QUESTION 1.

```
/** b[0..N-1] is sorted in ascending order.
    Return an integer k that truthifies this assertion:

        b[0..k] <= x < b[k+1..N-1] */

public static int search(int[] b, int x) {
    int k= -1; int h= N;
    // invariant: b[0..k] <= x < b[h..N-1]
    while (h != k+1) {
        int e= (k+h) / 2;
        // { -1 ≤ k < e < h <= N }
        if (b[e] <= x) k= e;
        else h= e;
    }
    return k;
}
```

QUESTION 2(a).

Prove the base and inductive cases:
Base case: Prove P(0).
Inductive case: Assume P(0), …, P(k),
for k >= 0, and prove P(k+1).

QUESTION 2(b).

A class C is made abstract so that it cannot be instantiated.
A method is made abstract so that it must be overridden in any subclass of C.

QUESTION 2(c).

```
try {
    dn= Integer.parseInt(somestring);
}
catch (NumberFormatException e) {
    dn= 1;
}
```

QUESTION 3

```
/** = longest common subsequence of s1 and s2 */
public static String lcs(String s1, String s2) {
    if (s1.length() == 0  ||  s2.length() == 0)
        return "";

    // { s1 and s2 have at least one char each }
    if (s1.charAt(0) == s2.charAt(0)) {
        return s1.charAt(0) +
                lcs(s1.substring(1), s2.substring(1));
    }

    // { first chars are different }
    String first=  lcs(s1, s2.substring(1));
    String second=  lcs(s1.substring(1), s2);
    if (first.length() >= second.length())
        return first;
    else return second;
}
```

QUESTION 4(a).

```
public class Celebrity implements Personal {
    private String name;
    private int age;

    // Constructor: instance with name n and age a
    public Celebrity(String n, int a)
        {  name = n; age = a; }

    // = the name of this Celebrity
    public String getName()  {  return name; }

    // = the age of this Celebrity
    public int getAge() {  return age; }
}
```

QUESTION 4(b).

```
public class Actor extends Celebrity
                        implements Business {
    private String address;
    private int earnings;

    // Constructor: instance with name n, age a,
    // address addr, and earnings e
    public Actor(String n, int a, String addr, int e)
        { super(n,a); address = addr; earnings = e; }

    // = the address of this Actor
    public String getAddress() {  return address; }

    // = the earnings of this Actor
    public int getEarnings() {  return earnings;  }
}
```

QUESTION 5. The inner class. Place it right after method iterator, and add a blank remove() function to make it compilable. There is no need for an explicit constructor; we use the default.

```
/** An iterator of the Processes in this instance */
public class ProcessIterator implements Iterator {
   int posA = 0; // index of next position in userA
   int posB = 0; // index of next position in userB

   /** = there exists another item to enumerate */
   public boolean hasNext()
       {  return (posA < sizeA || posB < sizeB);  }
   /** = next element in the enumeration. Throw
         NoSuchElementException if no more items */
   public Object next() {
       if (!hasNext())
           throw new NoSuchElementException();
       if (posB == sizeB ||
          (posA < sizeA && userA[posA].priority >=
                           userB[posB].priority)) {
         posA= posA+1;
         return userA[posA-1].priority;
       }
       posB= posB+1;
       return userB[posB-1].priority;
   }
}
```