NAME (last, first)_____ Cornell NetId _____

This 90-minute exam has 6 questions worth a total of 100 points. Look at the whole test before beginning to see what is expected. Budget your time wisely. Use the back of these pages if you need more space. You can tear the pages apart; we have a stapler at the front of the room.

Question 0 (2 points): Write your name and NetID, legibly, at the top of each page.

Question 1 (19 points): **(a)** Define *two* of the following: argument, precondition, return type.

(b) Suppose b and c contain (names of) objects of the a class CL. Suppose that (1) CL does not explicitly extend any other class and (2) CL does not override equals. What do b == c and b.equals(c) do?

(c) Write down the four steps in executing a procedure call.

(d) To the right is a class definition with a procedure declared in it. Below is a procedure call. Do *just the first step* (see part (c) above) in executing this procedure call, i.e. draw the frame for the call.

P1.met(3+2);

```
public class P1 {
   static int x;
   public static void met(int y) {
      if (x > 0) {
         int z;
         z= y+1;
         x= z;
      }
   }
```

Name:_____ NetID:_____

Question 2 (20 points): Below are two class definitions. To the right of each, draw one folder (object, instance) of the class. You need not draw the Object partition. It does not matter what values you put in the fields.

```
public class Candidate {

 private String name;
 private static int nextID= 0;

 /** = a new, unique id number*/
 public static int getNextID() {
      nextID= nextID + 1;
      return nextID;
 }

 /** = this Candidate's name */
 public String getName() {
      return name;
 }

 /** set Candidate's name to n */
 public void setName(String n) {
      name= n;
 }

 /** = String repr. of Candidate */
 public String toString() {
      return getName();
 }
}
```

```
public class Federal extends Candidate {

  private double contributions= 0.0;

  /** Constructor: object with name n and
      contributions s */
  public Federal(String n, double s) {
        setName(n);
        contributions= s;
    }

  /** = this candidate's contributions */
  public double getContributions() {
      return contributions;
  }

  /* Set this candidate's contributions
     to s */
  public void setContributions(double s) {
      contributions = s;
  }

  /** = String repr. of this candidate */
  public String toString() {
     return "Federal: " + super.toString();
  }
}
```

Name:_____ NetID:_____

Question 3 (15 points) Assume that the following two variables have been initialized to contain the names of folders (see question 2 for the two class definitions):

```
Candidate x;          Federal y;
```

(a) State the general steps involved in evaluating a new-expression, for example `new Federal("Obama", 100000);` .

(b) Name the methods that `Federal` overrides.

(c) Suppose these statements have been executed:

```
Candidate x=  new Candidate();    x.setName("McCain");
Federal y=  new Federal("Obama", 100000.0);
```

For each of the following method calls, if the call does not produce an error, write the value returned. If an error does occur, explain why.

```
x.toString()
```

```
y.toString()
```

```
y.getName()
```

```
y.getContributions()
```

```
x.getContributions()
```

Name:_____ NetID:_____

Question 4 (24 points): This question refers to classes `Candidate` and `Federal` defined in Question 2.
Write a subclass `Presidential` of `Federal`.

1. The methods you write must have suitable specifications, as javadoc comments.

2. Class `Presidential` should contain a field for the number of states this presidential candidate won.

3. The constructor should allow one to specify the candidate's name and number of states won. The contributions should be 0 —do not have a parameter for the contributions. The body of the constructor *must* begin with an explicit call on a superclass constructor.

4. The class should have a method `atLeast25()` that returns true if this candidate won at least 25 states and returns false otherwise.

5. The class should have a method `toString()` that returns whatever toString in the superclass would return. In addition, if the candidate won at least 25 states, append the string " Good chance". For example, toString might return: "Federal: Abraham Lincoln Good Chance".

Name:_____ NetID:_____

Question 5 (20 points): In class `Candidate from Question 2`, we want a function to convert a name into an ID. The ID has the following properties:

1. The ID is made up of two or three letters and an integer. Each ID should have a different number.
2. For a person with a first, middle, and last name, the ID has three letters: the first letter of the first, middle, and last name. For a person without a middle name, the ID has two letters: the first letter of the first name and last name.
3. All letters in the ID must be lowercase.

Examples: "Barack Obama" may become bo6
"John Sidney McCain" may become jsm5

Question 0	/02
Question 1	/19
Question 2	/20
Question 3	/15
Question 4	/24
Question 5	/20
Total	/100

Write function makeId, whose header is given below. In addition to the methods in class `Candidate`, you can use the following methods, assuming `s` is a `String`. **All the methods you need are given in the following table or in class `Candidate`.**

int	`s.length()`	= the number of characters in `s`
char	`s.charAt(i)`	= the character at position `i` in `s`
int	`s.indexOf(n)`	= the index within `s` of the first occurrence of `String n`
Int	`s.lastIndexOf(n)`	= the index within `s` of the last occurrence of `String n`
`String`	`s.toLowerCase()`	= `s` with uppercase letters replaced with lowercase letters
`String`	`s.substring(h,k)`	= a `String` consisting of characters in `s[h..k-1]`, i.e. `s[h], s[h+1], ..., s[k-1]`
`String`	`s.substring(h)`	= a `String` consisting of characters `s[h..s.length()-1]`

```
/** = An ID corresponding to s. The ID is the lowercase version of the
    letters that begin each part of the name, followed by a unique integer.
    Precondition: s has the form "first last" or "first middle last",
         with exactly one space separating the parts of the name and no
         spaces at the beginning or end. */
public static String makeID(String s) {




}
```