Camp Out
This problem should look similar to the Museum Guard problem covered in the network flow lecture. In this problem, the difference is that there is a fixed number of students (10), and that the number of shifts is 42 (6 per day, 7 days for the whole week). From the source, put an edge to each student with capacity 20 (max 80 hours for the week for any student is 20 shifts of 4 hours), from each student, only put edges with capacity one to the shifts that they can completely cover and from each of the shifts put an edge of capacity 3 to the sink to indicate that we want 3 students to cover every shift. If the maximum flow through this graph equals $3 \times 42 = 126$, then the students can man the tent. If it is any lower than that, they can’t man the tent.

Plane and a Line
Get the plane equation by taking two vectors on the plane, getting their cross product to get the normal vector to the plane and the taking the dot product between this normal and a point on the plane. Then, plug in the parametric equation of the line into the plane equation and solve for lambda, if the lambda coefficient is non-zero. In this case, the calculation will yield a unique point of intersection on the line (plugging in the solution for lambda.) If the coefficient to lambda is 0, then you must check to see if the constant term in the equation was 0 or not. If it was, the line is on the plane and if it was not, there is no intersection.

Islands in the Data Stream
Since the input is so small, just try all possible starting and ending points and see if it satisfies the definition of an island.

Lucas's Letters
There are only $2^{16}$ possible subsets of words Lucas could form. That means there is enough time to try each subset, and for each subset see if he has enough letters, and if so, score the subset. Of all viable subsets, return the best score. If written recursively, we could just keep track of the current number of tiles Lucas has left in all possible states, potentially backtracking when he doesn't have enough tiles to form a word. The bounds are such that without this backtracking a solution runs in time.

Evil Ninja vs. Tetra-Bot
This problem is easier than it looks. Recall that given a single point, we can calculate the distance of that point from the plane. Furthermore, the "D" value in the parallel plane equation indicates which side of the plane the point is on. If all four points of the tetrahedron are on the same side of the ninja plane and not touching it, Tetra-Bot Survives! Alternatively, if any of the four points are either on the plane or if any pair of points are on opposite sides of the plane, the Evil Ninja Prevails! Thus, calculate the plane equation using the three plane points. (Get two vectors on the plane, take their cross product, and then plug in one of the three plane points to get the constant D.) Then, plug in all four tetrahedron points into the same plane equation, getting four new values of D, the four parallel planes on which the tetrahedron points lie. If all four are greater than D or if all four are less than D, Tetra-Bot Survives! Otherwise, the Evil Ninja Prevails!

Perfect Shuffle
Just do exactly as it says. To avoid confusion, copy all values from the original array into a new array and pay attention to indexing issues. Make sure to handle both the even and odd input cases.