1. (5 points) What are advantages and disadvantages of decimal data types?

Advantages: accuracy

Disadvantages: limited range, wastes memory

3. (10 points) What are the arguments for and against Java's implicit heap storage recovery, when compared with the explicit heap storage recovery required in C++?

Advantages over C++: It is automatic reclamation of storage with the consideration of lifetime of objects, prevents the creation of dangling pointers and the heap fragmentation by handling removed objects in such a way they become freed space for other, still existing objects.

5.(10 points) Make two lists of applications of matrices, one for those that require jagged matrices and one for those that require rectangular matrices. Now argue whether just jagged, just rectangular, or both should be included in a programming language.

Jagged:

-Efficiency improved models over Rectangular arrays (unused entries are not included).

-Classification of objects or storage of values is not symmetric, but instead unregular in distribution, for example a calendar.

-Any unprecise (in terms of dimensions length) configuration of arrays.

Rectangular:

-Matrix/vector operations (the row major or column major orders are suitable for this operations).

-Editing Distance operations

-Any precise (in terms of dimensions length) configuration of arrays.

In our opinion both ragged/jagged arrays and rectangular arrays should be part of any programming language. In the case of ragged arrays this kind of arrays allows us to reduce the space complexity of any arrangement because we are able to no utilize the arrangements we do not need, whereas in the rectangular kind of arrays we must utilize entries that might not be needed. The reduction of space complexity, the locality of the objects/values stored, after the first entry of every array in the array of arrays is located, permit a reduction an efficient use of memory space. In comparison, multidimensional arrays are fundamental for matrix operations and any other comparable arrangements. The fact that multidimensional arrays can be arranged in row major and column major allow us to easily access and retrieve values from the possibly huge arrangement in constant time. This is powerful when the computational goal is to collapse or produce a result from the arrangement represented by the multidimensional array. This is due to the fact computationally all positions and memory locations can be retrieved by using: f(i, j, k) = L + S\*(j\*m\*p + i\*p + k) or f(i, j, k) = L + S\*(i\*n\*p + j\*p + k).

7.(10 points) Summarize the fundamental argument for dynamic method binding. Why do C++ and C# use static method binding by default?

The argument for dynamic method binding is that when we have a collection of subclasses that emanate from the same class, usually those classes have similar methods, suppose shape superclass and circle, oval, square, and rectangle, each with a draw method. In order to decide which of the possible draw methods to use throughout the inheritance relationships. This means the main idea of dynamic method binding is that dynamically, or as the context changes, the methods referenced used by the object along its inheritance relationship(s) change.

The reason static method binding is a default in both languages is because most of the methods implemented in classes are not meant to be overridden, if the methods are meant to be overridden then both languages require you to indicate it. Also, a second concern might be that if a language like C++ allows multiple inheritance, then even if dynamic binding is activated it is not clear which of two equivalently named and inherited methods will be run from a specific subclass.

9.(5 points) Explain why allowing a class to implement Multiple interfaces in Java and C# does not create the same problems that multiple inheritance in C++ creates.

In C++ multiple inheritance can create the conditions of the diamond problem, in which a subclass implements two superclasses, but it might be that these two super classes contain a method that is equal in terms of the method signature. If the subclass were to make a call of this doubly-referenced method, then at the compilation stage of the program an error will show saying more or less: “member ‘[name of method]’ found in multiple base classes of different types” at which point the developer will have to indicate which of the two or more implementations does the developer is referring to.

In contrast, Java only allows for multiple inheritance of interfaces at which point even if the two interfaces were to require the inclusion of a method with the same signature the implementation will be made at the scope of the subclass and not in the inheritance scope. C#, on a similar idea, follows Java implementation where interfaces can be implemented multiple times, but again, the possible problem issue with diamond problem does not occur for the same reasons as in Java.

11. (15 points) Rewrite the following C++ classes in Java and compare the result with the C++ version in terms of readability and writability. Hint: Both stack\_2 and queue\_2 are \*private\* subclasses.

Java Code:

//////////////////////////////////////////////////////////////////////////////

class single\_linked\_list {

private class node {

public node link;

public int contents;

};

node head;

public single\_linked\_list() {head.contents=0};

public void insert\_at\_head(int);

public void insert\_at\_tail(int);

public int remove\_at\_head();

public boolean empty();

}

//////////////////////////////////////////////////////////////////////////////

private class stack\_2 {

private single\_linked\_list list;

public stack\_2()

{

list = new single\_linked\_list();

}

void push(int value) {

list.insert\_at\_head(value);

}

int pop() {

if(!list.empty())

return list.remove\_at\_head();

else

return null;

}

}

//////////////////////////////////////////////////////////////////////////////

private class queue\_2{

private single\_linked\_list list;

public queue\_2()

{

list = new single\_linked\_list();

}

void enqueue(int value)

{

list.insert\_at\_tail(value);

}

int dequeue()

{

if(!list.empty())

return list.remove\_at\_head();

else

return null;

}

}

//////////////////////////////////////////////////////////////////////////////

After the conversion process, it can clearly be seen that the code needed to express an equivalent class and subclass arrangement in Java is much longer. This means that writability when compared between C++ and Java is in favor of C++. Readability, for the consequences of having a concise set of statements, is also in favor of C++. In general, if a programming language requires a smaller set of statements to express a desired arrangement of classes and subclasses it is not only easy to write but also easy to understand, hence, readability and writability are higher in C++ for this specific purpose.