A.A. [This was an uncollected homework problem.] Our classic PDA solution loaded left parentheses onto the stack, popping them off as they were matched to right parentheses. So the stack space used was $\mathcal{O}(n)$. (In fact, the worst-case space usage was exactly n. This worst case happened when the input string was $(^n$.)

A.B. [This was an uncollected homework problem.] In essence, the algorithm scans the input once, left to right. As it scans, it keeps track of how many left and right parentheses it has seen. If it any time it has seen more right parentheses than left parentheses, then it rejects. If it makes it to the end of the input, and the numbers of parentheses are equal, then it accepts. Otherwise, it rejects.

In actuality, there is not just one scan, because the algorithm needs to manage extra memory (on the end of the tape after the input, or on a second tape). This extra memory stores a single non-negative integer, which counts the excess left parentheses seen thus far. This extra memory takes logarithmic space, because it just needs to store a single number, and that number is $\mathcal{O}(n)$ and hence requires only $\mathcal{O}(\log n)$ space when written in any base $b \geq 2$.

B.A. *MAX–CUT* is the set of strings $\langle G, \ell \rangle$, where G is an undirected graph, $\ell \geq 1$ is an integer, and there exists a cut of G that has size ℓ or greater.

B.B. I describe a polynomial-time verifier V. On input of the form $\langle G, \ell, c \rangle$, V does these steps:

1. Interpret c as a subset S of the nodes in G, which form one side of a cut. The nodes not listed in c are taken to be the other side T of the cut.
2. Loop over all of the nodes in S, counting how many edges they share with nodes in T, to compute the size of the cut.
3. If this size is at least ℓ, then accept; otherwise, reject.

The running time of V is a low-degree polynomial in the number of nodes in G. Hence it is some low-degree polynomial in $|\langle G, \ell \rangle|$, as desired.

C. We must check two or three things, depending on how one counts. First, if there exists a satisfying assignment for ϕ, then there exists an $\neq$-satisfying assignment for ψ. Second, if there exists an $\neq$-satisfying assignment for ψ, then there exists a satisfying assignment for ϕ. Third, the time complexity of F is polynomial in $n = |\langle \phi \rangle|$.

D.A. Every TM is trivially also an NTM (that happens never to use its non-determinism).

D.B. This result follows from Savitch's theorem (and is otherwise far from obvious).

D.C. This result follows from our basic space-time lemma.

D.D. A bound on the space used gives a bound on the number of configurations and hence a bound on the running time possible before the TM enters into an infinite loop.

E. The ϕ_{move} part of ϕ requires that every 2×3 "window" of the tableau be "legal". For the formula to be satisfied, the two rows of a legal window either must be identical or must involve the area of the tape very close to the tape head, where the tape head alters the contents of a single cell and then moves left or right, as indicated by the location of the state in the configuration.