Here are my solutions for the simple c++ problems assigned to us. I hope to come back to them once I have a better grasp on the c++ language.

IntCell.h

```
#ifndef IntCell_H
#define IntCell_H

/**
 * A class for simulating an integer memory cell.
 */
class IntCell
{
  public:
    explicit IntCell( int initialValue = 0 );
    int read( ) const;
    void write( int x );
    int add(IntCell* x);
    int sub(IntCell* x);
    int addAndReturnIntCell(int y);

  private:
    int storedValue;
};

#endif
```

Intcell.cpp

```
#include "IntCell.h"

/* So, this is my generalized summary of what I was able to accomplish for this
exercise.
As you'll notice my code will not be runable in this state and this would be due to
the fact
that while doing hw1_question1.5 I accidentally dumped a bunch of hex into my
original Intcell.cpp
file and overwrote it without realizing. Here's what I was able to reproduce. */

/**
 * Construct the IntCell with initialValue
 */
IntCell::IntCell( int initialValue ) : storedValue( initialValue )
{
}

/**
 * Return the stored value.
 */
int IntCell::read( ) const
{
   return storedValue;
}

/**
 * Store x.
 */
void IntCell::write( int x )
{
   storedValue = x;
}

int Intcell::add (/*parameter was a pointer to an Intcell*/) {
        /*Here I had the add function read the stored value from the Intcell
        calling the function and add it to the read value of the incoming
        Intcell*/
}

int Intcell::sub (/*paramater was a pointer to an Intcell*/) {
        /*This was the same code as th addition function but had the minus operator
instead of
        addition*/
}
```

```
int Intcell::addAndReturnIntcell(/*paramater here was an int, say y*/) {
        /*Since I passed it a value for the int y, I was able to add it to an existing
Intcell
        and although I never got it to be able to return an Intcell, I was able to output
        the result of the constant y and the Intcell*/
}
```

TestIntCell.cpp

```
#include <iostream>
#include "IntCell.h"
using namespace std;

int main( )
{
   IntCell m;   // Or, IntCell m( 0 ); but not IntCell m( );
   IntCell n;

   int y = 12;

   m.write( 5 );
   n.write( 7 ) ;
   cout << "Cell M contains: " << m.read( ) << endl;
   cout << "Cell N contains: " << n.read( ) << endl;
   cout << "N + M added: " << n.add(&m) << endl;
   cout << "N - M subtracted: " << n.sub(&m) << endl;
   cout << "N + int y, which is 12: " << n.addAndReturnIntCell(y) << endl;
   return 0;
}
```