1. Short Answer Questions 20 points.

(a) What is printed by the following code snippet?

```cpp
vector <int> c(5,5);
vector <int> d;
d.push_back(77);
d.push_back(44);
c.push_back(100);
cout << c.front() << " " << d.front() << " " << c[1]
    << " " << d.back() << " " << d.size() << endl;
```

(b) What is printed by the following code snippet?

```cpp
int Array[4] = {7, 5, 8, 9};
int *pArray = Array;
++pArray;
(*pArray)++;
pArray++;
--(*pArray);
```
2. Inheritance and Virtual Functions 30 points.

What is printed by the attached inheritance program. Hint, there are 8 “Hello from” outputs.
class Base
{
    // Define a base class
    public:
        void Sub1();
    virtual void Sub2() = 0;
    virtual void Sub3();
    virtual void Sub4();
};
class A : public Base
{
    // Class A derives from Base
    public:
        void Sub1();
        void Sub2();
};
class B : public Base
{
    // Class B derives from Base
    public:
        void Sub1();
        void Sub2();
        void Sub4();
};
// Base Class Methods
void Base::Sub1() { cout << "Hello from Base::Sub1()" << endl; }
void Base::Sub3()
{
    cout << "Hello from Base::Sub3()" << endl;
    Sub1(); // DON'T MISS THIS CALL IN YOUR ANSWER
    Sub4(); // DON'T MISS THIS CALL IN YOUR ANSWER
}
void Base::Sub4() { cout << "Hello from Base::Sub4()" << endl; }
// Class A Methods
void A::Sub1() { cout << "Hello from A:Sub1()" << endl; }
void A::Sub2() { cout << "Hello from A:Sub2()" << endl; }
// Class B Methods
void B::Sub1() { cout << "Hello from B:Sub1()" << endl; }
void B::Sub2() { cout << "Hello from B:Sub2()" << endl; }
void B::Sub4() { cout << "Hello from B:Sub4()" << endl; }
// A Helper Subroutine
void Sub(Base& x)
{
    x.Sub1();
x.Sub2();
x.Sub3();
}
int main()
{
    A a;
    B b;
    Sub(a);
    Sub(b);
}
3. Smart Pointers 25 points

A code snippet is given below using the Smart Pointers concept. In the table below, fill in the value the specified variable at the specified line number. For at least one entry, you will have to make some assumptions and an educated guess. A few of the rows of the table are filled in for you as a starting point.

```cpp
// SPointer class and implementation defined here.

void Sub(SPointer sp)
{
    SPointer sp0(sp);
}

int main()
{
    SPointer sp1("This is a test");
    SPointer sp2("Another test");
    SPointer sp3 = sp1;
    SPointer sp4 = sp1;
    Sub(sp4);
    SPointer sp5("This is a test");
}
```

Program smart-pointers-question.cc

<table>
<thead>
<tr>
<th>After Line Number</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>sp1.st</td>
<td>0xB0001000</td>
</tr>
<tr>
<td>11</td>
<td>*(sp1.refCount)</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>sp2.st</td>
<td>0xB0001020</td>
</tr>
<tr>
<td>12</td>
<td>*(sp2.refCount)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>sp3.st</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>*(sp3.refCount)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>sp4.st</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>*(sp4.refCount)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>sp.st</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>*(sp.refCount)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>sp0.st</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>*(sp0.refCount)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>sp4.st</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>*(sp4.refCount)</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>sp5.st</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>*(sp5.refCount)</td>
<td></td>
</tr>
</tbody>
</table>
4. Matrix Calculator Assignment (25 Points)

(a) Suppose that in your Matrix class implementation, you did not define or implement a copy constructor, assignment operator or destructor (in other words you just used the default implementation for each of these. What would the matrix-calculator program do when processing the matrix operations found in the sample input.txt given in the assignment? Explain your answer.

(b) Suppose you implemented the destructor as shown below, but again did not implement either the copy constructor or assignment operator?

```cpp
Matrix::~Matrix() // destructor
{
    delete[] elements; // Free the elements array
}
```

(c) A correct implementation of the copy constructor and the assignment operator are nearly identical. State two differences in the implementation of these two operations.