CLOSED BOOK, CLOSED NOTES, CALCULATORS ALLOWED.

1. Subroutine Parameter Passing **25 points**
   
   What is printed by the program parameters.cc on the next page?
// Subroutine Parameter Passing Example
#include <iostream>
using namespace std;
int Sub2(int k, int j)
{
    cout << "Sub2 k is " << k << endl;
    int r = k * j;
    // Change value of both k and j arguments
    k = j;
    j = 10;
    return r;
}
void Sub1(int& x, int& y)
{
    cout << "Sub1 x is " << x << " y is " << j << endl;
    // Change value of x
    x = 5;
    // Pass x and y to Sub2
    Sub2(y, x);
}
int main()
{
    int i = 2;
    int j = 3;
    int k = 4;
    Sub1(i, j);
    int m = 2 * Sub2(j, k) + 10;
    cout << "Main i is " << i << " j is " << j << " k is " << k << " m is " << m << endl;
}
2. Complex Calculator Assignment 25 points

(a) How many operands would be counted (the value of the “count” variable in the skeleton code) if the input was the following. Explain your answer.

\[+ (1, 2)\]

(b) What is the value of \(\text{delims}[1]\) if the input was the following?

\[(1, 1) + (2, 2)\]

(Note. \(\text{delims}\) is the array of length 2 that is passed as the fourth argument to \text{StringParse} in the skeleton code).

(c) What should be the return type for Complex class \text{Mag} and \text{Angle} member functions? Explain your answer.

(d) How many arguments are needed to define and implement a non-member function addition operator? Explain your answer.

(e) How could you implement a non-member function for the \text{Print()} function? Explain your answer.
3. What is printed by the program subroutines on the next pages? (25 Points).

_Hint_: There are ten outputs.
// Subroutine parameter passing example.
// ECE2036, Fall 2013

#include <iostream>
using namespace std;

int Sub2( int arg0, int arg1)
{
    // Subroutine with two args; does some manipulation of the arguments
    // and then multiplies the results and returns the product.
    return ++arg0 * arg1++;
}

int Sub1( int* pInt0, int* pInt1)
{
    return (*pInt0++) * (*pInt1++);
}

int main()
{
    // Two arrays for manipulating
    int A[10] = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 };  
    int B[10];

    int* pZero = A;
    int* pOne = &A[1];
    cout << "*pZero is " << *pZero << endl;
    cout << "*pOne is " << *pOne << endl;

    int sub1Return = Sub1(pZero++, pOne++);
    cout << "Sub1 Return is " << sub1Return << endl;
    cout << "*pZero is " << *pZero << endl;
    cout << "*pOne is " << *pOne << endl;

    int sub2Return = Sub2(A[1], A[2]);
    cout << "Sub2 Return is " << sub2Return << endl;

    int* pA = A;
    int* pB = B;
    *pB++ = *pA++;
    *pB++ = pA[2];
    cout << "B[0] is " << B[0] << endl;

}  

Program subroutines.cc
4. Constructors and destructors  *(25 Points)*

In the program `constructors-destructors.cc` attached, identify where (what line number) each of the default constructors, int constructors, copy constructors, and destructors are called for each class `A` and `B`. Specify which line of code causes each of the above and a brief explanation of why the constructor was called.

As an example of how to fill in the table, one entry for the `A` int constructor is filled in.

<table>
<thead>
<tr>
<th>A Default Constructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A int Constructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td></td>
<td>Declaration of local variable “a” with int argument</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A Copy Constructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A Destructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B Default Constructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B int Constructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B Copy Constructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B Destructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
class A {
    public:
        A();       // Default constructor
        A(int);    // int Constructor
        A(const A&); // Copy constructor
        A();       // Destructor
        A operator+(A rhs) const; // Addition operator
    public:
        int x;     // Single data member
};

class B {
    public:
        B();       // Default Constructor
        B(int);    // int Constructor
        B(const B&); // Copy constructor
        B();       // Destructor
        B operator+(const B& rhs) const; // Addition operator
    public:
        int x;     // Single data member
};

A A::operator+(A rhs) const
{
    A r(x + rhs.x);
    return r;
}

class B {
    public:
        B();       // Default Constructor
        B(int);    // int Constructor
        B(const B&); // Copy constructor
        B();       // Destructor
        B operator+(const B& rhs) const; // Addition operator
    public:
        int x;     // Single data member
};

B B::operator+(const B& rhs) const
{
    return B(x + rhs.x);
}

// Implementation of all constructors/destructors ommitted for brevity

int main()
{
    A a(1);
    B b(2);
    A a2;
    B b2;
    a2 = a + a;
    b2 = b + b;
}

Program constructors destructors.cc