A. [Strangely, Problem 8.4 omits this operation.] Yes, $PSPACE$ is closed under concatenation. Let $A, B \in PSPACE$. Let M, N be deciders for A, B that use space $\mathcal{O}(n^k), \mathcal{O}(n^\ell)$. Define a Turing machine R that, on input w, tries splitting w into all possible concatenations $w = xy$. For each split, R runs M on x and N on y. If in any split M and N both accept, then R accepts. Otherwise, R rejects. Then R decides AB. It can use the same space to decide each split, and the space required is no more than $\mathcal{O}(n^k) + \mathcal{O}(n^\ell) = \mathcal{O}(n^{\max(k,\ell)})$.

B. It is easy to see that $A \in NP$, because a verifier for A could take the stranger set as a certificate and check that the stranger set is correct. To show that A is NP-hard, we reduce $3SAT$ to A, as suggested. Given a 3CNF Boolean formula ϕ, we build a graph G. This G has one node per literal in ϕ. For each clause in ϕ, we connect all three corresponding nodes in G. Further, we connect each node to any other node containing the opposite literal — for example, we connect x nodes with $\bar{x}$ nodes. The reduction outputs $\langle G, k \rangle$, where k is the number of clauses in ϕ. The reduction requires only polynomial time. Now we check that it works.

Suppose that ϕ is satisfiable. In a satisfying assignment, at least one literal in each clause is made true. Select one true literal from each clause. The corresponding set of k nodes in G is a stranger set. Why? None of the intra-clause edges have two nodes in the stranger set, because the stranger set contains just one node per clause. None of the inter-clause edges have two nodes in the stranger set, because those two nodes would be marked with opposite literals (e.g., x and $\bar{x}$) that can't both be true.

Conversely, suppose that G has a stranger set of size k. No two nodes within a single clause can be in this stranger set. Therefore the stranger set contains exactly one node per clause. Choose a truth assignment for ϕ that makes these literals true (and sets other literals arbitrarily). How do we know that such an assignment can be chosen? The only problem that could arise is that two true literals in different clauses contradict each other — for example, x and $\bar{x}$. But such nodes would share an edge in G, and hence could never belong to the same stranger set.

C. [This problem is similar to the second half of Spring 2012, Exam B, Problem C.] Suppose, for the sake of argument, that a Turing machine R recognizes A. We will use R to build a recognizer P for $\overline{HALT_{TM}}$. Upon input $\langle M, w \rangle$, P performs these steps:

1. Build a Turing machine N that, on input x, runs M on w for $|x|^2$ steps. If M accepts or rejects during this time, then N enters into an infinite loop. If M does not accept or reject during this time, then N accepts.

2. Run R on $\langle N \rangle$, and output whatever R outputs.

If $\langle M, w \rangle \in \overline{HALT_{TM}}$, then M does not halt on w, so N accepts every input x. Specifically,

N accepts x in time $\mathcal{O}(|x|^2)$, so R accepts $\langle N \rangle$ and P accepts $\langle M, w \rangle$. Conversely, if $\langle M, w \rangle \in HALT_{TM}$, then let t be the time required by M to halt on input w. On all sufficiently long inputs x — namely, those x such that $|x|^2 \geq t$ — N fails to halt. Thus R does not accept $\langle N \rangle$ and P does not accept $\langle M, w \rangle$. Thus P is a recognizer for $\overline{HALT_{TM}}$, which is not recognizable. This contradiction implies that A is not recognizable either.