1. Assume you have a directed graph $G = (V, E)$ represented by an **adjacency list**. Assume that your graph consists only of simple cycles (so that every vertex is in one and only one simple cycle). Give a $\Theta(|V|)$ time algorithm to print the cycles. Briefly justify the time of your algorithm.

2. Assume you have a directed graph $G = (V, E)$ represented by an **adjacency matrix**. Assume that your graph consists only of simple cycles (so that every vertex is in one and only one simple cycle). Give a $\Theta(|V|^2)$ time algorithm to print the cycles. Briefly justify the time of your algorithm.

3. Assume you have an undirected graph $G = (V, E)$. Give an efficient algorithm to find one cycle if there is one, and otherwise print "no cycle". You may assume the graph is represented by an **adjacency matrix** or **adjacency list**, whichever you prefer. State your representation and give the pseudo code. Briefly justify the time of your algorithm (in Theta notation).

4. CHALLENGE PROBLEM (not part of your grade). Assume that you have a path (graph) with n vertices. Assume that you randomly choose a vertex and keep it, but delete its (up to two) neighbors. Continue in this fashion until the graph is empty. For example, if $n = 3$, if initially the first vertex is chosen that leaves only the last vertex, which must be chosen next. If initially the last vertex is chosen that leaves only the first vertex, which must be chosen next. If initially the second vertex is chosen that deletes both other other vertices. So on average you will keep $\frac{5}{3}$ vertices. If $n = 4$ on average you will keep two vertices (in fact you will always keep exactly two vertices). If $n = 1$ or $n = 2$ you will keep exactly one vertex.

   On average, how many vertices will you keep? Just give the high order term (for n large).