Problem 1: (10 points) In class we “showed” that a van Emde Boas tree in which clusters are stored in hash tables uses $O(n)$ space. Recall that we argued as follows, where a $u$-vEB tree stores integers in the range $\{0, \ldots, u - 1\}$:

1. A $u$-vEB tree contains three types of pointers: (i) one to its min, (ii) one to a summary $\sqrt{u}$-vEB, and (iii) several pointers, in a hash table, to cluster $\sqrt{u}$-vEBs.
2. Each cluster pointer can be “charged to” the minimum in that cluster.
3. Each summary pointer of a $u$-vEB structure $V$ can be “charged to” $V$’s own minimum.
4. Thus, each min pointer receives charges from at most 2 pointers of types (ii) and (iii).
5. Therefore the total space is big-Oh of the total number of min pointers.
6. Therefore the total space is big-Oh of the total number of items, which is $n$.

Now for the problems:

(a) (1 points) Which of the following statement(s) above (from (1)-(6)) are wrong? What is the potential error?

(b) (1 points) Show that a $u$-vEB tree on $n$ items consumes space $O(n \lg u)$.

(c) (2 points) Show that a $u$-vEB tree on $n$ items consumes space $O(n \lg \lg u)$ (note if you solve this, you don’t need to solve part (b) separately).

(d) (4 points) Give a family of examples of $n$ items that, when stored in a $u$-vEB tree, consume space $\Omega(n \lg \lg u)$. In your family of examples, $n$ and $u$ should go to infinity.

(e) (2 points) How would you use indirection to modify vEB trees to solve the static predecessor problem with $O(n)$ space and $O(\lg \lg u)$ query time?
Problem 2: (5 points) Let \( w \) be a perfect square. Show that there exist positive integers \( m \) and \( t, m < 2^w \) and \( 0 \leq t \leq w \), such that for all \( x \in \{0, 1\}^{\sqrt{w}} \) we have that

\[
\left( \left( \left( \sum_{i=1}^{\sqrt{w}} x_i \cdot 2^i \right) \cdot m \right) \gg t \right) \& (2^{\sqrt{w}} - 1) = \sum_{i=1}^{\sqrt{w}} x_i \cdot 2^{i-1}.
\]

That is we can pick \( m \) and \( t \) so that, if we form a bitvector of length \( w \) which has the \( \sqrt{w} \) bits of \( x \) evenly spread out with a \( \sqrt{w} \)-spacing of zeroes in between bits, then multiplying by \( m \) and bitshifting right by \( t \) followed by masking perfectly compresses the bits of \( x \) into the rightmost \( \sqrt{w} \) bits of a machine word. This provides the proof of a lemma we needed for \( O(1) \) time most significant set bit in Lecture 2.

Problem 3: (10 points) Give an algorithm for computing the least significant set bit of a given input word in constant time. You may assume that you have spent some time in pre-processing to pre-calculate any special constant values that your algorithm needs.