We want to prove:

**Theorem 1.** $K(x)$ is not a computable function.

*Proof.* Assume, by contradiction, that $K(x)$ is computable. This means that there is a Turing machine $M$, that given input $x$ computes $K(x)$. The idea of the proof is that we can use the ability to compute Kolmogorov complexity to construct another machine which identifies and prints out a string of high Kolmogorov complexity – but, since the string has high Kolmogorov complexity, this should be totally impossible. We need to pick a high-complexity string which is easy to “find”. We do this by just looping through all strings and outputting the first high-complexity string we encounter.

Consider the function $FirstIncompr(n)$ that for every $n$ outputs the lexicographically first incompressible string of length $n$,

**Claim 2.** For every $n$, $FirstIncompr(n)$ is defined, and computable.

*Proof of claim.* We have proven in class that there are incompressible strings of every length. (By incompressible string of length $n$, we mean a string $w$ of length $n$, with $K(w) > n - 1$) So the function is well defined. Several students forgot to justify that this function is well-defined i.e. that incompressible strings exist therefore the Turing Machine below halts.

We now sketch a Turing machine that computes $FirstIncompr(n)$, assuming $K(x)$ to be computable.

On input $n$:
For $z \in \{0, 1\}^n$:
    Compute $K(z)$.
    If $K(z) \leq n - 1$ go to the next string, otherwise we found the lexicographically first incompressible string of length $n$. Output it.

Since we know that there are incompressible strings of every length, the loop above will always succeed. □

The high-level procedure above can be implemented by some Turing machine $N$. Now consider the sequence $a_1, a_2, \ldots a_m \ldots$ where $a_i = FirstIncompr(i)$.
$a_m$ can be specified by the Turing machine $N$, and the index $m$.
$N$ is specified by a constant-length string $T$ (an encoding of the set of 5-tuples defining $N$). Specifically, the pair $(T, m)$ is a description of $a_m$. 

1
This description requires only \( C + O(\log m) \) bits. (We saw that we could encode pairs by doubling the length of the first string using the encoding 0 \( \rightarrow \) 00, 1 \( \rightarrow \) 11 and \( \) \( \) \( \) \( \) \( \rightarrow \) 01 then concatenating the two strings. So \( C = 2|T| + 2 \).) Some online sources make the mistake that the encoding of \( \langle A, B \rangle \) can be done using \(|A| + |B| + O(1)\) bits. Minor nitpick: \( m \) requires \( \lceil \log_2(m) + 1 \rceil \) bits to represent, with edge case 1 bit for \( m = 0 \). Safe to just write \( O(\log m) \).

This proves the complexity of \( a_m \) is at most \( C + O(\log m) \).

On the other hand, \( a_m \) is incompressible: its shortest description has length at least \( m \). For large enough \( m \), \( m > C + O(\log m) \) for any fixed \( C \).

This is a contradiction.

\[ \square \]

**Comments** A proof could be less detailed and still get full credit.

The description of the algorithm for finding the first incompressible string could be simplified, with some details omitted.

For example, one could do away with the definition of the function \( \text{FirstIncompr()} \), and argue directly from the code of the Turing machine \( N \).

Some correct solutions that got points taken away: You have to be careful what is it that you are contradicting. Some students used the contradiction above to claim \( A_{TM} \) would be decidable. Here we are just producing a straight contradiction.

When claiming that there is an algorithm, sketch it, unless it is obvious. Also, if you make statements like “do x until y” you have to argue/prove that y will occur.