Name: _key_
PLEASE PRINT CLEARLY

ECE x70 Exam No. 2 (100pts. - 25% of the final grade)

General Remarks

This is in-class one-hour-long exam. You can use your notes, textbook, and any existing Web-based resources. You can use a lab or your own computer but must not use a cell phone. You must not communicate with other people or post the questions on an Internet forum. Provide a concise answer and to the point for maximum credit. Answers that are too long take too much time and may indicate that the author is unable to rank the importance of facts.

DL: ___ ERR: ___ PTS: ___

DL – exam difficulty level (adjustment), ERR – exam errors, PTS – exam points.

Problem 1A (20pts)

Fill in the blanks, implement a function that accepts a 1D `vector` of `double` *V*. The function computes a product of all vector elements utilizing a lambda expression and for_each function template from the <algorithm> library. Make an optimal choice among passing by value, by reference, and const reference. Some blanks may remain unused.

```
double FN ( _const_2p vector<double> ___&___ 2p V ) {

    double product = 1.0;

    auto myLambda = [ _&product___5p ]( _double x___5p ){ _product *= x______5p ; } ;

    for_each( _V.begin()_______ 2p , _V.end()__________ 2p , _myLambda_________ 2p );

    return(product);

} // const double &x is also OK (double is small though), double &x is -2pts.

  // note missed & in &product creates completely different instruction – no partial
```

Problem 1B (5pts)

Answer a few short yes/no questions based on the information that was provided about lambda expressions in the lectures and on what you can look up quickly during this test.

```
All lambda expressions are void functions (cannot return value)      no / yes

All lambda expressions can accept only one parameter

   passed as its argument (either by value or by references)         no / yes
```

All lambda expressions are void functions (cannot return value) [✓ no] / yes

All lambda expressions can accept only one parameter passed as its argument (either by value or by references) [✓ no] / yes

Problem 2 (20pts)

Someone wrote a function that operates on a vector<double>. Later during the project development it was determined that the operation needs to be performed on list data container that does not have operator[] and instead utilizes iterators. Rewrite the function with use of iterators.

```
double FN_old(const vector<double> & D)
{
  double acc=0.0; // accumulated computation
  size_t index;
  index = 1;
  while ( index < D.size() ) {
    acc = acc + abs(D[index-1] - D[index]);
    ++index;
  }
  return(acc);
}

double FN_new( // consider your recent homework with a 1D point
    list<double>::const_iterator BE,
    list<double>::const_iterator EN ) {
  double acc=0.0; // accumulated computation
  // below is one of multiple solutions, so addressing (1)-(5) was graded!
  if ( BE != EN ) {                      // (1) handle empty data containers
    list<double>::const_iterator AH = BE;       // (2) declare iterator(s)
    ++AH;   // (3) arrange for prev and next OR behind and ahead iterators
    while ( AH!=EN ) { // (4a) process until the end of the data container
      acc = acc + abs(*AH - *BE );    // (5)  correct formula including *s
      ++AH; ++BE;      // (4b) process until the end of the data container
    }         // graded:each (1)-(5) is worth 4pts., extras -2pts. each
  }           // graded:each (1)-(5) either V(check), P(artial) or M(issed)
  return(acc);
}
```

Problem 2B (5pts)

Answer a few short yes/no questions based on the information that was provided about <algorithm> in the examples provided in the lectures and lecture notes.

```
for_each function template can use reverse iterators              no / yes
```
no / [✓ yes]

```
for_each function template can use begin from one vector

    and end from a different vector as arguments                  no / yes
```
[✓ no] / yes

Problem 3A (15pts.)

Analyze the definition of the following class that encloses another class. Implement each class default constructor, copy constructor, assignment operator, and destructor.

```
class CONTAINER {
protected:
    double   data1[10];
    double * data2; // always holds an array of 10,000 elements
public:
    …

    CONTAINER()  { _data2 = new double[ 10000 ]________________ ; }

    ~CONTAINER() { _delete [] data2;___________________________ ; }

    CONTAINER(const CONTAINER & c) {

        _data2 = new double[ 10000 ];_

        _for( size_t i=0; i<10; ++i) data1[i] = c.data1.[i];_______

        _for( size_t i=0; i<10000; ++i) data1[2] = c.data2.[i];____
    }
    void operator=(const CONTAINER & c) {

        _if (this != &c) {___________________________________________

        _____for( size_t i=0; i<10; ++i) data1[i] = c.data1.[i];_____

        _____for( size_t i=0; i<10000; ++i) data1[2] = c.data2.[i];_

        _}___________________________________________________________
    }
};
```

Problem 3B (10pts.)

Answer a few short yes/no questions based on the limited information that is provided about the class above.

```
The destructor is needed                                       no / [✓ yes]

The copy constructor needs to be either implemented or disabled     no / [✓ yes]

The assignment operator needs to be either implemented or disabled  no / [✓ yes]

This class is an abstract base class                           [✓ no] / yes

This class is a pure abstract base class                       [✓ no] / yes
```

Problem 4 (25pts.)

Answer the questions about the accessibly of features defined in the indicated lines and the general questions below the code fragment. The case study was developed to test your understanding of the theory behind the topic and may not necessarily be practical in real life.

```
class A {
private:
    double x, y;                            Is this accessible in B?     Y / [✓ N]
protected:
    A() : x(0.0), y(0.0) {}
    void setX(double nx) { … ; }            Is this accessible in B?     [✓ Y] / N
    void setY(double ny) { … ; }            Is this accessible in main?  Y / [✓ N]
    double getX() const  { … ; }            Is this accessible in B?     [✓ Y] / N
    double getY() const  { … ; }            Is this accessible in main?  Y / [✓ N]
};
class B : private A {
private:
    double z;                               Is this accessible in A?     Y / [✓ N]
public:
    B() : z(0.0) { setX(0.0); setY(0.0); }
    void setX(double nx) { … ; }            Is this really needed?       [✓ Y] / N
    void setY(double ny) { … ; }            Is this accessible in A?     Y / [✓ N]
    void setZ(double nz) { … ; }            Is this accessible in main?  [✓ Y] / N
    double getZ() const  { … ; }            Is this accessible in A?     Y / [✓ N]
};
Can an object of class A be declared in main?   Y / [✓ N]
Can an object of class B be declared in main?   [✓ Y] / N
Does this example use aggregation?              Y / [✓ N]
Does this example use inheritance?              [✓ Y] / N
```