There are 4 problems. You can assume that the input to your program conforms to the description given in each problem, so you do not need to check for valid input.

1. (Next Prime) Write a program to find the smallest prime number that is at least as large as an input number. (A prime number is an integer divisible only by 1 and itself.) The input to your program is an integer $m$. The output is a single integer $p$ that is prime, $p \geq m$, and there is no other prime between $m$ and $p$.

Examples: for input 6, the output should be 7. For input 7, the output should be 7. For input 24, the output should be 29.

It may be useful, or at least reassuring, to know that for any positive integer $m$, there exists at least one prime number between $m$ and $2m$.

2. (Perfect Candidate) Suppose that a committee of $n$ people are trying to decide on a chairman. Each person has a subset of other members (which includes themselves) who they support for chairman. We can represent the preferences with a matrix that has a 1 in the $(i, j)$th position if member $i$ supports member $j$, and a 0 otherwise.

Before holding an election, the committee decides to compare their preferences and see if there is an individual who supports only themselves for chairman, and who everyone else also supports; call such a member a perfect candidate. A perfect candidate need not exist, but it is easy to see that there can not be more than one. Write a program that takes as input a matrix representing preferences, and prints the index (a number between 1 and $n$) of such a perfect candidate if one exists, and prints “none” if there is no perfect candidate.

Assume that the input is in the form: a first line containing the size $n$ of the committee, and then $n$ subsequent lines, each containing $n$ integers that are either 0 or 1, separated by a space. A 1 in row $i$ and column $j$ of the matrix indicates that member $i$ supports member $j$, and a 0 indicates that $i$ does not support $j$. For example, if the input is

\[
\begin{align*}
4 \\
1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{align*}
\]

then your program should print “2”, since all four candidates support the second candidate, while the second candidate supports only himself.
3. (Anagrams) An anagram is a word or phrase made by transposing the letters of another word or phrase. Let’s ignore case, spaces, and punctuation in forming anagrams. With this convention, some examples of anagram pairs are:

<table>
<thead>
<tr>
<th>Snooze Alarms</th>
<th>Alas! No More Z’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astronomer</td>
<td>Moon Starer</td>
</tr>
<tr>
<td>The eyes</td>
<td>They see</td>
</tr>
</tbody>
</table>

Note that each letter must be used once and only once.

Write a program that reads a line of text from the input and prints out true if the text represents an anagram of “contradiction” and false otherwise.

4. (Cargo Transport) Suppose that you have a collection of \( n \) items that you want to transport on an aircraft that can carry \( C \) tons. The \( i \)th item has weight \( w_i \) and value \( v_i \), for \( i = 1, 2, \ldots, n \). Assume that \( w_1 + w_2 + \cdots + w_n > C \), so that you can not load everything onto the plane. Write a program that finds the largest combined value of any subset of items that has total weight at most \( C \).

The input is a first line containing two integers, \( n \) and \( C \). Each of the following \( n \) lines contains a pair of integers; the first is the weight of the item, and the second is the value. The output is a single integer giving the maximum combined value of a feasible shipment. For example, if the input is

\[
\begin{align*}
5 & \quad 15 \\
4 & \quad 7 \\
8 & \quad 12 \\
3 & \quad 6 \\
5 & \quad 11 \\
2 & \quad 5 \\
\end{align*}
\]

then you have 5 items to ship on a plane that can carry at most 15 tons. The first item weighs 4 tons and has value 7, the second item weighs 8 tons and has value 12, and so on. Your program should print 29, since the items with weights 4, 3, 5 and 2 have total weight 14 \( \leq \) 15 and total value 7 + 6 + 11 + 5 = 29. This is better than the alternatives. For example, if we chose the items with weights 8, 5 and 2, the total weight would be 15 but the total value only 12 + 11 + 5 = 28. You can assume that \( n < 20 \).