*[EXAMPLE TEMPLATE]*

**WEEK 12**

**KU ID: 100059726**

**NAME: Obaid Saif Alhmoudi**

|  |
| --- |
| Q1.  Please write your GitHub Repository link for your week 12 Rustlings submission: |
| ANSWER:  https://github.com/Alhammoudi/rustlings/tree/main/WEEK12 |
| Q2.   1. Please **Find 1 (one)** specific part of the c2rust code that uses an **iterator** 2. and explain **why** that part uses an iterator **instead of a for loop**   -Important: Please also mention the **URL** where you found that iterator  -Please find **only in \*.rs** file(s) (not the \*.md, \*.html, \*.c, \*.h, etc. ) |
| ANSWER:  One part of c2rust that uses an iterator is in the c2rust-transpile/src/c\_ast/iterators.rs  In the code, there is an iterator for the DF nodes:  impl<'context> Iterator for DFNodes<'context> {  type Item = SomeId;  fn next(&mut self) -> Option<Self::Item> {  let result = self.stack.pop();  if let Some(i) = result {  // Compute list of immediate children  let children = immediate\_children\_all\_types(self.context, i);  // Add children in reverse order since we visit the end of the stack first  self.stack.extend(children.into\_iter().rev())  }  result  }  }  It uses an iterator because the AST (which it is iterating over) is a complex tree structure and the purpose of the iterator for DFNodes is making it easier to move through the tree step by step in a depth first way. It keeps the code clean and separates how to walk through the tree from what to do with each node.  Using a for loop wouldn’t work well here because the logic to visit eah node in correct would be too complex |