Write the *type* of each of the following OCaml expressions in the first blank provided, or *ill-typed* if the expression does not type check. Then, after the ⇒ symbol, write the most simplified *value* of the expression, or leave it blank if it’s ill-typed. We have done the first one for you as an example.

(a) let a : _______ int ______ = 3 + 4 ⇒ _______ 7 _______
(b) let b : _______ ________ = a = 8 ⇒ _______ ________
(c) let c : _______ ________ = 5 / 4 ⇒ _______ ________
(d) let d : _______ ________ = "hello " + "world" ⇒ _______ ________
(e) let e : _______ ________ = if a > 0 then "positive" ⇒ _______ ________
(f) let f : _______ ________ = if a > 0 then "positive"
    else "negative" ⇒ _______ ________
(g) let g : _______ ________ = if a > 0 then 42 else 41.5 ⇒ _______ ________
(h) let h : _______ ________ = let q = 3 <> 4 in
    (not q) && (a = 7) ⇒ _______ ________

Locate each binding in the code below. Then determine the scope of each identifier. (Give the range of lines.) If the binding is shadowed by a later one, indicate that as well. The first two are done as an example.

```
let profit_500 : int =  
    let price     = 500 in
    let attendees = 120 in
    let revenue   = price * attendees in
    let cost      = 18000 + 4 * attendees in
    revenue - cost

let attendees (ticket_price : int) : int = 
    (-15 * ticket_price)/10 + 870
let test () : bool = 
    (attendees 500) = 120
    ;; run_test "attendees at $5.00" test

let cost (ticket_price : int) : int = 
    18000 + (attendees ticket_price) * 4
let revenue (ticket_price : int) = 
    (attendees ticket_price) * ticket_price
let profit (ticket_price : int) : int = 
    (revenue ticket_price) - (cost ticket_price)
let test () : bool = 
    (profit 500) = profit_500
    ;; run_test "profit at $5.00" test
```