Hint

Start with the the following code. This code differs from the solution that you need, on two main grounds.

First, the following code defines a pattern-matcher generator that works only on DNA patterns and requires acceptors to return fragment options, which means its make_matcher is of type pattern -> nucleotide list -> (nucleotide list -> 'a option) -> 'a option. Your make_matcher should have the more-general type 'a pattern -> 'a list -> ('a list -> 'b option) -> 'b option.

Second, the following code does not support the Eager pattern; you need to add support for that.

```
(* DNA fragment analyzer.  *)

type nucleotide = A | C | G | T
type fragment = nucleotide list
type acceptor = fragment -> fragment option
type matcher = fragment -> acceptor -> fragment option

type pattern =
  | Frag of fragment
  | List of pattern list
  | Or of pattern list
  | Junk of int
  | Closure of pattern

let match_empty frag accept = accept frag

let match_nothing frag accept = None

let rec match_junk k frag accept =
  match accept frag with
    | None ->
        (if k = 0
        then None
        else match frag with
              | [] -> None
              | _::tail -> match_junk (k - 1) tail accept)
    | ok -> ok

let rec match_star matcher frag accept =
  match accept frag with
    | None ->
       matcher frag
              (fun frag1 ->
                 if frag == frag1
                 then None
                 else match_star matcher frag1 accept)
    | ok -> ok

let match_nucleotide nt frag accept =
```

```
  match frag with
    | [] -> None
    | n::tail -> if n == nt then accept tail else None

let append_matchers matcher1 matcher2 frag accept =
  matcher1 frag (fun frag1 -> matcher2 frag1 accept)

let make_appended_matchers make_a_matcher ls =
  let rec mams = function
    | [] -> match_empty
    | head::tail -> append_matchers (make_a_matcher head) (mams tail)
  in mams ls

let rec make_or_matcher make_a_matcher = function
  | [] -> match_nothing
  | head::tail ->
      let head_matcher = make_a_matcher head
      and tail_matcher = make_or_matcher make_a_matcher tail
      in fun frag accept ->
          let ormatch = head_matcher frag accept
          in match ormatch with
              | None -> tail_matcher frag accept
              | _ -> ormatch

let rec make_matcher = function
  | Frag frag -> make_appended_matchers match_nucleotide frag
  | List pats -> make_appended_matchers make_matcher pats
  | Or pats -> make_or_matcher make_matcher pats
  | Junk k -> match_junk k
  | Closure pat -> match_star (make_matcher pat)
```