

Questions 1-3

Here is our solution:

```
(define (names-under-20 p)           (1) define the top level function
  (local [(define (names-under-20--person p) (2) start the local expression
    (if (< (person-age p) 20)
      (cons (person-name p)
            (names-under-20--lop (person-children p)))
      (names-under-20--lop (person-children p))))
    (define (names-under-20--lop lop)
      (cond [(empty? lop) empty]
            [else
             (append (names-under-20--person (first lop))
                     (names-under-20--lop (rest lop))))]) (3) skip here
    and close ]
    (names-under-20--person p))      (4) write the trampoline
```

Question 1

1/1 point (graded)

Which tests should we keep and rename? (choose all that apply)

☒ (check-expect (names-under-20--person P1) (list "N1"))

☐ (check-expect (names-under-20--lop empty) empty)

☒ (check-expect (names-under-20--person P2) (list "N1"))

☒ (check-expect (names-under-20--person P4) (list "N3" "N1"))



Explanation

Since the function we are interested in is names-under-20--person, which we have renamed names-under-20, we should remove the tests for names-under-20--lop since it is now defined locally.

Submit

i Answers are displayed within the problem

Question 2

1/1 point (graded)

Which of the following changes did we make when we refactored the names-under-20--person and names-under-20--lop functions to use encapsulation.

☐ (a) we changed the behaviour of the program

☒ (b) we changed the structure of the program

☐ Both (a) and (b)


☐ None of the above



Explanation

This refactoring only changes the structure of the program, and we expect the behaviour of the program to remain the same after we encapsulate the functions using local.

Submit

 Answers are displayed within the problem

Question 3

1/1 point (graded)

What are the **advantages** of using local for encapsulation? Choose all that apply.

☒ We can choose whatever function names we want to define locally

☐ We don't have to write base case tests

☒ We can hide functions other parts of the program are not interested in

☐ The program becomes more efficient



Explanation

We have the advantage of choosing whatever function names we want to define locally, and hiding functions other parts of the program are not interested in, because the rest of the program cannot see the functions we define locally.

While we may not be able to write base case tests once the functions are encapsulated, but this is not an advantage, and is in fact often a disadvantage.

Submit

 Answers are displayed within the problem