



# Discussion 6

## Discussion 6 - Thursday, July 24th

### Reminders

- Project 5 is due on **Friday, July 25th @11:59 PM**
  - This project **does not** accept late submissions! Please get it done!
- GFA deadline is **Friday, July 25th**
- Your **Final Exam** is on **Friday, July 25th @9:30 - 11:30 AM**
- Refer to the following for more information:
  - [Final Exam Logistics](https://piazza.com/class/maxx5fjxl92bq/post/214)  [\(https://piazza.com/class/maxx5fjxl92bq/post/214\)](https://piazza.com/class/maxx5fjxl92bq/post/214)
  - If the timing causes a conflict, please **email Dr. Mamat**
- (Optional) Please fill out **course evals** and the [TA feedback form](https://docs.google.com/forms/d/e/1FAIpQLSdl4pcDMf5EZaQJ46EaT1OJ4VsyAD39Ont7sa9N_bt2JTphLQ/viewform?usp=header)  [\(https://docs.google.com/forms/d/e/1FAIpQLSdl4pcDMf5EZaQJ46EaT1OJ4VsyAD39Ont7sa9N\\_bt2JTphLQ/viewform?usp=header\)](https://docs.google.com/forms/d/e/1FAIpQLSdl4pcDMf5EZaQJ46EaT1OJ4VsyAD39Ont7sa9N_bt2JTphLQ/viewform?usp=header)!



### Final Exam Exercises! :D

#### 1. Regular Expressions:

##### Summer 2023 Exam 1:

(a) Which of the following strings are an exact match of the following Regular Expression? Mark all that apply.

[5 pts]

$^[A-Z][a-z0-9]+: ([0-9]{3} | [CS330]+)$$

- ☐ (A) Major: CS  
 ☐ (B) Age: 25  
 ☐ (C) Class: CS330  
 ☐ (D) Finitial: C  
 ☐ (E) None

(b) Write a regular expression that accepts phone numbers of all the following formats and rejects everything else. You may assume that any X can be any digit.

[5 pts]

XXX-XXX-XXXX    XXX-XXXXXXX    XXXXXXXXXX    (XXX)-XXX-XXXX    (XXX)-XXXXXXX    (XXX)XXXXXXX

(c) Write a regular expression that would accept all strings of odd length and have at least 1 lowercase vowel (a,e,i,o,u) and reject anything else

[5 pts]

#### ▼ Solution

(a) Which of the following strings are an exact match of the following Regular Expression? Mark all that apply.

$^{[A-Z][a-z0-9]+:([0-9]\{3\}|[CS330]+)}^{\$}$

☒ Major: CS

☐ Age: 25

☒ Class: CS330

☒ Final: C

☐ None

(b) Write a regular expression that accepts phone numbers of all the following formats and rejects everything else. You may assume that any X can be any digit.

XXX-XXX-XXXX

XXX-XXXXXXX

XXXXXXXXXX

(XXX)-XXX-XXXX

(XXX)-XXXXXXX

(XXX)XXXXXXX

$((\backslash d\{3\})|\backslash d\{3\})(-\backslash d\{3\}-?\backslash d\{4\})|\backslash d\{7\})$

(c) Write a regular expression that would accept all strings of odd length and have at least 1 lowercase vowel (a,e,i,o,u) and reject anything else

$(..)^*[aeiou](..)^*|(..)^*[aeiou](..)^*$

Note, the answer key forgets to escape the parentheses ^

Write me a regular expression that only accepts integers between in the range  $\$[0, 314]\$$  inclusive, without padding zeroes in front.

#### ▼ Solution

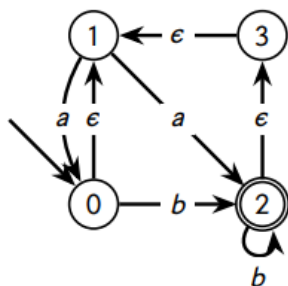
$([0-9]|([1-9][0-9]|([12][0-9]\{2\}|30[0-9]|31[0-4]))$

## 2. Finite State Machines

Spring 2024 Final Exam:

(a) Convert the below NFA to a DFA.  
Draw a **box** around your final answer.

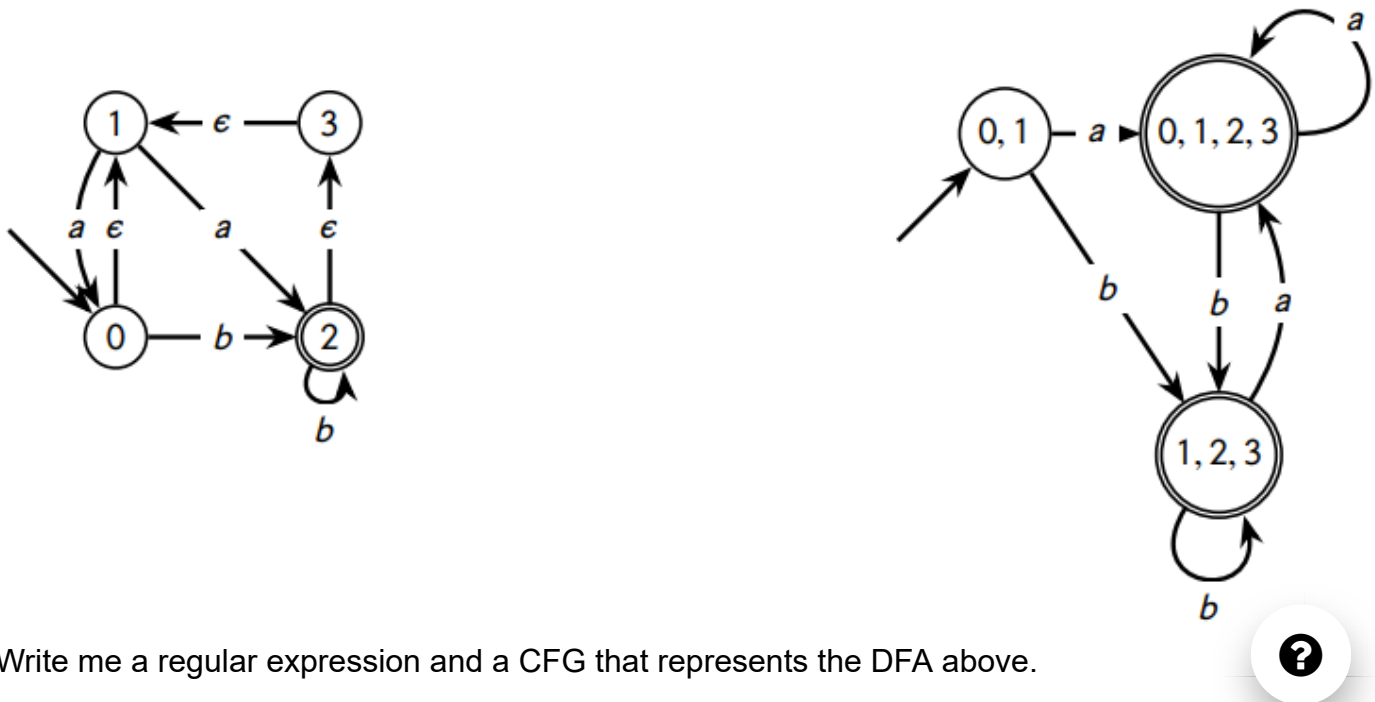
Scratch Space:



#### ▼ Solution

Convert the below NFA to a DFA. Draw a **box** around your final answer.

Version A:



Write me a regular expression and a CFG that represents the DFA above.

#### ▼ Solution

RegEx:  $[ab]^+$   
 CFG:  $S \rightarrow aT \mid bT$   
 $T \rightarrow aT \mid bT \mid \epsilon$

### 3. OCaml Typing and Evaluation

For the following expressions, give the type of the variable 'r'. If there is a type error, put "ERROR".

1. `let r = fun a b c d -> (a b)@(c::d)`
2. `let r x y = fun z -> fold_left y [1] (map x z)`

#### ▼ Solution

1. `('a -> 'b list) -> 'a -> 'b -> 'b list -> 'b list`
2. `('a -> 'b) -> (int list -> 'b -> int list) -> 'a list -> int list`

Now evaluate the following expressions. If there is a compilation error, put "ERROR"

1. `let r = fun a b c d -> (a b)@(c::d) in r (fun x -> [x]) 3 1 [4]`
2. `let r x y = fun z -> fold_left y [4] (map x z) in r (( * ) (-1)) (fun a x -> x::a) [-1; -3]`

#### ▼ Solution

1. `[3; 1; 4]`
2. `[3; 1; 4]`

## 4. Interpreters

From Spring 2024 Final Exam

Grammar:

$$\begin{aligned} M &\rightarrow ME + \mid ME - \mid E \\ E &\rightarrow OE / \mid OE * \mid O \\ O &\rightarrow WO > \mid WO < \mid W \\ W &\rightarrow n \mid b \end{aligned}$$

**Note:**  $n \in \mathbb{Z}$ ,  $b \in \{true, false\}$

The opsem for this grammar is given below:

OpSem:

$\overline{A; n \Rightarrow n}$			$\overline{A; b \Rightarrow b}$		
$A; e_1 \Rightarrow v_1$	$A; e_2 \Rightarrow v_2$	$v_3 = v_1 + v_2$	$A; e_1 \Rightarrow v_1$	$A; e_2 \Rightarrow v_2$	$v_3 = v_1 - v_2$
$A; e_1 e_2 + \Rightarrow v_3$			$A; e_1 e_2 - \Rightarrow v_3$		
$A; e_1 \Rightarrow v_1$	$A; e_2 \Rightarrow v_2$	$v_3 = v_1 / v_2$	$A; e_1 \Rightarrow v_1$	$A; e_2 \Rightarrow v_2$	$v_3 = ?$
$A; e_1 e_2 / \Rightarrow v_3$			$A; e_1 e_2 * \Rightarrow v_3$		
$A; e_1 \Rightarrow v_1$	$A; e_2 \Rightarrow v_2$	$v_3 = v_1 > v_2$	$A; e_1 \Rightarrow v_1$	$A; e_2 \Rightarrow v_2$	$v_3 = v_1 < v_2$
$A; e_1 e_2 > \Rightarrow v_3$			$A; e_1 e_2 < \Rightarrow v_3$		

Consider the above CFG and OpSem. Assume OCaml's typing system. Determine whether they would fail in the Lexer, Parser, Evaluator, or are Valid expressions.

- 1) 3 10 ?
- 2) true false not
- 3) + 1 / 3 4
- 4) \* 1 2 \* 7 + 6
- 5) 3 true <
- 6) 3 14 +

### ▼ Solution

- 1) Fails Lexer
- 2) Fails Lexer
- 3) Fails Parser
- 4) Fails Parser
- 5) Fails Evaluator
- 6) Valid

## 5. OpSem/Typechecking

From Fall 2024 Discussion 8

Using the rules given below, show:  $A; \text{let } x = 3 \text{ in let } x = x + 6 \text{ in } x \Rightarrow 9$

$$\frac{}{A; n \Rightarrow n} \quad \frac{A(x) = v}{A; x \Rightarrow v} \quad \frac{A; e_1 \Rightarrow v_1 \quad A, x: v_1; e_2 \Rightarrow v_2}{A; \text{let } x = e_1 \text{ in } e_2 \Rightarrow v_2} \quad \frac{A; e_1 \Rightarrow n_1 \quad A; e_2 \Rightarrow n_2 \quad n_3 \text{ is } n_1 + n_2}{A; e_1 + e_2 \Rightarrow n_3}$$

Using the rules given below, show `let x = 3 in let x = x + 6 in x` is well typed:

$$\frac{}{G \vdash x : G(x)} \quad \frac{}{G \vdash \text{true} : \text{bool}} \quad \frac{}{G \vdash \text{false} : \text{bool}} \quad \frac{}{G \vdash n : \text{int}}$$

$$\frac{G \vdash e_1 : t_1 \quad G, x : t_1 \vdash e_2 : t_2}{G \vdash \text{let } x = e_1 \text{ in } e_2 : t_2} \quad \frac{G \vdash e_1 : \text{int} \quad G \vdash e_2 : \text{int} \quad + = (\text{int}, \text{int}, \text{int})}{G \vdash e_1 + e_2 : \text{int}}$$

### ▼ Solution

Show  $A; \text{let } x=3 \text{ in let } x=x+6 \text{ in } x \Rightarrow 9$  :

$$\frac{\frac{\frac{A, x:3(x)=3}{A, x:3; x \Rightarrow 3} \quad \frac{A, x:3; 6 \Rightarrow 6 \quad 9 \text{ is } 3+6}{A, x:3, x:9(x)=9}}{A, x:3; x+6 \Rightarrow 9} \quad \frac{}{A, x:3, x:9; x \Rightarrow 9}}{A, 3 \Rightarrow 9 \quad A, x:3; \text{let } x = x+6 \text{ in } x \Rightarrow 9} \quad \frac{}{A; \text{let } x=3 \text{ in let } x=x+6 \text{ in } x \Rightarrow 9}$$

Show `let x=3 in let x=x+6 in x` is well-typed:

$$\frac{\frac{G, x:\text{int} \vdash x : \text{int} \quad G, x:\text{int} \vdash 6 : \text{int} \quad + = (\text{int}, \text{int}, \text{int})}{G, x:\text{int} \vdash x+6 : \text{int}} \quad \frac{}{G, x:\text{int}, x:\text{int} \vdash x : \text{int}}}{G \vdash 3 : \text{int} \quad G, x:\text{int} \vdash \text{let } x = x+6 \text{ in } x : \text{int}} \quad \frac{}{G \vdash \text{let } x=3 \text{ in let } x=x+6 \text{ in } x : \text{int}}$$

## 6. Lambda Calculus

From Spring 2023 Final Exam

Reduce the following lambda expression. Show every step:

`((λx . (λy . y x)) y) (λx . x b)`

### ▼ Solution

$((\lambda x . (\lambda y . y x)) y) (\lambda x . x b)$   
 $((\lambda x . (\lambda a . a x)) y) (\lambda x . x b)$  - Note, alpha conversion is necessary here  
 $(\lambda a . a y) (\lambda x . x b)$   
 $(\lambda x . x b) y$   
 $y b$

Determine the free variables in the following lambda expressions:

$$(\lambda x . (\lambda x . x x) x) x (\lambda y . y f) a$$

$$(\lambda x . (\lambda x . a x) x) a (\lambda y . y y) x$$

### ▼ Solution

$$\begin{array}{ccccccc} (\lambda x . (\lambda x . x x) x) & x & (\lambda y . y f) & a \\ & \wedge & \wedge & \wedge \\ (\lambda x . (\lambda x . a x) x) & a & (\lambda y . y y) & x \\ & \wedge & \wedge & \wedge \end{array}$$

Which of the following expressions are alpha equivalent to the following lambda expression:

$$(\lambda x . (\lambda x . x x) x) x (\lambda y . y f) a$$

- a)  $(\lambda x . (\lambda b . b b) b) x (\lambda w . w f) a$
- b)  $(\lambda w . (\lambda b . b b) w) w (\lambda c . c f) a$
- c)  $(\lambda y . (\lambda d . d d) y) x (a f)$
- d)  $(\lambda w . (\lambda z . z z) w) x (\lambda y . y f) a$
- e)  $(\lambda x . (\lambda x . x x) x) x (\lambda x . x f) a$

### ▼ Solution

d) and e) are the only alpha equivalent expressions



## 7. Rust Ownership

From Spring 2024 Final Exam

Determine if the following code snippets will compile. If they don't, explain why.

1:

```
1 fn main(){
2   let x = 4;
3   let y = x;
4   println!("{x},{y}");
5 }
```

2.

```
1 fn main(){
2   let x = String::from("Hello");
3   let y = &mut x;
4   println!("{y}");
5 }
```

3.

```
1 fn main(){
2   let mut x = String::from("Hello");
3   let y = &mut x;
4   x.push_str(" world");
5   println!("{x},{y}");
6 }
```

4.

```

1 fn main(){
2   let mut x = String::from("Hello");
3   let y = &mut x;
4   y.push_str(" world");
5   println!("{x},{y}");
6 }

```

5.

```

1 fn main(){
2   let mut x = String::from("Hello");
3   let y = &mut x;
4   y.push_str(" world");
5   println!("{y}");
6   println!("{x}");
7 }

```

6.

```

1 fn function<'a>(s1:&'a String, s2:&'a String, f:bool)->usize{
2   if f {s1.len()} else{s2.len()}
3 }
4 fn main(){
5   let a = String::from("hello");
6   let b = a.clone();
7   let c = function(a,b,true);
8   println!("{a} has length {c}");
9 }

```



7.

```

1 fn function<'a>(s1:&'a String, s2:&'a String, f:bool)->usize{
2   if f {s1.len()} else{s2.len()}
3 }
4 fn main(){
5   let a = String::from("hello");
6   let b = a.clone();
7   let c = function(&b,&a,true);
8   println!("{a} has length {c}");
9 }

```

8.

```

struct Pokemon{
  name:String, hp:usize
}
fn main(){
  let mut x = Pokemon{name:String::from("Pikachu"), hp: 30};
  let z = x.hp;
  println!("{}", {z}, x.name)
}

```

9.

```

struct Pokemon{
  name:String, hp:usize
}

```

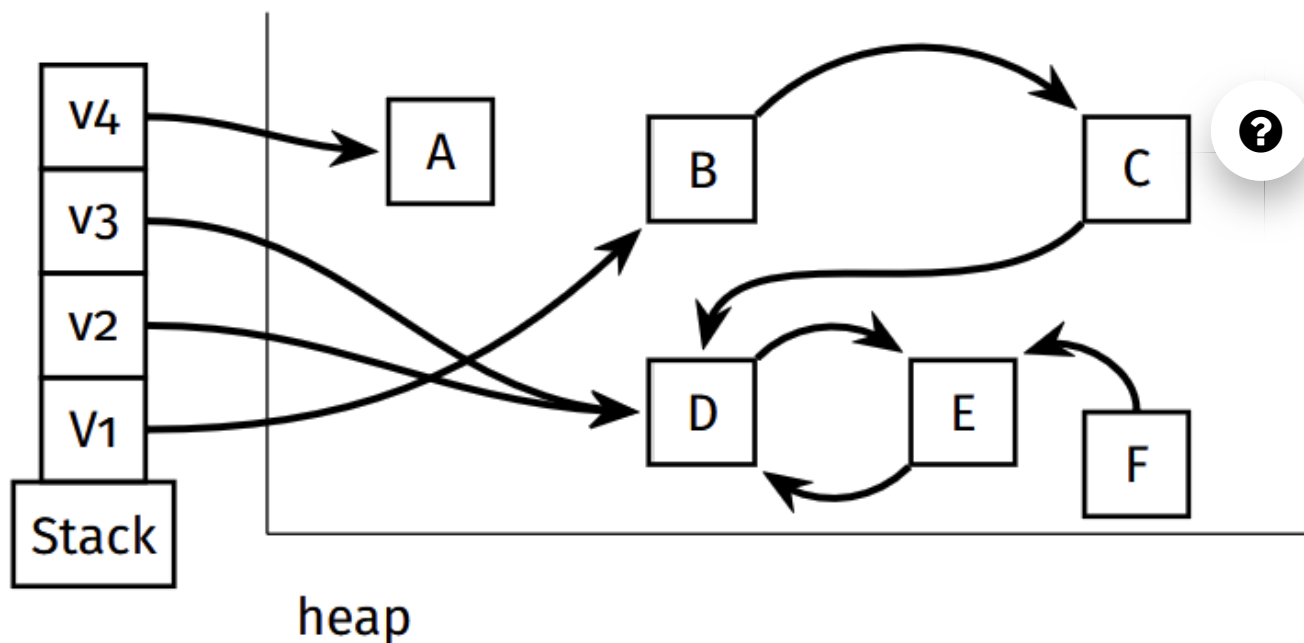
```
fn main(){
  let mut x = Pokemon{name:String::from("Pikachu"), hp: 30};
  let z = x.name;
  println!("{}", {z}, x.name)
}
```

### ▼ Solution

1. Compiles
2. FAILS to compile - Cannot borrow mut from immutable value
3. FAILS to compile - Two mut borrows exist and used at the same time
4. FAILS to compile - println! tries to borrow x as immutable while y borrows x as mutable
5. Compiles - lifetime of y ends before x is used
6. FAILS to compile - Arguments are not references
7. Compiles
8. Compiles
9. FAILS to compile - one mut and one immut borrow to the name, need to use .clone()

## 8. Garbage Collection

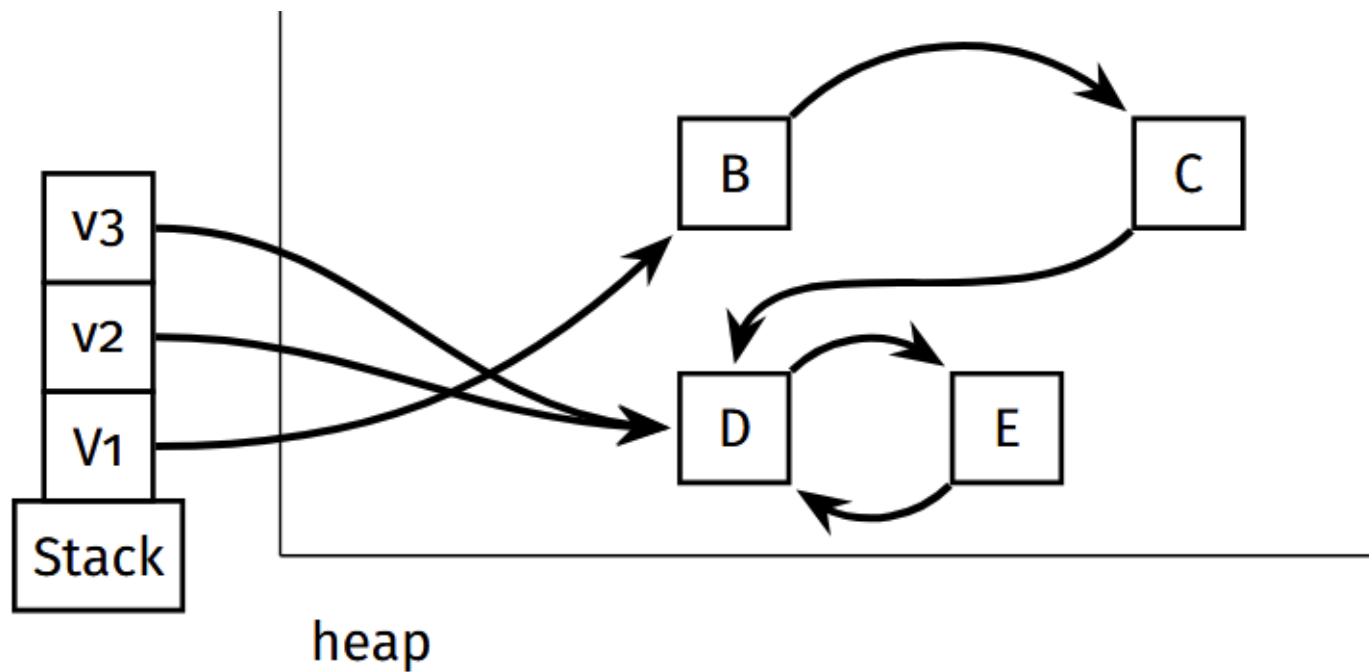
From Fall 2024 Final Exam



Given the above diagram, draw the result of calling the Mark and Sweep garbage collection technique after popping off x4.

### ▼ Solution





### True/False

1. The program must stop while reference counting is occurring
2. Mark and Sweep can handle cycles
3. Not freeing memory that isn't in use will not have any downsides



### ▼ Solution!

1. False - Mark and Sweep and Stop and Copy both stop the program, reference counting doesn't
2. True - Mark and Sweep does fine with cycles, reference counting doesn't
3. False - If your garbage collector doesn't free memory that should be freed, your program will run out of memory faster and be less efficient

### 9. OCaml/Rust Coding

*I would strongly urge you to try these problems on your own!*

OCaml: Given an int list list matrix, return the sum of the diagonal elements of the matrix. Assume the matrix we give you is a square matrix  $(m \times m)$ . Do not use any other List module functions other than map, fold\_left, or fold\_right.

```

ex.
[[1; 2; 3];
 [4; 5; 6]; => 15
 [7; 8; 9]]

let rec sum_diag matrix = ...
  
```

### ▼ Solution

```

Solution 1:
let rec sum_diag matrix =
  let _, sum =
  
```

```

let helper_1 (row_index, sum) row =
  let _, element =
    let helper_2 (col_index, value) elt =
      if row_index = col_index then (col_index+1, elt) else (col_index+1, value) in
      fold_left helper_2 (0, -1) row in
    (row_index + 1, element + sum) in
  fold_left helper_1 (0, 0) matrix in
sum

```

Solution 2:

```

let rec helper lst target curr = match lst with
| [] -> failwith "element not found"
| x::xs -> if curr = target then x else (helper xs target (curr + 1));;

(* our accumulator is a tuple with the sum as the first value, and the index as the second value *)
(* we use a helper function that gets us the value in the list at the index 'target' *)
(* small note, fst gets you the first value in a tuple and snd gets you the second value *)
let rec sum_diag matrix = fst (List.fold_left (fun acc x -> let value = (helper x (snd acc) 0)
in (fst acc + value, snd acc + 1)) (0,0) matrix)

```

Rust: Given a `Vec<Vec>` matrix, return the sum of the diagonal elements of the matrix. Assume the matrix we give you is a square matrix  $(m \times m)$ .

```

ex.
[[1; 2; 3];
 [4; 5; 6]; => 15
 [7; 8; 9]]

fn sum_diag(matrix: Vec<Vec<u32>>) -> u32 { ...

```



### ▼ Solution

```

fn sum_diag(matrix: Vec<Vec<u32>>) -> u32 {
  let mut sum = 0;
  for i in 0..matrix.len() {
    sum += matrix[i][i];
  }
  sum
}

```

## 10. Project-Based Questions

### Try this coding question on your own!

Consider the following (modified) NFA variant type from project 3:

```

type nfa = {
  q0 : int;
  fs : int list;
  delta : (int * string * int) list
}

```

Write a function `epsilon_paths` that, given an `nfa` and a state in the `nfa`, returns a list of all the paths that can be taken using one or more epsilon transitions (in any order). You may also use the `move` and `e_closure` functions from project 3 (shown below).





Assumption: There are no infinite epsilon loops in the NFA.

- ▶ Example
- ▶ **Example Solution**

Good luck, and have a spectacular rest of summer!!

*TA's, this is your opportunity to say something cheesy like how much u appreciate everyone :)*

## Resources

- [All Lecture Quiz Solutions](https://piazza.com/class/m6687mvdhig1iw/post/101)  [\\_ \(https://piazza.com/class/m6687mvdhig1iw/post/101\)](https://piazza.com/class/m6687mvdhig1iw/post/101)
- [Course Resources \(Including Past Final Exams\)](https://bakalian.cs.umd.edu/330/resources)  [\\_ \(https://bakalian.cs.umd.edu/330/resources\)](https://bakalian.cs.umd.edu/330/resources)
- Extra [Exam 1](https://piazza.com/class/m6687mvdhig1iw/post/789)  [\\_ \(https://piazza.com/class/m6687mvdhig1iw/post/789\)](https://piazza.com/class/m6687mvdhig1iw/post/789) and [Exam 2](https://piazza.com/class/m6687mvdhig1iw/post/1770)  [\\_ \(https://piazza.com/class/m6687mvdhig1iw/post/1770\)](https://piazza.com/class/m6687mvdhig1iw/post/1770) **Practice Problems**

