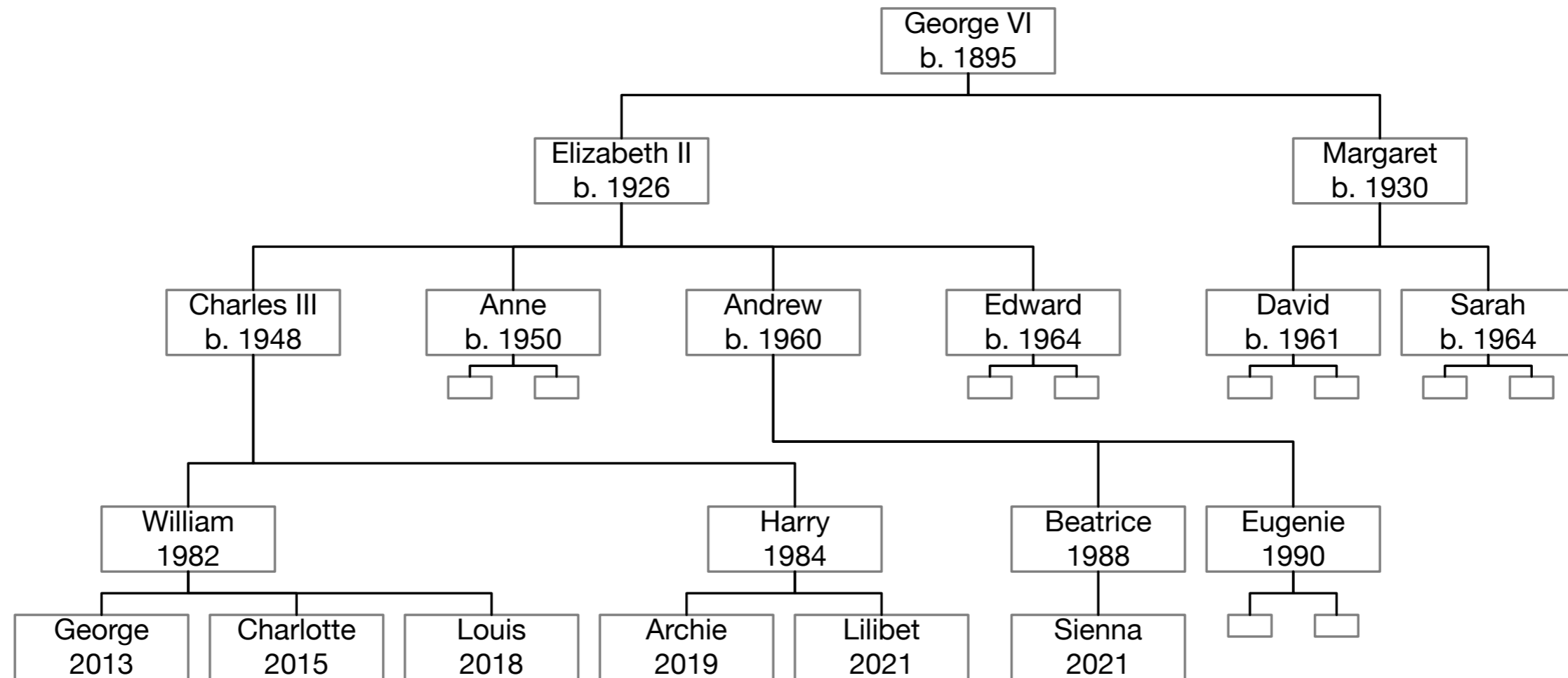


CS135 T08

General Trees, Mutual Recursion, and Local

Descendants

- Develop a data definition to store descendants of King George VI.
- Write templates for functions to process the descendants.



Descendants Data Definition

```
(define-struct person (name birth children))  
;; A Person is a (make-person Str Nat (listof Person))  
;; Requires: names are unique  
  
;; person-template: Person -> Any  
(define (person-template p)  
  (... (person-name p)  
        (person-birth p)  
        (listof-person-template (person-children p))))  
  
;; listof-person-template: (listof Person) -> Any  
(define (listof-person-template lop)  
  (cond [(empty? lop) ...]  
        [else (... (person-template (first lop))  
                    (listof-person-template (rest lop)))]))
```

Birthdate

- Write `(birthdate name p)`, which finds the descendant of `p` with the given `name` and produces their birthdate or `false` if not found.

```
(define george
  (make-person "George VI" 1895
    (list (make-person "Elizabeth II" 1926
      (list (make-person "Charles III" 1948
        (list (make-person "William" 1982
          (list (make-person "George" 2013 (list))
                (make-person "Charlotte" 2015 (list))
                (make-person "Louis" 2018 (list))))
            (make-person "Harry" 1984
              (list (make-person "Archie" 2019 (list))
                    (make-person "Lilibet" 2021 (list)))))))
          (make-person "Andrew" 1960
            (list (make-person "Beatrice" 1988
              (list (make-person "Sienna" 2021 (list))))
                (make-person "Eugenie" 1990 (list))))
            (make-person "Edward" 1964 (list))
            (make-person "Anne" 1950 (list))
            ))
      (make-person "Margaret" 1930
        (list (make-person "David" 1961 (list))
              (make-person "Sarah" 1964 (list))))
    )))
```

Birthdate tests

```
(check-expect (birthdate "George VI" george) 1895)  
(check-expect (birthdate "Anne" george) 1950)  
(check-expect (birthdate "Sarah" george) 1964)  
(check-expect (birthdate "Justin" george) false)
```

```
;; (birthdate name p) finds the birthdate of the named person.  
;; birthdate: Str Person -> (anyof false Nat)  
(define (birthdate name p) ...)
```

Born After

```
;; (born-after year p) produces a list of all the names in p and  
;; p's descendants that were born after the specified year.
```

```
;; born-after: Nat Person -> (listof Str)  
(define (born-after year p) ...)
```

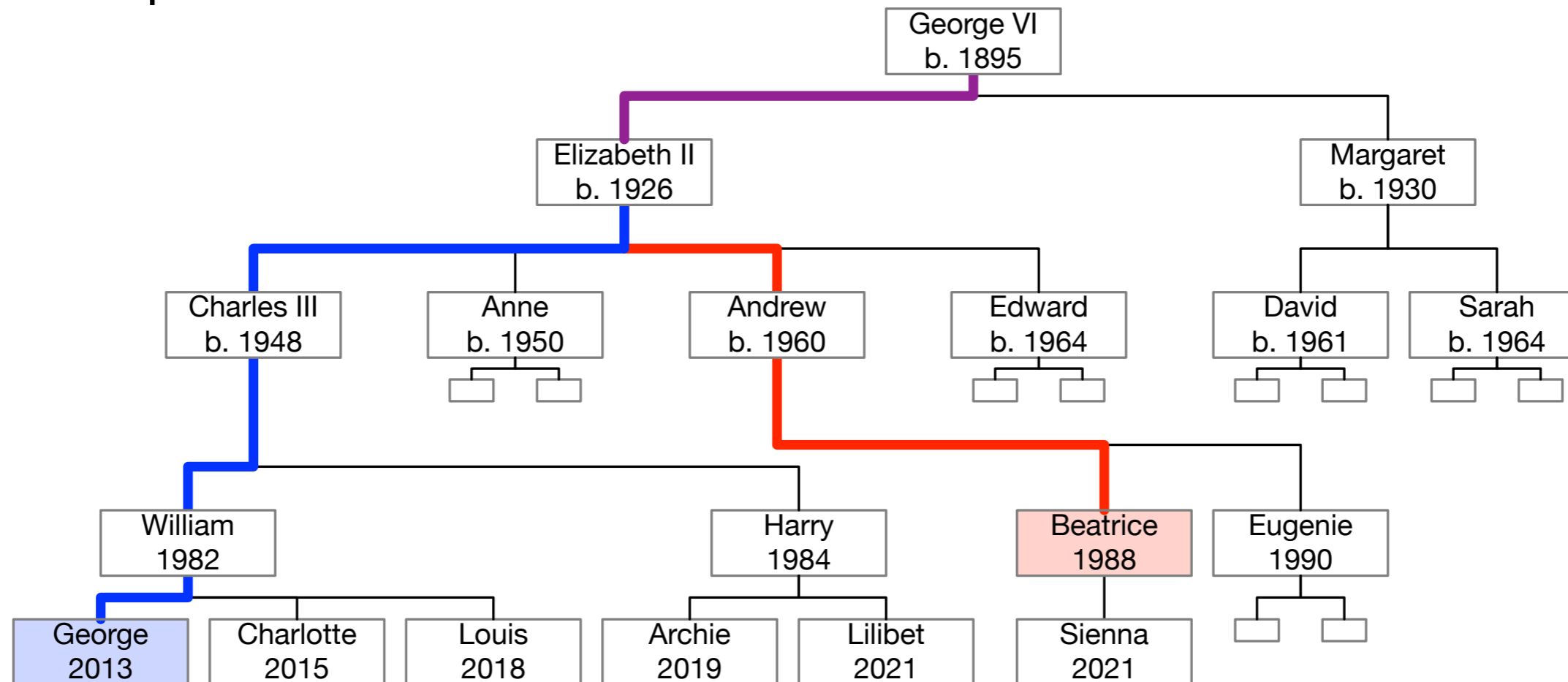
```
(check-expect (born-after-v1 2023 george) empty)  
(check-expect (born-after-v1 2018 george)  
              (list "Sienna" "Lilibet" "Archie" "Louis"))
```

We'll solve this problem two different ways:

- with **append**
- with an accumulator

Nearest common ancestor

(`nearest-common-ancestor name1 name2 p`) finds the nearest common ancestor, within `p` and `p`'s descendants, of `name1` and `name2`. We assume the names are unique within the descendants tree.



Nearest common ancestor – strategy

- Find the “path” – the list of people – that lead from p to $name1$ (but not including $name1$).
- Find the “path” – the list of people – that lead from p to $name2$ (but not ...).
- Produce the last item on the common prefix.

Considerations

- The paths might be different lengths.
- What if one or both of the names are not found?
- Who is the nearest common ancestor of George VI?
- Who is the nearest common ancestor of “Beatrice” and “Beatrice”?

Nearest common ancestor – design recipe

```
;; (nearest name1 name2 p) finds the name of the nearest common ancestor  
;; of name1 and name2 within the descendants of p.
```

```
;; nearest: Str Str Person -> (anyof Str false)  
(define (nearest name1 name2 p) ...)
```

```
(check-expect (nearest "Elizabeth II" "Margaret" george) "George VI")  
(check-expect (nearest "Charles III" "Edward" george) "Elizabeth II")  
(check-expect (nearest "George" "Charlotte" george) "William")  
(check-expect (nearest "Archie" "Bilbo Baggins" george) false)  
(check-expect (nearest "Beatrice" "Beatrice" george) "Andrew")  
(check-expect (nearest "George VI" "Beatrice" george) false)
```