```java
/*
 * Author: Ruiqing Qiu
 * PID: A98022702
 * TA: Vineel
 * File name: hw2.java
 */
import java.io.*;
import java.util.*;

public class hw2
{
  //Total number of zip codes
  final static int TOTAL = 32845;

  static int PopTotal = 0;
  //One array to store the zip codes
  static String [] zip = new String [TOTAL];
  //One array to store the population
  static int [] population = new int [TOTAL];

  //Initialize a int array that are all -1
  static int [] GuessedZip = {-1,-1,-1,-1,-1};
  //Keep track of all numbers that need to be guessed, -1 indicate guessed
  static int [] numbers = {0,1,2,3,4,5,6,7,8,9};

  public static void main(String[]args){
    int lines = 0;
    try{
      FileInputStream fstream = new FileInputStream("zipcode.txt");
      BufferedReader br =
          new BufferedReader(new InputStreamReader(fstream));
      String strLine;

      int zipIndex = 0;
      int popIndex = 0;
      while ((strLine = br.readLine()) != null){
        //Skip the first line
        if(lines == 0){
          lines++;
          continue;
        }
        //Incrment the lines
        lines++;

        String[] splited = strLine.split("\\s");
        zip[zipIndex++] = splited[0];
        population[popIndex++] = Integer.parseInt(splited[1]);
        //Get the total population
        PopTotal += Integer.parseInt(splited[1]);
      }
      br.close();
    }
    catch (Exception e){
      e.printStackTrace();
    }
    System.out.println("Total Population: " + PopTotal);

    //This part is for problem 2.5 (a)
    PriorityQueue<Pair> pq = new PriorityQueue(TOTAL,
            new Comparator<Pair>(){
              public int compare(Pair a1, Pair a2){
                return a2.pop.compareTo(a1.pop);
              }});

    for(int i = 0;i < zip.length; i++){
      pq.add(new Pair(zip[i],population[i]));
    }
    System.out.println("The five most populated zip codes are: ");
    for(int i = 0; i < 5; i++){
      System.out.println("zip code: " + pq.peek().zip
                + " with population "+ pq.poll().pop);
    }
```

```
    //Program Start, the very first guess
    double [] guess = new double[10];
    int guessed = 0;
    System.out.println("Which mode do you want? \n" +
            "Y for enter you own or X for suggested guess");
    try{
      BufferedReader bufferRead =
           new BufferedReader(new InputStreamReader(System.in));
      String line = bufferRead.readLine();
      //If enter your own
      if(line.compareTo("Y") == 0)
      {
        for(int i = 0;i < 10; i++)
        {
          char c = (char) ('0'+i);
          guess[i] = getProb(c);
        }
        int max = 0;
        for(int i = 0; i < guess.length; i++){
          if(guess[i] > guess[max])
            max = i;
            System.out.println(i+ ": " + guess[i]);
        }
        System.out.printf("\n\n");
        int nextGuess = max;
        System.out.println("The next guess is : " + max + "\n" +
                  "with probability: " + guess[max]);

        while(true)
        {
          System.out.print("Zip code: ");
          for(int i = 0; i < GuessedZip.length; i++){
            System.out.print(GuessedZip[i] + " ");
          }
          System.out.println("\nEnter you guessed digit and its " +
                  "position:\n"
           + "(For example, 0 1 to indicate it's 0 at the first position)"
           + "\n(Or, 2 -1 to indicate a wrong guess)");
          line = bufferRead.readLine();
          String[] splited = line.split("\\s");
          //Getting the guessed number
          guessed = Integer.parseInt(splited[0]);
          //Wrong guess
          if(splited[1].compareTo("-1") == 0){
            numbers[guessed] = -1;
          }
          else{
            for(int i = 1; i < splited.length; i++){
              GuessedZip[Integer.parseInt(splited[i])-1] = guessed;
            }
            for(int i = 0; i < GuessedZip.length; i++){
              System.out.print(GuessedZip[i] + " " );
            }
            for(int i = 0; i < numbers.length; i++)
            {
              if(numbers[i] == guessed)
                numbers[i] = -1;
            }
          }
          System.out.println();
          boolean found = checkIfFound();
          if(found){
            System.out.println("The zip code is: ");
            for(int i = 0; i < GuessedZip.length; i++){
              System.out.print(GuessedZip[i]);
            }
            System.out.println();
            return;
          }
          //Calculate next guess given the previous evidence
          for (int i = 0; i < numbers.length; i++){
            if(numbers[i] == -1){
              guess[i] = 0;
```

```
            continue;
          }
          char c = (char)('0' + numbers[i]);
          guess[i] = evidenceProb(c);
        }
        for(int i = 0; i < guess.length; i++){
          if(guess[i] > guess[max])
            max = i;
          System.out.println(i+ ": " + guess[i]);
        }
        System.out.printf("\n\n");
        nextGuess = max;
        System.out.println("The next guess is : " + max + "\n" +
                "with probability: " + guess[max]);
      }//End of while
    }//End of if
  }//End of try
  catch(Exception e){
    e.printStackTrace();
  }

  for(int i = 0;i < 10; i++){
    char c = (char) ('0'+i);
    System.out.println(getProb(c));
    guess[i] = getProb(c);
  }
  int max = 0;
  for(int i = 0; i < guess.length; i++){
    if(guess[i] > guess[max])
      max = i;
  }
  guessed = max;

  for(int i = 0; i < numbers.length;i++){
    if(numbers[i] == guessed)
      numbers[i] = -1;
  }

  while(true){
    System.out.print("Where is digit " + guessed +
          "? (Enter position range from 1 - 5, separated by space)" +
           " or -1 to indicate none): ");
    try{
      BufferedReader bufferRead =
        new BufferedReader(new InputStreamReader(System.in));
      String line = bufferRead.readLine();
      String[] splited = line.split("\\s");
      //If guess incorrectly continue
      if(splited[0].compareTo("-1") == 0)
        continue;
      else{
        for(int i = 0; i < splited.length; i++){
          GuessedZip[Integer.parseInt(splited[i])-1] = guessed;
        }
      }
      for(int i = 0; i < GuessedZip.length; i++){
        System.out.print(GuessedZip[i]);
      }
      System.out.println();
    }
    catch (Exception e){
      e.printStackTrace();
    }
    finally{
      boolean found = checkIfFound();
      if(found){
        System.out.println("the zip code is: ");
        for(int i = 0; i < GuessedZip.length; i++){
          System.out.print(GuessedZip[i]);
        }
        System.out.println();
        return;
      }
```

```
        //Calculate next guess given the previous evidence
        for(int index = 0; index < numbers.length; index++){
          //If the number is already guessed
          if(numbers[index] == -1){
            guess[index] = 0;
            continue;
          }
          char c = (char)('0' + numbers[index]);
          //System.out.println(c);
          //System.out.println(evidenceProb(c));
          guess[index] = evidenceProb(c);
        }
        for(int i = 0; i < guess.length; i++){
          System.out.println(i + ": " + guess[i]);
          if(guess[i] > guess[max])
            max = i;
        }
        guessed = max;

        for(int i = 0; i < numbers.length;i++){
          if(numbers[i] == guessed)
            numbers[i] = -1;
        }
        continue;
      }
    }
  }
  //A method to check if the zip code is already found
  public static boolean checkIfFound()
  {
    for(int i = 0; i < GuessedZip.length; i++){
      if(GuessedZip[i] == -1)
        return false;
    }
    return true;
  }

  //A method to calculate the predictive probability
  public static double evidenceProb(char a)
  {
    double prob = 0.0;
    double total = 0.0;
    boolean good = true;
    double denominator = denominator();
    for(int i = 0; i < zip.length; i++){
      good = true;
      //This for loop checks things already guessed
      for(int j = 0; j < 5; j++){
        //The place is guessed and correct
        if(GuessedZip[j] != -1){
          //If condition not satisfied, go to next zip code
          if(zip[i].charAt(j) != ('0' + GuessedZip[j])){
            good = false;
            break;
          }
        }
        //Things that have not guessed
        if(GuessedZip[j] == -1){
          for (int k = 0; k < 10; k++){
            //The number that has been guessed
            if(numbers[k] == -1){
              if(zip[i].charAt(j) == ('0'+k)){
                good = false;
                break;
              }
            }
          }
        }
      }//End of for loop
      //If it pass the test
      if(good){
        //Reach here if it's still good
        for(int j = 0; j < 5; j++){
```

```
          if(GuessedZip[j] == -1){
            if(zip[i].charAt(j) == a){
              total = 1*population[i];
              //System.out.println("zip: " + zip[i] + "total: " + total);
              prob += total/denominator;
              break;
            }
          }
        }
      }//End of if good
    }//End of for loop through zip
    return prob/PopTotal;
  }

  //A method to calcuated the denominator of the posterior probability
  public static double denominator(){
    double denominator = 0.0;
    boolean good = true;
    for(int i = 0; i < zip.length; i++){
      good = true;
      for(int j = 0; j < 5; j++){
        if(GuessedZip[j] != -1){
          if(zip[i].charAt(j) != (char)('0' + GuessedZip[j])){
            good = false;
            break;
          }
        }
        else{
          for(int k = 0; k < 10; k++){
            if(numbers[k] == -1){
              if(zip[i].charAt(j) == (char)('0'+k)){
                good = false;
                break;
              }
            }
          }
        }
      }
      if(good){
        denominator += population[i];
      }
    }
    return denominator/PopTotal;
  }

  //Get the probability of the every first guess when no evidence needed
  public static double getProb(char a){
    double total = 0;
    for(int i = 0; i < zip.length; i++){
      for(int j = 0; j < 5; j++){
        if(zip[i].charAt(j) == a){
          total += 1 * population[i];
          break;
        }
      }
    }
    return total/PopTotal;
  }
  //An internal class for finding the most populated zip code
  static class Pair implements Comparable{
    public String zip;
    public Integer pop;

    public Pair(String zip, int pop){
      this.zip = zip;
      this.pop = pop;
    }
    public int compareTo(Object o)
    {
      Pair a2 = (Pair)o;
      if(this.pop > a2.pop)
        return 1;
      else if (this.pop < a2.pop)
```

```
        return -1;
      else
        return 0;
    }
  }
}
```