Problem 1 (4 pts) As a function of *n*, what is the ***exact number of lines of output*** of the call `lois(n)`?

```
void lois(int n)  {

   int i = 1, j = 1, k = 1;

   while (i <= n && j <= n && k <= n)  {

      sopln(i + ", " + j + ", " + k);
      k++;

      if (k > n) {
         j++;
         k = 1;

          if (j > n) {
             i++;
             j = 1;
          }
      }
   }
}
```

Problem 2 (8 pts) Consider the following method:

```
int meg(int n, int [] A)
{
   int i = 1, ans = 0;

   while (i < n) {

      if (A[i] > A[i+1]) {
         ans += A[i]
         i = i * 2;
      else
         i++;
   }
   return ans;
}
```

(a) In terms of θ-notation, what is the worst-case running time of `meg`?

(b) In terms of θ-notation, what is the best-case running time of `meg`?

Problem 3 (4 pts) In terms of θ-notation, what is the running time of `stewie`?

```
int stewie(int n, int [] A)  {

   int i, j, sum = 0;

   for (i = 1; i <= n; i = i * 2)
      for (j = 1; j <= i; j++)
         sum = sum + j * A[i];

   return sum;
}
```

Problem 4 (8 pts) Consider the following method:

```
int brian(int n, int [] A, int [] B)  {

   int i = 1, j = 1, ret = 5;

   while (i <= n && j <= n)  {

      ret = ret + A[i] * B[j];

      if (A[i] > B[j])
         i++;
      else
         j++;
   }

   return ret;
}
```

4.1) As an ***exact function*** of *n*, what is the **best**-case (fewest) number of array element comparisons made by method `brian`? Give arrays *A* and *B* of size $n = 6$ that achieve this best case.

4.2) As an ***exact function*** of *n*, what is the **worst**-case (largest) number of array element comparisons made by method `brian`? Give arrays *A* and *B* of size $n = 6$ that achieve this worst case.

Problem 5 (8 pts) In order to receive ***ANY credit*** for this problem, your method ***must run in time $O(n)$***.

Write the method below which takes an integer *n* and an array *A* (indexed from 1 to *n*) and determines if it is a permutation of the set $\{1, 2, \ldots, n\}$. In other words, the method returns true if each value in the set $\{1, 2, \ldots, n\}$ is located somewhere in the array:

```
boolean isPermutation(int n, int [] A)
```

For example, the array below is not a permutation with $n = 4$ because the value 2 is missing:

1	2	3	4
3	1	4	3

The following array is not a permutation with $n = 6$ because the value -90 is out of range:

1	2	3	4	5	6
2	1	-90	6	5	3

But the following array is a permutation with $n = 7$:

1	2	3	4	5	6	7
7	3	2	6	1	5	4

In addition to returning true or false, if the array is NOT a permutation, your method should print out a reason why (see sample output).

Problem 6 (8 pts) In order to receive ***ANY credit*** for this problem, your method ***must run in time $O(n^2)$***.

Write the method below which takes a sorted array *A* (indexed from 1 to *n*) and an integer *g*. This method determines if there are three distinct indices *i*, *j*, and *k* such that $A[i] + A[j] + A[k] = g$. If there are, it returns the array `{i, j, k}`. If not, it returns the array `{-1}`.

For example, if passed array *A* below with *g* = 96, it would return the array `{2, 5, 7}`, since $A[2] + A[5] + A[7] = 8 + 37 + 51 = 96$. However, for the same array with *g* = 101, it would return the array `{-1}` since no three distinct elements in *A* sum to 101.

	1	2	3	4	5	6	7	8
A	5	8	16	22	37	44	51	82

```
int [] sumOfThree(int n, int [] A, int g)
```

Problem 7 (8 pts) Without declaring **ANY** arrays, write the method below, which finds the most frequent element in array *A*. **In a comment at the top of your program, what is the running time of your method, in terms of θ-notation?**

```
int mostFrequent(int n, int [] A)
```

Problem 8 (8 pts) For this problem, you **MAY** declare auxiliary arrays. Write a new version of the "mostFrequent" method from Problem 7, ***but this time it must run in time*** $\theta(n)$. ***Assume that all elements in array A are between 1 and n***.

Problem 9 (+3 pts extra credit) As a function of *n*, what is the ***exact number of lines of output*** of the call `joe(n)`?

```
public static void joe(int n)  {

   int i = 1, j = 1, k = 1;

   while (i <= n)  {
      j = i+1;
      while (j <= n) {
         k = j+1;
         while (k <= n) {
            System.out.println(i + ", " + j + ", " + k);
            k++;
         }
         j++;
      }
      i++;
   }
}
```

Problem 10 (+3 pts extra credit) In terms of θ-notation, what is the running time of "mystery"?

```
public static void mystery(int n) {

   int j = 1, sum = 0;
   while (sum < n)  {
      sopln(j);
      sum += j;
      j++;
   }
}
```