```
public class Frequencies {

public static final int LEN = 50;

public static void putchar(char c){ System.out.print(c);}

public static char low_case (char c){
 if ('A' <= c && c <= 'Z') return (char) ((int) c - (int)'A' + (int) 'a');
 else return c;
}

public static void initialize (int [] Table, int val, int N){
 for (int i=0; i < N; i++) Table[i] = val;
}

public static void draw_line (int len, char c){
 for (int i=1; i <= len; i++) putchar(c);
}

public static void count_freqs(int [] Table){
 char c;
 initialize(Table,0,26);
 while (!StdIn.isEmpty()) { c = StdIn.readChar();
 if (('A' <= c && c <= 'Z') || ('a' <= c && c <= 'z'))
  {c = low_case(c);
   Table[c-'a']++;}
 }
}

public static void display_freqs(int [] Table){
 System.out.printf("\n\nLetter Frequencies\n");
 for (char c='a'; c <= 'z'; c++){
  if ((c - 'a') % 5 == 0) putchar('\n');
  System.out.printf("%c%5d\t", c, Table[c-'a']);
}}
```

```
public static void draw_histogram(int [] Freqs){
 char c;
 int max = 0;
 System.out.printf("\n\nHistogram of Letter Frequencies\n");
 for (c='a'; c <= 'z'; c++)
  {System.out.printf("\n%c\t", c); draw_line(Freqs[c-'a']/2,'*');}

 for (c = 'a'; c <= 'z'; c++)
   if (max < Freqs[c-'a']) max=Freqs[c-'a'];

 System.out.printf("\n\nLetters with max frequency: ");
 for (c = 'a'; c <= 'z'; c++)
   if (Freqs[c-'a'] == max) System.out.printf(" %c  ", c);

 System.out.printf("\n\nLetters with zero frequency: ");
 for (c = 'a'; c <= 'z'; c++)
    if (Freqs[c-'a'] == 0) System.out.printf(" %c  ", c);
}

public static void main (String [] args) {
 int [] Freqs = new int[26];

 count_freqs(Freqs);

 display_freqs(Freqs);

 draw_histogram(Freqs);
 putchar('\n'); putchar('\n');

}

}
```

```
public class Athletes {

 public static final int NUM_ATHLETES = 10;

 public static void swap (int [] athlete_1, int [] athlete_2){
  int [] temp = new int[4];
  for (int i = 0; i < 4; i++) temp[i]= athlete_1[i];
  for (int i = 0; i < 4; i++) athlete_1[i]= athlete_2[i];
  for (int i = 0; i < 4; i++) athlete_2[i]= temp[i];
 }

 public static int min_pos (int [][] Table, int start_row, int end_row, int col){
  int pos = start_row;
  for (int i=start_row+1; i<=end_row; i++)
   if (Table[i][col] < Table[pos][col]) pos=i;
   else if (Table[i][col] == Table[pos][col] && Table[i][0] < Table[pos][0]) pos=i;
  return pos;
}

 public static void select_sort (int [][] Athletes, int col, int len){
  for (int i=0; i < len-1 ; i++)
    swap(Athletes[i], Athletes[min_pos(Athletes, i, len-1, col)]);
 }

 public static void get_data (int [][] Athletes, int num){
   for (int i=0; i<num; i++)
      for (int j=0; j < 4; j++)
           Athletes[i][j] = StdIn.readInt();
 }

 public static void display_data (int [][] Athletes, int rows, int col){
  switch (col) {
    case 1: System.out.printf("\n\nAthletes Performance in 100 m");
            System.out.printf("\n-----------------------------"); break;
    case 2: System.out.printf("\n\nAthletes Performance in 200 m");
            System.out.printf("\n-----------------------------"); break;
```

```
    case 3: System.out.printf("\n\nAthletes Performance in 400 m");
            System.out.printf("\n-----------------------------"); break;
  }
  for (int i=0; i < rows; i++) System.out.printf("\nAthlete %d has performance %d seconds", Athletes[i][0],
Athletes[i][col]);
 }

 public static void main (String[] args){
 int [][]Athletes_Data = new int [NUM_ATHLETES][4];
 get_data(Athletes_Data, NUM_ATHLETES);
 for (int i=1; i <= 3; i++) {
   select_sort(Athletes_Data,i,NUM_ATHLETES);
   display_data(Athletes_Data, NUM_ATHLETES,i);
  }
  System.out.println(); System.out.println();
 }

}
```

AthletesData.txt

```
1     10    22    40
2     17    25    47
3     11    19    39
4     14    20    41
5     18    25    42
6     12    24    39
7     10    20    30
8     13    24    46
9     14    26    47
10    10    19    32
```

```
public class CSet {

 protected static final int ALPHABET = 26;

 private int size;
 private boolean[] elements = new boolean[ALPHABET];

 public CSet (boolean empty){
   for (int i=0; i < ALPHABET; i++) elements[i]=empty;
   if (empty) size = 0; else size = ALPHABET;
 }

 public CSet (String cs){
  int count = 0;
  for (int i=0; i < ALPHABET; i++) elements[i]=false;
  for (int i=0; i < cs.length(); i++){
    if (is_letter(cs.charAt(i))){
       char newC = lower_case(cs.charAt(i));
       int k = (int) (newC - 'a');
       if (!elements[k]){elements[k] = true; count++;}
     }
  }
  size = count;
 }

 public CSet (CSet s) /* {this (s.toString());} */
  { for (int i = 0; i < 26 ; i++) elements[i] = s.elements[i];
    size = s.size;}


 protected static char lower_case (char c){
  char r = 'a';
  if ('a' <= c && c <= 'z')  r = c;
  else if ('A' <= c && c <= 'Z') r = (char) (c - 'A' + 'a');
  return r;
 }
```

```
protected static boolean is_letter(char c){
    return ('a' <= c && c <= 'z') || ('A' <= c && c <= 'Z');
}

public String toString(){
 String rs = ""; char c = 'a';
 for (int i = 0; i < ALPHABET; i++){
    if (elements[i]) {rs = rs + c;}
    c++;
 }
 return rs;
}

public void insertMember (char c){
   if (is_letter(c) && !elements[(int) (lower_case(c)-'a')]){
          size++;
          elements[(int) (lower_case(c)-'a')] = true;
   }
}

public void deleteMember (char c){
    if (is_letter(c) && elements[(int) (lower_case(c)-'a')]){
         size--;
         elements[(int) (lower_case(c)-'a')] = false;
    }
}

public int Size (){return size;}

public boolean isMember (char c){
   if (!is_letter(c)) return false;
   else return elements[(int)(lower_case(c)-'a')];
}

public boolean equal (CSet cs){
```

```
   return this.toString().equals(cs.toString());
}

public boolean subset (CSet cs){
  return this.toString().contains(cs.toString());
}

public CSet union (CSet cs){
 CSet xs = new CSet(this.toString() + cs.toString());
 return xs;
}

public CSet intersection (CSet cs){
  CSet xs = new CSet(false);
  for (char c = 'a'; c <= 'z'; c++)
         if (elements[(int) (c - 'a')])
              if (cs.isMember(c)) xs.insertMember(c);
  return xs;
}

public CSet difference (CSet cs){
  CSet xs = new CSet(false);
  for (char c = 'a'; c <= 'z'; c++)
         if (elements[(int) (c - 'a')])
              if (!cs.isMember(c)) xs.insertMember(c);
  return xs;
}

public CSet complement (){
  CSet xs = new CSet(true);
  return xs.difference(this);
  /* for (char c = 'a'; c <= 'z'; c++)
         if (elements[(int) (c - 'a')]) xs.deleteMember(c);
  return xs; */
}
```

```
 protected static CSet[] concat (CSet[] a, CSet[] b){
  int N = a.length + b.length;
  CSet [] r = new CSet[N];
  for (int i = 0; i < a.length; i++)
         r[i] = new CSet(a[i]);
  for (int i = a.length; i < N; i++)
         r[i] = new CSet(b[i - a.length]);
  return r;
 }

 public static CSet[] powerSet (CSet a){
    return powerSet(a.toString());}

 public static CSet[] powerSet (String s){
   if (s.length() == 0) {
       CSet[] r = new CSet[1];
       r[0] = new CSet(false);
       return r;}
   else {
      CSet [] r = powerSet(s.substring(1, s.length()));
      CSet [] x = new CSet[r.length];
      for (int i = 0; i < x.length; i++){
           x[i] = new CSet(r[i]);
           x[i].insertMember(s.charAt(0));}
      return concat(r,x);
   }
 }

 public static void main(String[] args){
   /* CSet cs = new CSet(true);
   System.out.print(cs);
   System.out.println(); System.out.println();
   CSet xs = new CSet(false); xs.insertMember('a'); xs.insertMember('w'); xs.insertMember('b');
xs.deleteMember('y'); xs.deleteMember('w');
   CSet as = new CSet(xs);
   System.out.print("as = " + as); System.out.print("\n\n");
```

```
   CSet ys = new CSet("elpida"), ps = new CSet("papailiou");
   System.out.println(ys.complement());
   System.out.println(ps.complement());
   System.out.println("cs = " + cs.complement());
   CSet [] A = new CSet[1];
   A[0] = new CSet(cs);
   CSet [] B = new CSet[1];
   B[0] = new CSet(as);
   CSet [] C  = concat(A, B);
   System.out.print("\n\n" + cs.toString());
   System.out.print("\n\n" + as.toString());
   for (int i = 0; i < C.length; i++) System.out.print("\n\n" + C[i].toString() + "\n\n");
  */

   CSet xs = new CSet(args[0]);
   CSet [] ps = powerSet(xs);
   for (int i = 0; i < ps.length; i++) System.out.print("\nps-" + i + " ->  " + ps[i].toString());
   System.out.print("\n\nEND\n\n");
   }
}
```

```
public class Anagram {

 public static final int WORDNUM = 100;

 public static boolean anagram(String w1, String w2){
    if (w1.length() != w2.length()) return false;
    else {for (int k = 0; k < w1.length(); k++) if (occurs(w1, w1.charAt(k)) != occurs(w2, w1.charAt(k))) return
false;
          return true;}

 }

 public static int occurs (String s, char c){
   int n = 0;
   for (int k = 0; k < s.length(); k++) {if (c == s.charAt(k)) n = n+1;}
   return n;
 }

 public static int ReadWords(String[] Words){
    int n = 0;
    while(!StdIn.isEmpty()){
           Words[n] = new String(StdIn.readString()); n = n+1;}
    return n;
 }

 public static void main (String [] args){
      String [] Words = new String[WORDNUM];
        int size = ReadWords(Words);
      for (int w1 = 0; w1 < size - 1; w1++)
            for (int w2 = w1+1; w2 < size; w2++)
                  if (anagram(Words[w1], Words[w2]))System.out.printf("\n%s = %s", Words[w1], Words[w2]);
        System.out.println(); System.out.println();

 }

}
```

```
public class Prog{


public static String[] Fn (String[] as, String[] bs){
  int N = as.length + bs.length;
  String [] rs = new String[N];
  for (int i = 0; i < as.length; i++)
         rs[i] = new String(as[i]);
  for (int i = as.length; i < N; i++)
         rs[i] = new String(bs[i - as.length]);
  return rs;
 }

 public static String[] Fn (String s){
   if (s.length() == 0) {
       String[] xs = new String[1];
       xs[0] = new String("");
       return xs;}
   else {
      String [] xs = Fn(s.substring(1, s.length()));
      String [] ys = new String[xs.length];
      for (int i = 0; i < ys.length; i++){
           ys[i] = new String(xs[i]);
           ys[i] = ys[i].concat(s.substring(0,1));}
      return Fn(xs,ys);
   }
 }

 public static void main (String[] args){
   String [] ss = Fn(args[0]);
   for (int i = 0; i < ss.length; i++) System.out.println("ss-" + i + " = " + ss[i]);
 }
}
```

```
$ java Prog ABC
ss-0 =
ss-1 = C
ss-2 = B
ss-3 = CB
ss-4 = A
ss-5 = CA
ss-6 = BA
ss-7 = CBA
```

```
public class Unknown {

  public static void Rn (double s, int N){
    if (StdIn.isEmpty()) System.out.println("Result = " + (s/N));
    else {
       double x = StdIn.readDouble();
       Rn (s+x, ++N);
       System.out.println(x);
    }
  }

  public static void main(String[] args){
      Rn(0.0, 0);
  }

}
```

Data.txt

```
34 25 67 90 10
```

```
$ java Unknown < Data.txt
Result = 45.2
10.0
90.0
67.0
25.0
34.0
```

```
public class Unknown {

  public static void Rn (double s, int N){
    if (StdIn.isEmpty()) System.out.println("Result = " + (s/N));
    else {
       double x = StdIn.readDouble();
       Rn (s+x, N++);
       System.out.println(x);
    }
  }

  public static void main(String[] args){
      Rn(0.0, 0);
  }
}
```

Data.txt

```
34 25 67 90 10
```

```
$ java Unknown < Data.txt
Result = Infinity
10.0
90.0
67.0
25.0
34.0
```