Assignment

Lab #3

Course

CIS-7

Section

27168

Due Date

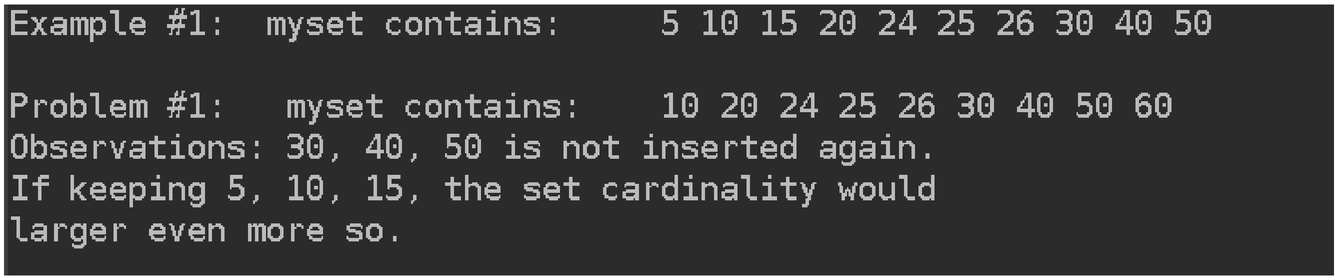
September 12th, 2021

Author

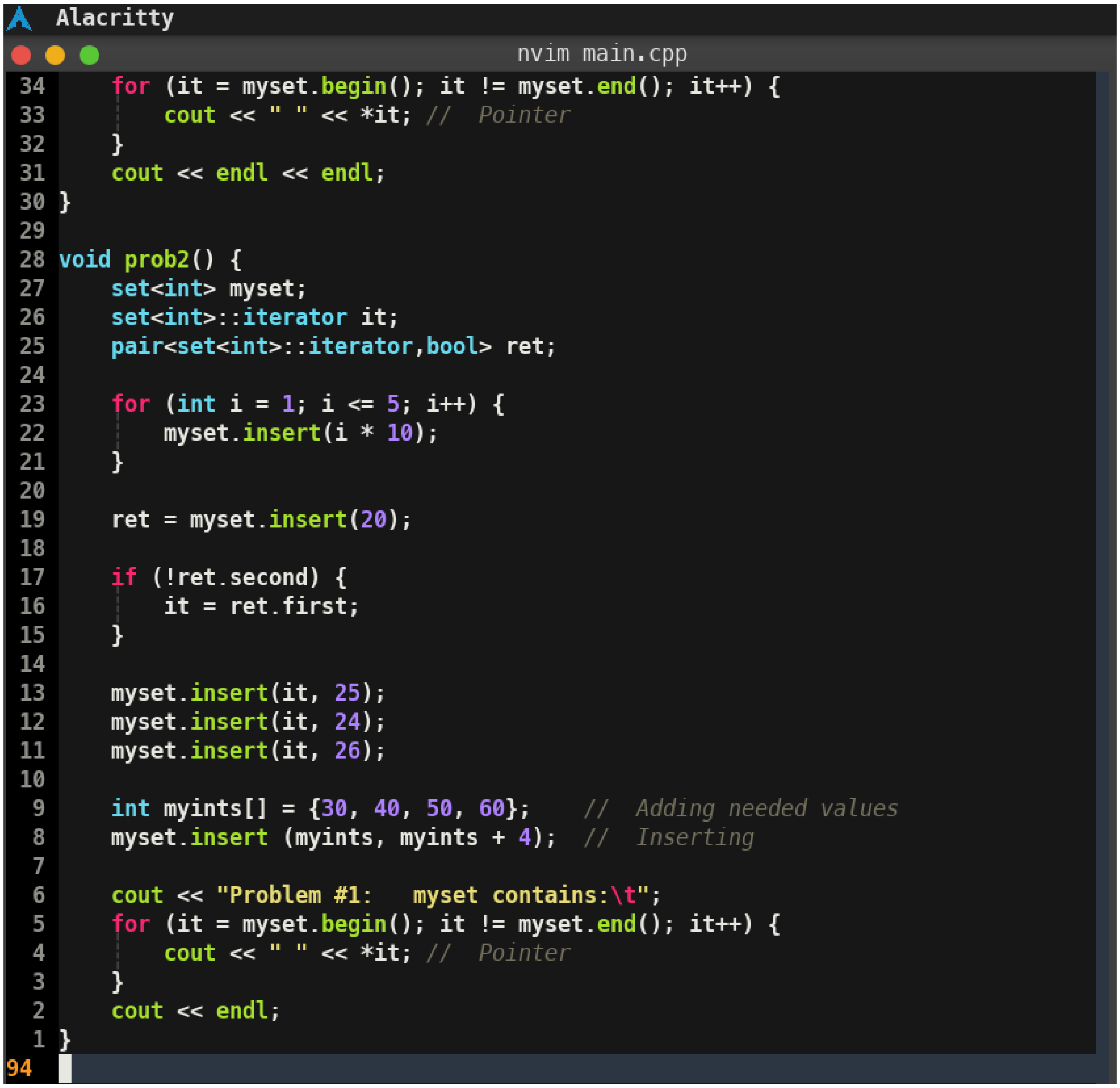
Matthew Mickey

Github repo

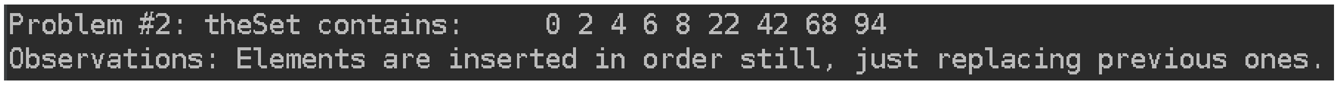
https://github.com/PocketPiggy/MVC_Fall_CIS-7/tree/master/Labs/Lab3-Sets



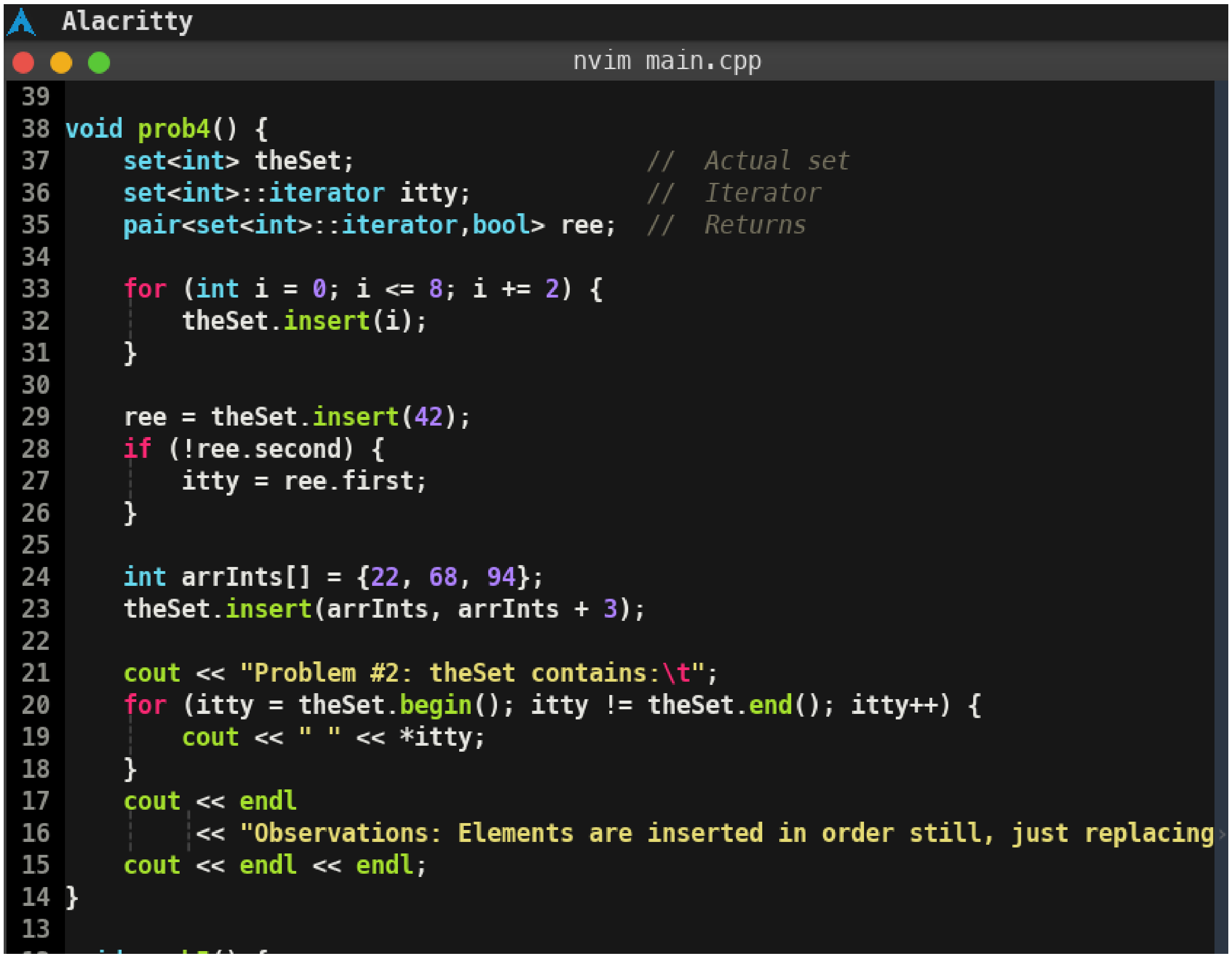
```
Example #1:  myset contains:     5 10 15 20 24 25 26 30 40 50

Problem #1:   myset contains:    10 20 24 25 26 30 40 50 60
Observations: 30, 40, 50 is not inserted again.
If keeping 5, 10, 15, the set cardinality would
larger even more so.
```



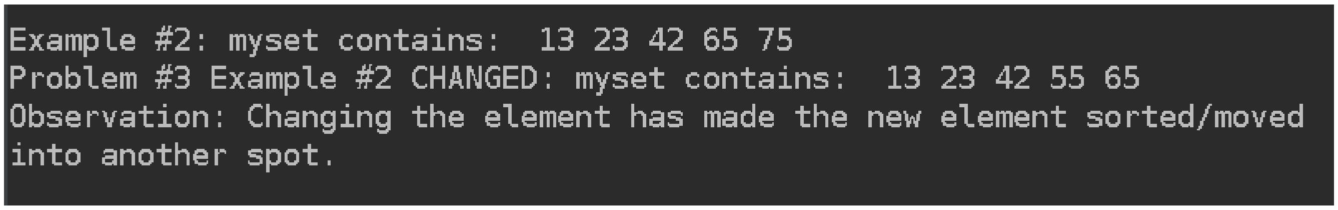
```cpp
    for (it = myset.begin(); it != myset.end(); it++) {
        cout << " " << *it; //  Pointer
    }
    cout << endl << endl;
}

void prob2() {
    set<int> myset;
    set<int>::iterator it;
    pair<set<int>::iterator,bool> ret;

    for (int i = 1; i <= 5; i++) {
        myset.insert(i * 10);
    }

    ret = myset.insert(20);

    if (!ret.second) {
        it = ret.first;
    }

    myset.insert(it, 25);
    myset.insert(it, 24);
    myset.insert(it, 26);

    int myints[] = {30, 40, 50, 60};    //  Adding needed values
    myset.insert (myints, myints + 4);  //  Inserting

    cout << "Problem #1:   myset contains:\t";
    for (it = myset.begin(); it != myset.end(); it++) {
        cout << " " << *it; //  Pointer
    }
    cout << endl;
}
```



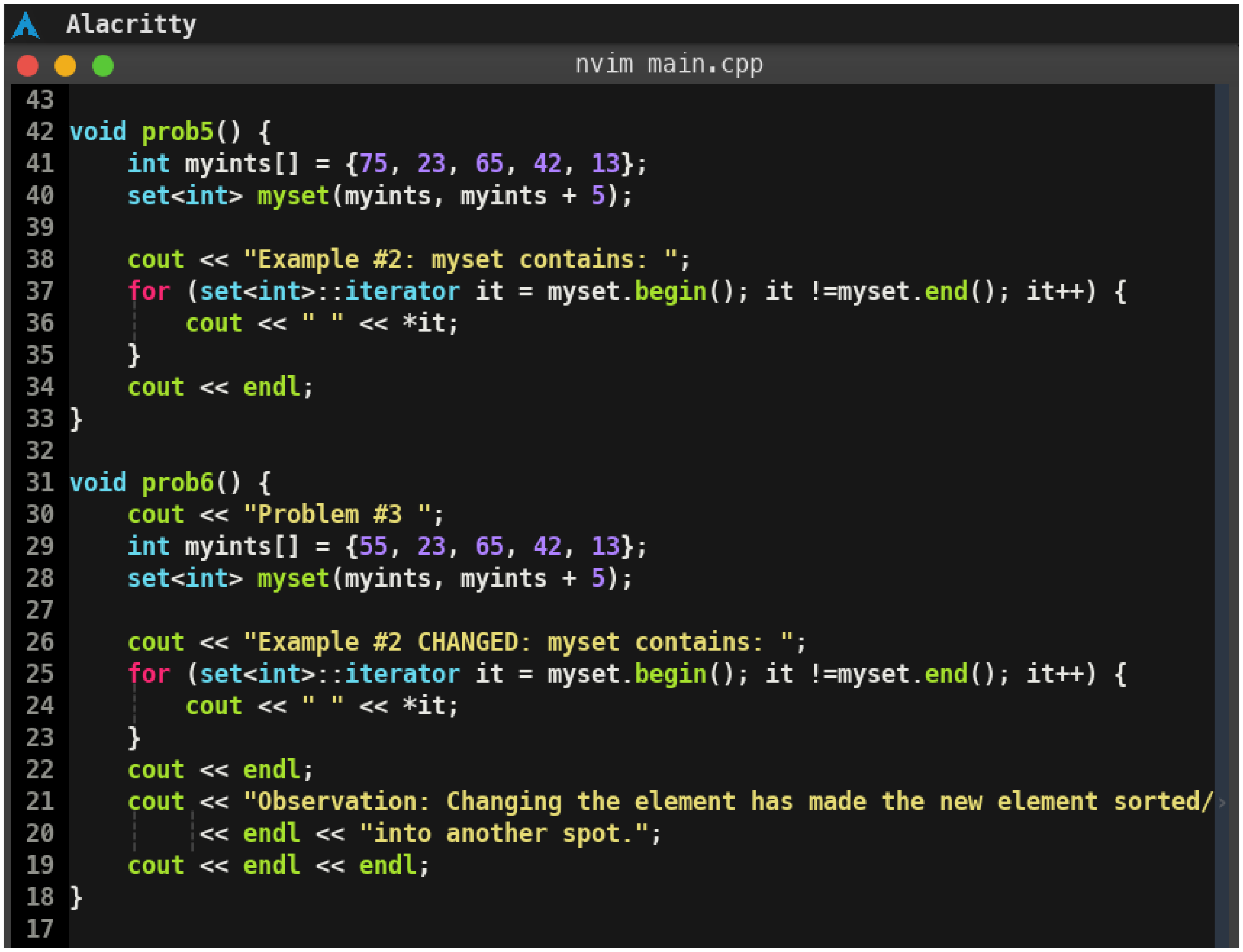
```
Problem #2: theSet contains:      0 2 4 6 8 22 42 68 94
Observations: Elements are inserted in order still, just replacing previous ones.
```



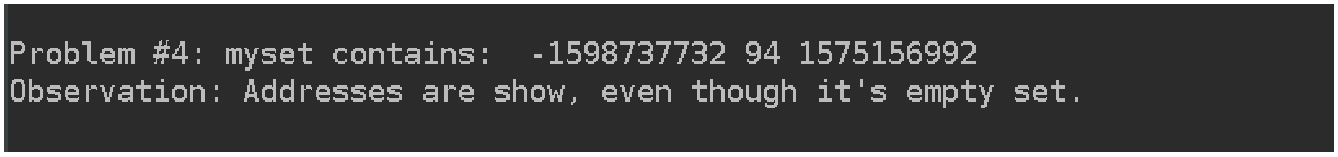
```cpp

void prob4() {
    set<int> theSet;                    //  Actual set
    set<int>::iterator itty;            //  Iterator
    pair<set<int>::iterator,bool> ree;  //  Returns

    for (int i = 0; i <= 8; i += 2) {
        theSet.insert(i);
    }

    ree = theSet.insert(42);
    if (!ree.second) {
        itty = ree.first;
    }

    int arrInts[] = {22, 68, 94};
    theSet.insert(arrInts, arrInts + 3);

    cout << "Problem #2: theSet contains:\t";
    for (itty = theSet.begin(); itty != theSet.end(); itty++) {
        cout << " " << *itty;
    }
    cout << endl
        << "Observations: Elements are inserted in order still, just replacing›
    cout << endl << endl;
}

```



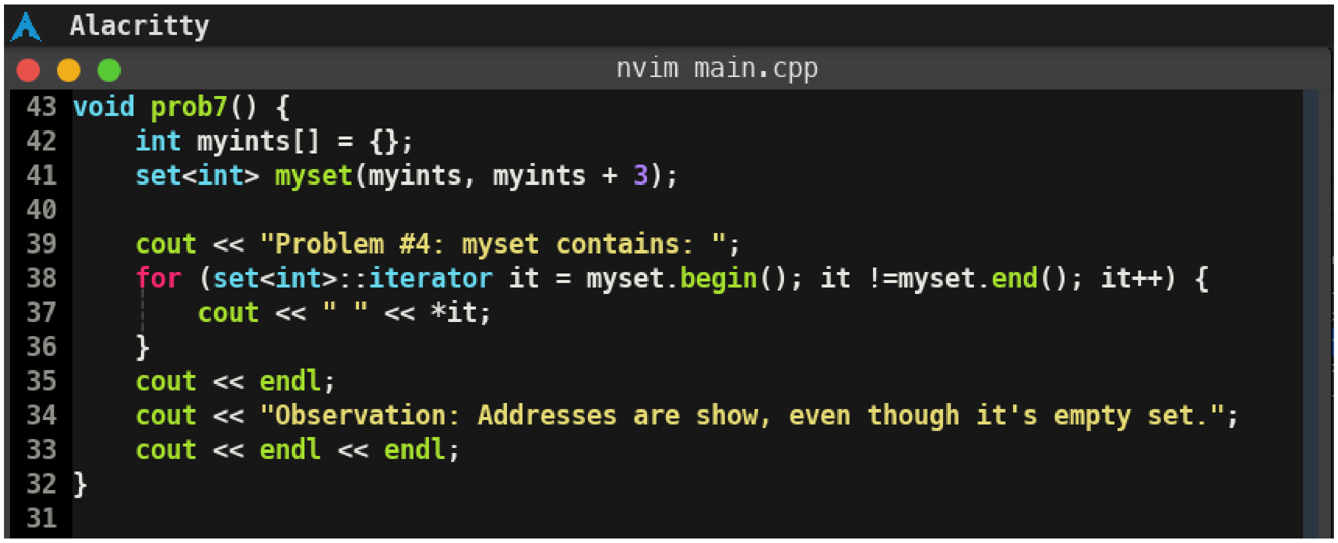
```
Example #2: myset contains:  13 23 42 65 75
Problem #3 Example #2 CHANGED: myset contains:  13 23 42 55 65
Observation: Changing the element has made the new element sorted/moved
into another spot.
```



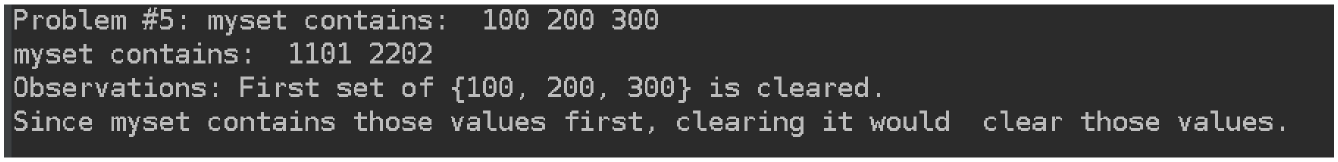
```cpp
void prob5() {
    int myints[] = {75, 23, 65, 42, 13};
    set<int> myset(myints, myints + 5);

    cout << "Example #2: myset contains: ";
    for (set<int>::iterator it = myset.begin(); it !=myset.end(); it++) {
        cout << " " << *it;
    }
    cout << endl;
}

void prob6() {
    cout << "Problem #3 ";
    int myints[] = {55, 23, 65, 42, 13};
    set<int> myset(myints, myints + 5);

    cout << "Example #2 CHANGED: myset contains: ";
    for (set<int>::iterator it = myset.begin(); it !=myset.end(); it++) {
        cout << " " << *it;
    }
    cout << endl;
    cout << "Observation: Changing the element has made the new element sorted/
         << endl << "into another spot.";
    cout << endl << endl;
}
```



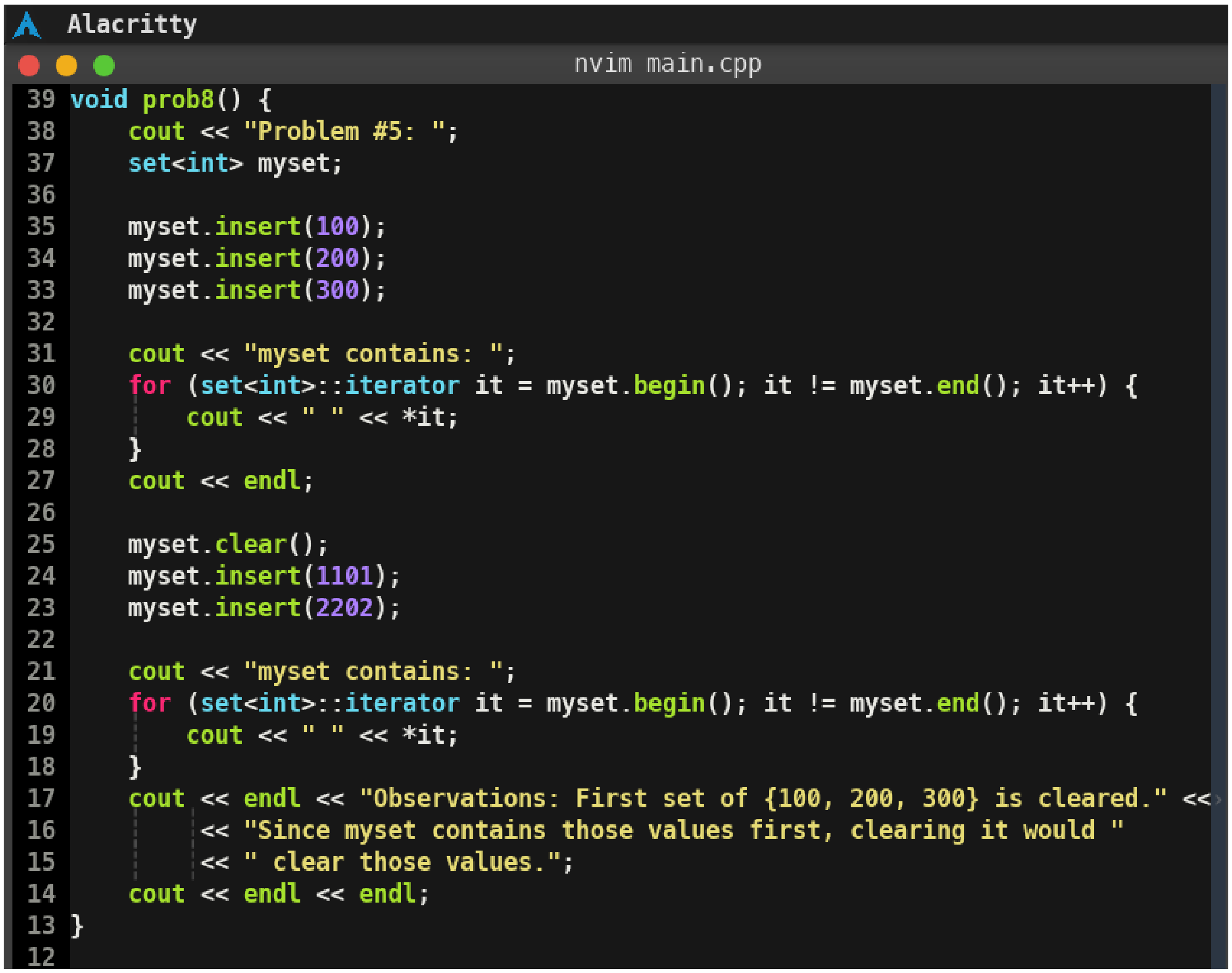
```
Problem #4: myset contains:  -1598737732 94 1575156992
Observation: Addresses are show, even though it's empty set.
```



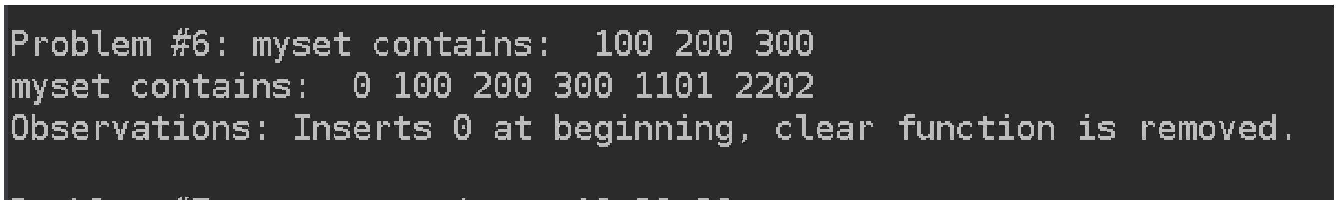
```cpp
void prob7() {
    int myints[] = {};
    set<int> myset(myints, myints + 3);

    cout << "Problem #4: myset contains: ";
    for (set<int>::iterator it = myset.begin(); it !=myset.end(); it++) {
        cout << " " << *it;
    }
    cout << endl;
    cout << "Observation: Addresses are show, even though it's empty set.";
    cout << endl << endl;
}
```



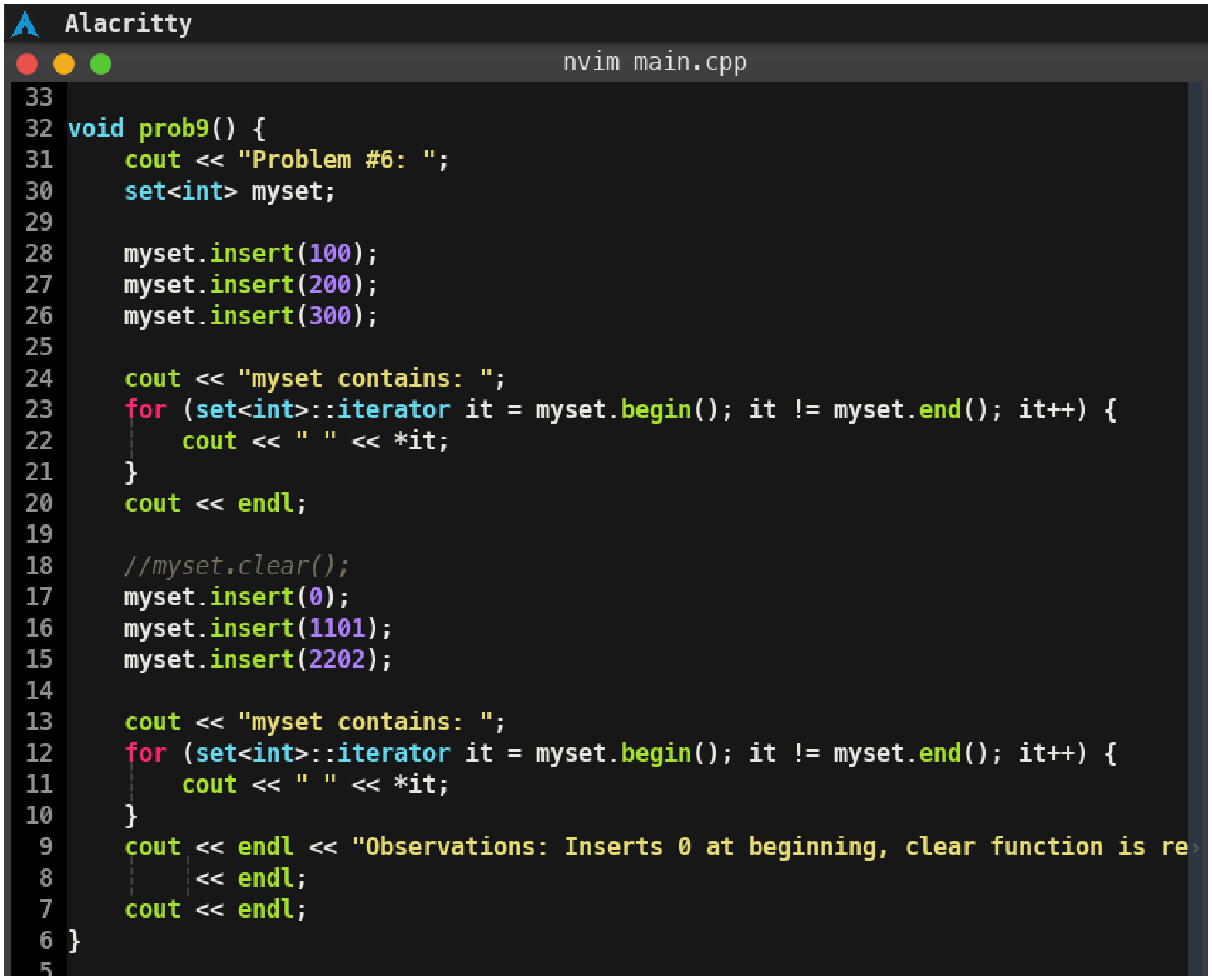
```
Problem #5: myset contains:  100 200 300
myset contains:  1101 2202
Observations: First set of {100, 200, 300} is cleared.
Since myset contains those values first, clearing it would  clear those values.
```



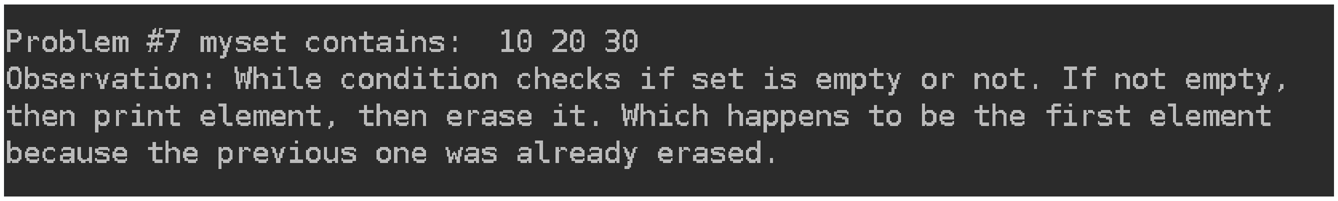
```cpp
void prob8() {
    cout << "Problem #5: ";
    set<int> myset;

    myset.insert(100);
    myset.insert(200);
    myset.insert(300);

    cout << "myset contains: ";
    for (set<int>::iterator it = myset.begin(); it != myset.end(); it++) {
        cout << " " << *it;
    }
    cout << endl;

    myset.clear();
    myset.insert(1101);
    myset.insert(2202);

    cout << "myset contains: ";
    for (set<int>::iterator it = myset.begin(); it != myset.end(); it++) {
        cout << " " << *it;
    }
    cout << endl << "Observations: First set of {100, 200, 300} is cleared." <<
         << "Since myset contains those values first, clearing it would "
         << " clear those values.";
    cout << endl << endl;
}
```



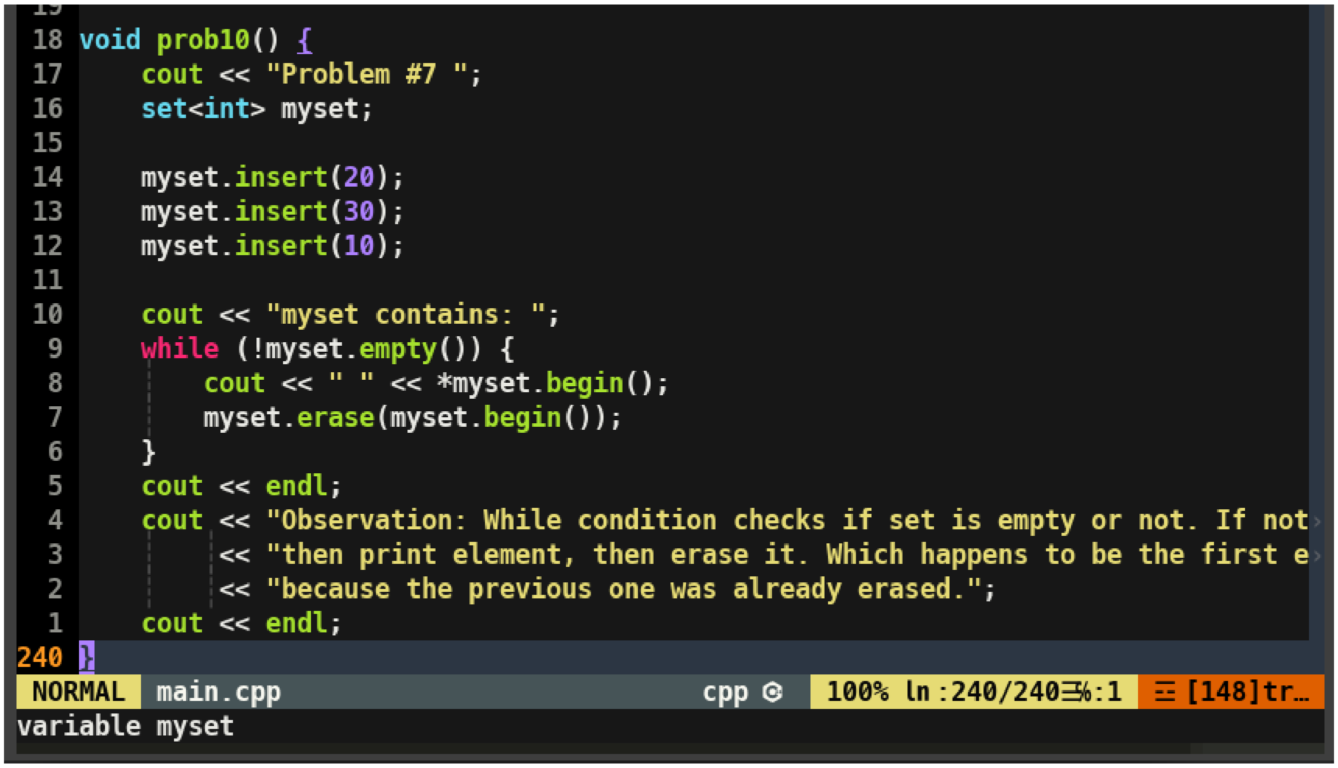
```
Problem #6: myset contains:  100 200 300
myset contains:  0 100 200 300 1101 2202
Observations: Inserts 0 at beginning, clear function is removed.
```



```cpp

void prob9() {
    cout << "Problem #6: ";
    set<int> myset;

    myset.insert(100);
    myset.insert(200);
    myset.insert(300);

    cout << "myset contains: ";
    for (set<int>::iterator it = myset.begin(); it != myset.end(); it++) {
        cout << " " << *it;
    }
    cout << endl;

    //myset.clear();
    myset.insert(0);
    myset.insert(1101);
    myset.insert(2202);

    cout << "myset contains: ";
    for (set<int>::iterator it = myset.begin(); it != myset.end(); it++) {
        cout << " " << *it;
    }
    cout << endl << "Observations: Inserts 0 at beginning, clear function is re
         << endl;
    cout << endl;
}
```



```
Problem #7 myset contains:  10 20 30
Observation: While condition checks if set is empty or not. If not empty,
then print element, then erase it. Which happens to be the first element
because the previous one was already erased.
```



```
void prob10() {
    cout << "Problem #7 ";
    set<int> myset;

    myset.insert(20);
    myset.insert(30);
    myset.insert(10);

    cout << "myset contains: ";
    while (!myset.empty()) {
        cout << " " << *myset.begin();
        myset.erase(myset.begin());
    }
    cout << endl;
    cout << "Observation: While condition checks if set is empty or not. If not
        << "then print element, then erase it. Which happens to be the first e
        << "because the previous one was already erased.";
    cout << endl;
}
NORMAL  main.cpp          cpp   100%  ln :240/240☰%:1   ☲ [148]tr…
variable myset
```