Rabeea Shahid

2.a. provide an explanation of the function, the answer should provide a description of what the function does, and a detailed explanation of each parameter of the function.

The cons(car, cdr) function forms a list with the car element prepended in front of the original cdr list. The cdr list isn't modified, instead a new list is created and returned. The car parameter is the element that is to be prepended to the list. It can be of any type. The cdr parameter is the original list that gets an element inserted in front of. It can be of any type as well. The function returns the new list with the element prepended and the cdr list being placed afterwards. It is of the type List<A> so a list of any type.

Reference:
https://docs.rs/im/5.0.0/im/list/fn.cons.html

3. Executing the previous example should result in an error, mention the error and explain how interior mutability can be applied to the problem to solve it.

The error is that the value 100 cannot be assigned to task.id because task is not declared as mutable (does not have a "mut" after the "let" statement). Interior mutability can be applied to solve it by making only the id field of the task mutable while keeping the task variable as immutable itself. This allows for only the id field to be modifiable. This can be done by wrapping the id field with Cell<T> to allow it to change value.

4. a. Explain what the program is doing?

The program creates two double nodes, node A which has a value of 100 and no next and prev node links, and node B which has a value of 1000 and has a next node link to A and no prev node link. Thus, B points to A next but A shows no relationship with another node through a link. After printing the nodes, A's prev node link is modified to point to B. Now, B points to A and A shows that B points to it, so the link is reflected in both nodes. The nodes are then printed again.

4. b. Explain the data structure DoubleNode; what is it trying to implement?

The data structure DoubleNode is a node that allows a node to hold information related to what node comes after it as well as what node comes before it. It is trying to implement a doubly linked list so that iterations through the linked list can be done both ways, which is only possible if a node's precedent and successor are known for each node.

Reference:
https://www.geeksforgeeks.org/dsa/doubly-linked-list/

4. c. Explain how Weak<RefCell<Option<DoubleNode>>> differs from

Rc<RefCell<Option<DoubleNode>>>?

The two differ in the first wrapping with Weak<T> and the second wrapping with Rc<T>. These differ by the Weak<T> wrapping (weak reference) being a version of Rc<T> (strong reference) that references the allocation without owning the reference. Thus, the first does not count towards ownership while the second that uses Rc<T> does. Thus, the first that uses Weak<T> will not prevent the value in the allocation from being dropped when strong references reach zero while the second that uses Rc<T> will as it itself counts as a strong reference. The first also makes no guarantee that the value in the allocation is still present when accessed while the second does.

References:
https://doc.rust-lang.org/std/rc/struct.Weak.html
https://www.reddit.com/r/learnrust/comments/t0kt2m/weak_and_strong_references_rc_vs_weak/

4. d. Explain what is achieved by the line if let Some(ref mut x) = *a.borrow_mut() {(*x).prev = Rc::clone(&b);}

This line first checks if node A is a node that exists (by calling Some()), and if so, changes its prev node link field to point to node B through a new cloned Rc<T> wrapper. This makes it so that A shows that B points to it.

References:
https://doc.rust-lang.org/std/borrow/trait.BorrowMut.html
https://doc.rust-lang.org/rust-by-example/std/rc.html