Notes

Original recordings can be provided upon request.

Subject 1

Task #1

1. 0:30 - Identifies data declaration for “Nat”, but thinks it’s incomplete due to empty boxes
2. 0:40 - Realizes the data declaration is not incomplete
3. 0:50 - Identifies the data declaration for “Fin”

Task #2

4. 1:25 - Hits the plus button to add more data
5. 1:40 - Sets the name correctly
6. 2:00 - Is confused where different parts of “Nat -> Type -> Type” go
7. 2:40 - Realizes that “Nat -> Type” should be in the premiss, and “Type” should go in the conclusion.
8. 3:00 - Correctly writes “Nil” constructor, correcting “Vect n a” to “Vect Z a”
9. 3:50 - Writes “(::)” case, but is confused about whether “a” should have a type
10. 5:10 - Corrects the size of the vector to “Vect (S n) a” in the cons case.
11. 5:20 - Has trouble with the onscreen keyboard

Task #3

12. 6:05 - Hits the plus button to add a function
13. 6:10 - Correctly sets the type
14. 7:10 - Quick case splits and autocompletes through the rest of the function

Task #4

15. 7:50 - Moves data definition without any problems

Feedback

16. 8:20 - Thinks the way data is written is fine, but it takes getting used to
17. 8:55 - The placeholders “Identifier” and “Type” in data declarations is helpful
18. 9:05 - Does not think the way data is written is better than the standard way
19. 9:55 - Usually thinks about type constructors as functions, this way of writing it is incongruent with that.
20. 10:35 - All the grey input fields are confusing. Makes you think you haven’t finished.
21. 11:20 - Likes the idea of improved flow between input fields
22. 11:40 - Likes the idea of more help when choosing a suggestion

Subject 2

Task #1

1. 0:30 - Identifies the Nat type, along with its constructors, correctly
2. 0:50 - Is confused by the extra grey input fields
3. 1:20 - Identifies the Fin type, along with its constructors, correctly

Task #2

4. 2:15 - Hits the plus button and chooses daga
5. 3:00 - Sets the correct type after being confused by bugs with the auto suggestion
6. 3:30 - Correctly defines the Nil constructor, changing “Vect n a” to “Vect Z a”
7. 4:50 - Correctly defines the Cons constructor, but is irritated when the auto suggestion suggests “Vect n a” and not “Vect (S n) a”, and has to write the full type on the keyboard.

Task #3

8. 5:35 - Correctly defines zip, but is again irritated by having to write out the type on the keyboard because auto suggest doesn’t suggest the needed type
9. 5:50 - Correctly completes both cases, but has a bit of trouble with focus

Task #4

10. 6:30 - Correctly moves the Vect definition

Feedback

11. 7:00 - Aside from the bugs, Subject 2 thinks the data declarations worked well
12. 7:20 - Subject 2 wants it to be easier to customize auto suggestions, so you don’t have to write it out with the keyboard
13. 8:00 - Defining zip was very easy
14. 8:20 - Found the placeholder text helpful

Subject 3

Task #1

1. 0:20 - Misidentifies the handles as meaning “declaration”
2. 0:50 - Identifies the area under the first line as the name and type for the data declaration
3. 1:00 - Identifies the constructors, but thinks the lines separate constructors
4. 1:15 - Correctly identifies the lines
5. 1:30 - Identifies parameters for constructors, but is confused by the empty boxes
6. 1:50 - Identifies the empty boxes as placeholders
7. 2:00 - Identifies the plus button for constructors and definitions

Task #2

8. 2:35 - Subject 3 is shown the Vect definition in regular Idris
9. 2:40 - Subject 3 correctly hits the plus button and chooses data
10. 2:45 - Subject 3 gives it the correct name and type
11. 3:00 - Subject 3 sets the first argument correctly without trouble, but has some difficulty setting the next argument, due to the suggestion box not popping up when the input field gains focus.
12. 3:20 - Hits the plus button to add a new constructor
13. 3:25 - Gives the constructor the correct name
14. 3:30 - Correctly sees that it takes no arguments
15. 3:35 - When setting the type, chooses “Vect n a” from the popup, and then has a bit of touble setting the cursor after the “n”, and changing it to “Z”.
16. 3:55 - Hits plus again
17. 4:00 - Gives it the name “cons”
18. 4:05 - Incorrectly sets the first argument to “Nat”, but correctly sets the second argument. We don’t notice.

Task #3

19. 5:25 - Hits the plus button and chooses “function”
20. 5:30 - Subject 3 is impressed the app knew to call the function “zip”
21. 5:45 - Is not sure how to give the type at first, but realizes it is like regular Idris
22. 6:15 - Realizes the cons case only takes “Nat”
23. 6:20 - Sets the arguments correctly, correcting the “Vect n a” from the popup to “Vect n b”
24. 6:30 - Subject 3 runs into a bug where the popup appears above the previous input field
25. 6:45 - After being reminded of how the type of a tuple is written in Idris, sets the correct return type
26. 7:00 - Starts by hitting the plus button for the first case
27. 7:25 - Quickly case splits and autocompletes through the function

Task #4

28. 13:20 - Drags and drops the Vect definition into the correct place

Feedback

29. On data declarations
    a. 8:05 - Thinks the lines in data definitions are too thick, making them look like separators. More space would also help.
    b. 8:40 - Is confused by the extra input fields.
    c. 9:45 - Thinks he could identify the Nat and Fin declarations with his dependent type knowledge. Does not think novices would be able to.
30. On function declarations:
    a. 9:15 - Individual touch areas for each argument would make it easier to change, so you don't have to struggle with the cursor
    b. 11:40 - Thinks that the function declarations are understandable, overall
    c. 12:10 - Suggest that when you fill out one argument, focus should automatically flow to the next field.
31. On pulling declarations apart to create a new one between
    a. 14:10 - Thinks it would be a useful feature, but not very discoverable

Subject 4

Task #1

1. 0:10 - Looks like proofs, Epigram
2. 0:30 - Identifies data declarations for Nat and Fin

Task #2

3. 1:40 - Is confused, as he tries to write the entire type (Nat -> Type -> Type) in the conclusion
4. 2:30 - Correctly fills it out after being prompted, but is confused by the arrows in the premiss. Does it take a function?
5. 3:30 - Starts filling out the constructors.
6. 4:05 - Wonders if he should write the implicits
7. 4:50 - Correctly fills out constructors

Task #3

8. 6:05 - Correctly fills out type for “zip”
9. 6:10 - Has trouble autocompleteting

Task #4

10. 6:50 - Moves “Vect” without problems

Feedback

11. 7:10 - Overall thinks the data declarations are good
12. 7:40 - Thinks the premisses are confusing with arrows.
13. 8:00 - Notices that the data declarations look different than regular proofs, as there are no explicit names
14. 8:50 - Thought the “zip” flow was okay, but thinks the keyboard should have shortcuts for common programming structures such as pairs “(a, b)”, etc.