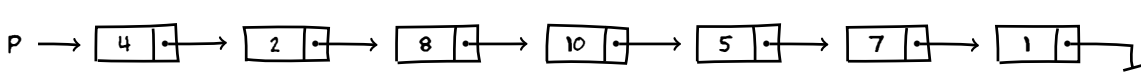
Separando pares e ímpares

Imagine que o ponteiro p aponta para uma lista encadeada

A tarefa é

→ *Colocar todos os elementos pares no início da lista*
e todos os elementos ímpares no fim da lista

No exemplo acima, isso nos daria



Bom, a ideia é fazer as coisas na ida dessa vez.

Quer dizer, a função recursiva vai ter dois parâmetros pp e pi, onde nós vamos colocar os elementos pares e ímpares.

Daí, quando a chamada que vê os `Nulo` recebe essas duas listas, ela faz a sua concatenação e retorna o resultado.

Para facilitar a concatenação, nós vamos utilizar um terceiro ponteiro up que aponta para o último elemento da listas dos pares.

A coisa fica assim

```
func SPI-Rec (q,pp,up,pi)                          // Separa Pares e Ímpares

      Se ( q == Nulo )                             // lá no fim
            Se (pp == Nulo)   Retorna (pi)
            Senão
                  up->prox = pi;   Retorna (pp)    // concatena pp e pi

      Sse (q->num é par)
            m = q->prox                            // segura o próximo elemento
            q->prox = pp;   pp = q                 // e adiciona elemento à lista pp
            Se (up == Nulo)   up = q

      Senão
            m = q->prox                            // segura o próximo elemento
            q->prox = pi;   pi = q                 // e adiciona elemento à lista pi

      resp = SPI-Rec (m,pp,up,pi)
      Retorna (resp)
```

Legal.

Mas, essa não é a única maneira de fazer as coisas.

Quer dizer, a gente pode ser um pouquinho mais esperto.

A ideia é já ir construindo a lista resultado na ida

- inserindo os elementos pares no início
- inserindo os elementos ímpares no fim

Para isso, a gente utiliza dois parâmetros ppi e upi.

A coisa fica assim

```
func SPI2-Rec (q,ppi,upi)                          // Separa Pares e Ímpares

      Se (q == Nulo)   Retorna (ppi)               // lá no fim

      Sse (q->num é par)
            m = q->prox                            // segura o próximo elemento
            q->prox = ppi;   ppi = q               // e adiciona elemento no inicio
            Se (upi == Nulo)   upi = q

      Senão
            m = q->prox                            // segura o próximo elemento
            q->prox = Nulo
            Se (upi == Nulo)
                  ppi = q;   upi = q               // primeiro elemento (no fim)
            Senão
                  upi->prox = q;   upi = q         // próximos elementos (no fim)

      resp = SPI2-Rec (m,ppi,upi)
      Retorna (resp)
```

E a função `main()` fica assim

```
func main ()

      p = SPI2-Rec (p,Nulo,Nulo)
```

◇