[EXAMPLE TEMPLATE]

**WEEK 9**

**KU ID: 100059727**

**NAME: Obaid Saif Alhmoudi**

|  |
| --- |
| Q1. Rustlings exercise GitHub link: |
| ANSWER:  https://github.com/Alhammoudi/rustlings |
| Q2.  A. Find and explain two traits used in C2RUST? (also mention the URLs)  B. Explain where those are used, and why?  C. Explain the alternatives to using traits? |
| ANSWER:  A:  <https://github.com/immunant/c2rust/blob/master/c2rust-refactor/src/ast_manip/fold.rs>  c2rust-refactor/src/ast\_manip/fold.rs  The first trait found in this file is *MutVisitor,* it is like a visitor pattern specifically designed for mutable AST transformations. It defines a series of methods that allow walking over and rewriting different parts of the tree. Each method handles a specific kind of node, like function declarations, blocks, statements or expressions. This trait allows developers to implement custom logic that changes AST as needed.  The second trait is *MutVisit*, it works with the first trait. While MutVisitor defines how the traversal happens, MutVisit provides the interface for AST nodes to accept visitors. If a type implements MutVisist, it can be visited and maybe changed by a MutVisitor. This trait ensures that ech node knows how to allow visitors to traverse its children.  B: Both those traits are used in C2’s refactoring tool, which is responsible for analyzing and transforming the Rust code produced. The produced code from C is often low-level and not structured, so the refactor tool applies changes to restructure it and improve it.  C: Instead of using MutVistor and MutVisit, C2Rust could have used more traditional ways like manual recursive functions or the visitor pattern.  The first alternative is to manually write the functions that traverse and modify the AST node by node. This would require explicit recursion at every level, which would take a lot of time and be difficult to maintain, especially when working with complex trees.  The other alternative would be to use the classic visitor pattern, where a separate visitor class defines methods for each node type. This method is better for reading and analysing and it does not integrate as well with Rust’s ownwership or borrowing rules, so maybe another alternative is using external libraries. |
|  |