Scenario: You've decided to get out of this computer science racket and start your own dress-making and suit-making shop. You'll make a variety of different patterns of dresses and suits, each of which will require a certain number of yards of fabric. And there are different fabrics you can use for each project. You're going to have to do some budgeting before you head to the fabric store, because fabric is expensive.

```
type fabric = Linen | Cotton | Wool;;                                                    6

let cost_per_yard_of_fabric f =
    match f with
          Linen -> 15
        | Cotton -> 6
        | Wool -> 18
;;

print_endline ("Linen cost per yard: " ^ string_of_int (cost_per_yard_of_fabric Linen));;
print_endline ("Cotton cost per yard: " ^ string_of_int (cost_per_yard_of_fabric Cotton));;
print_endline ("Wool cost per yard: " ^ string_of_int (cost_per_yard_of_fabric Wool));;
```

```
type item =  Fabric_Sample | Dress of int | Suit of int * int ;;                        7

let yards_required_for_item p =
    match p with
          Fabric_Sample -> 1
        | Dress (yards) -> yards
        | Suit (jacket_yards, pants_yards) -> jacket_yards + pants_yards
;;


let sample = Fabric_Sample;;
let mini_dress = Dress 4;;
let maxi_dress = Dress 8;;
let a_suit = Suit (3, 6);;

print_endline ("Yards for sample: " ^ string_of_int (yards_required_for_item sample));;
print_endline ("Yards for mini dress: " ^ string_of_int (yards_required_for_item mini_dress));;
print_endline ("Yards for maxi dress: " ^ string_of_int (yards_required_for_item maxi_dress));;
print_endline ("Yards for suit: " ^ string_of_int (yards_required_for_item a_suit));;
```

```
type project = { itm: item; fab: fabric } ;;                                             8

let fabric_budget_for_project proj =
    (yards_required_for_item proj.itm) * (cost_per_yard_of_fabric proj.fab)
;;

let summer_suit = { fab=Linen; itm=a_suit };;
let winter_suit = { itm=a_suit; fab=Wool };;

print_endline ("Summer suit: $" ^ string_of_int (fabric_budget_for_project summer_suit));;
print_endline ("Winter suit: $" ^ string_of_int (fabric_budget_for_project winter_suit));;
```

```
let good_idea_for_project proj =                                                         9
    match proj with
          { fab = Wool; itm = Dress yards } -> (if yards < 5
            then "why are you making a summer dress out of wool??"
            else "nice winter dress, sounds good")
        | { fab = Linen; itm = Suit (_, _) } -> "hm, linen suits aren't really in right now"
        | _ -> "eh, not sure"
;;

print_endline ("Good idea?  " ^ good_idea_for_project { fab=Wool; itm=mini_dress });;
print_endline ("Good idea?  " ^ good_idea_for_project { fab=Wool; itm=maxi_dress });;
print_endline ("Good idea?  " ^ good_idea_for_project { fab=Linen; itm=Suit(4, 6) });;
```

Remember how we talked in the first class session about how we'd use : to mean "has type"? Let's go through and add the types for some of the items from the last few activities. What should be the types of the following? If you want to check your types, just try running the program. OCaml will tell you if you got it wrong!

```
type fabric = Linen | Cotton | Wool;;                                                    10
type item =  Fabric_Sample | Dress of int | Suit of int * int ;;
type project = { itm: item; fab: fabric } ;;

let x:_____________________________________ = 164;;
let y:_____________________________________ = 164.0;;
let z:_____________________________________ = "164";;
let linen:_____________________________________ = Linen;;
let sample:_____________________________________ = Fabric_Sample;;
let a_line_dress:_____________________________ = Dress (8);;
let suit:_____________________________________ = Suit (4, 6);;
let summer_dress:_____________________________ = {itm=a_line_dress; fab=linen};;
```

Awesome! What about the types of these functions?

```
type fabric = Linen | Cotton | Wool;;                                                    11
type item =  Fabric_Sample | Dress of int | Suit of int * int ;;
type project = { itm: item; fab: fabric } ;;

let cost_per_yard_of_fabric (f:_____________________________) : _____________________________ =
    match f with
          Linen -> 15
;;

let yards_required_for_item (p:_____________________________) : _____________________________ =
    match p with
          Fabric_Sample -> 1
;;

let fabric_budget_for_project (proj:_____________________________) : _____________________________ =
    (yards_required_for_item proj.itm) * (cost_per_yard_of_fabric proj.fab)
;;

let good_idea_for_project (proj:_____________________________) : _____________________________ =
    match proj with
          { fab = Wool; itm = Dress yards } -> "it's a wool dress"
;;
```

In the snippet below, please highlight everything that's a **type**. Then please highlight everything that's a **constructor**. It's ok if you're not quite sure. Make some educated guesses based on the role each piece of the program is playing.

```
type fabric = Linen | Cotton | Wool;;                                                    12
type item =  Fabric_Sample | Dress of int | Suit of int * int ;;

let linen = Linen;;
let sample:item = Fabric_Sample;;
let a_line_dress = Dress (8);;
let suit:item = Suit (4, 6);;
```

Answer key for 12, with types highlighted in blue and constructors highlighted in orange.

```
type fabric = Linen | Cotton | Wool;;                                                    12
type item =  Fabric_Sample | Dress of int | Suit of int * int ;;

let linen = Linen;;
let sample:item = Fabric_Sample;;
let a_line_dress = Dress (8);;
let suit:item = Suit (4, 6);;
```