Please answer the following five questions, each of which is worth 30 points.

1. This question deals with object-oriented design principles.

(a) Define the term *casting*.
(b) Define the term *polymorphism*.
(c) Consider the following Java classes:

```java
public class Organization extends Object {
    Organization() { /* null constructor */ }
    public void printMe() { System.out.println("Organize."); }
}

public class Company extends Organization {
    Company() { /* null constructor */ }
    public void printMe() { System.out.println("Be productive."); }
}

public class MidCap extends Company {
    MidCap() { /* null constructor */ }
    public void printMe() { System.out.println("Think big."); }
}

public class InternetCo extends MidCap {
    InternetCo() { /* null constructor */ }
    public void printMe() { System.out.println("Be cool."); }
    public static void main(String[] args) {
        Company mid = new MidCap();
        MidCap netscape = new InternetCo();
        Object obj = new Organization();
        Organization startup = new Company();
        mid.printMe();
        netscape.printMe();
        ((Organization) obj).printMe();
        obj = netscape;
        ((MidCap) obj).printMe();
        obj = startup;
        ((Organization) obj).printMe();
    }
}
```

What is the output from calling the main() method of the InternetCo class?
2. This question deals with algorithm analysis.

(a) Show, using the definition of the big-Oh, that the function \( f(n) = 20n^3 + 15n + 12 \) is \( O(n^3) \).

(b) Characterize using the big-Oh notation the worst-case running time of the following algorithm:
   
   Let \( A \) be a given array of \( n \) integers.
   
   for \( i \leftarrow 0 \) to \( n - 1 \) do
     
     for \( j \leftarrow n - 1 \) to \( i \) do
       
   
   end for
   
   end for

3. This question deals with the Sequence ADT.

(a) Draw a picture representing the ordered contents of an initially-empty sequence \( S \) after each of the following operations:
   
   i. \( S\text{-insertLast}(a) \)
   
   ii. \( S\text{-insertLast}(b) \)
   
   iii. \( S\text{-insertAtRank}(1,c) \)
   
   iv. \( S\text{-remove}(S\text{-before}(S\text{-last}())) \)
   
   v. \( S\text{-insertAfter}(S\text{-last}()),d) \)

(b) Assume you are implementing the sequence with a doubly-linked list with header and trailer nodes. Fill in the missing line in the following Java implementation of the \textit{insertAfter} method. Assume that doubly-linked nodes themselves are implementing the \textit{Position} interface; hence, they contain instance variables \textit{element} and \textit{container} and implement methods with these same names.

   ```java
   public Position insertAfter(Position p, Object e) {
     DoubleNode temp = new DoubleNode();
     temp.element = e;
     temp.container = this;
     temp.prev = p;
     temp.next = p.next;
     // Fill in this step.
     p.next = temp;
     return temp;
   }
   ```

4. Consider a sequence \( S \) implemented with a doubly-linked list. Suppose further that \( S = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) \), in that order, so that each element is actually equal to its rank.

(a) What is the rank of an element that causes the \textit{elemAtRank} method to run the slowest? Why?

(b) What is the rank of an element that causes the \textit{insertAtRank} method to run the fastest? Why?

5. Write a Java class that adapts a Positional Sequence to a Stack. That is, it uses a Positional Sequence to implement the Stack interface.