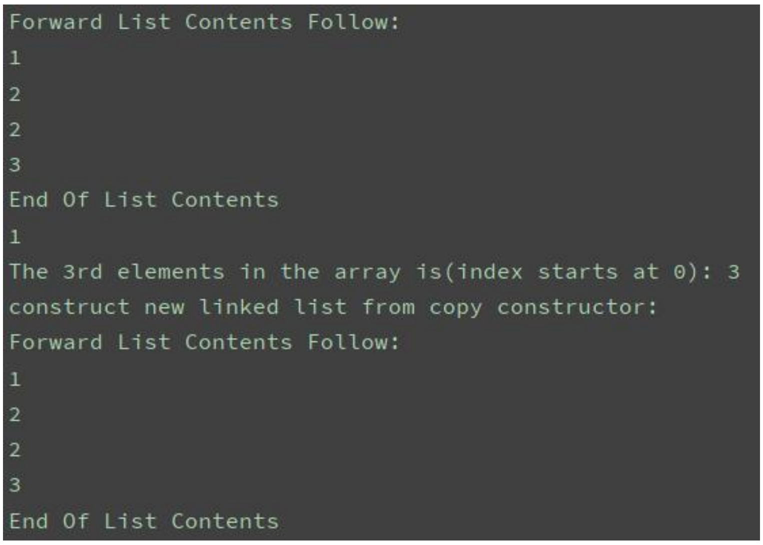


```
    SortedList LIST;
    LIST.insertValue( value: 2.0);
    LIST.insertValue( value: 3.0);
    LIST.insertValue( value: 5);
    LIST.insertValue( value: 0);
    LIST.insertValue( value: 2);
    LIST.insertValue( value: 1);
// Test print out function
    LIST.printBackward();
    LIST.printForward();
// Test removeFront and Test remove Last
    pair<bool, int> result;
    result = LIST.removeFront();
    cout << "bool: " << result.first << " , front value: " << result.second << endl;
    result = LIST.removeBack();
    cout << "bool: " << result.first << " , front value: " << result.second << endl;
    LIST.printForward();
// Test bool getElemAtIndex function
    cout << LIST.getElemAtIndex( index: 3).first << endl;
    cout << "The 3rd elements in the array is(index starts at 0): " << LIST.getElemAtIndex( index: 3).second << endl;
// Test the copy constructor and = operator
    cout << "construct new linked list from copy constructor: "<< endl;
    SortedList LIST_copy(LIST);
    LIST_copy.printForward();
    cout << "construct new linked list from assingment operator = : "<< endl;
    SortedList LIST_equal = LIST;
    LIST_equal.printForward();
// Test getNumElems() function and the clear function:
    cout << "The number of elements in the LIST is: " << LIST.getNumElems() << endl;
    LIST.clear();
    cout <<"Now the number of elements in this linked list is: " << LIST.getNumElems() << endl;
```

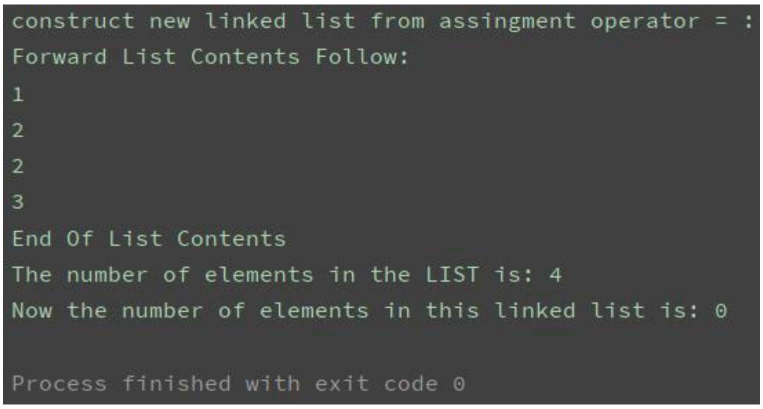
Above is my test code, and then comes output:



```
E:\Stevens\2020-SPRING\FE-522\homework2\Q4\cmake-build-debug\Q4Q5.exe
Backward List Contents Follow:
5
3
2
2
1
0
End Of List Contents
Forward List Contents Follow:
0
1
2
2
3
5
End Of List Contents
bool: 1 , front value: 0
bool: 1 , front value: 5
```



```
Forward List Contents Follow:
1
2
2
3
End Of List Contents
1
The 3rd elements in the array is(index starts at 0): 3
construct new linked list from copy constructor:
Forward List Contents Follow:
1
2
2
3
End Of List Contents
```



```
construct new linked list from assingment operator = :
Forward List Contents Follow:
1
2
2
3
End Of List Contents
The number of elements in the LIST is: 4
Now the number of elements in this linked list is: 0

Process finished with exit code 0
```