# CSC 151 Spring 2020
# Topic: Vectors

**April 3, 2020**

**Day 26**

# Announcements

## CS Table:

- Virtually on WebEx, Tuesdays at 12:00pm Central Time. If you're looking for community/socialization, join us! Email sent through CS student mailing list.

## Mentor Sessions:

- Virtually. See announcement on Piazza. If the times don't work for you please let me or a mentor know!!

## Assignment 6:

- Assignment 6 will go live today (Friday). You can get help from everyone, just make sure you cite them. And remember what you submit needs to be your own work.

# Vectors

Think about the answers to the following questions.

**What is a vector?**

**What operations does a vector support?**

# Vectors

**What is a vector?**

A data type that holds a fixed number of values. It allows us to change the contents without making a *new* vector.

**What operations does a vector support?**

make-vector, vector-length, list->vector, vector-fill!, vector-ref, vector-set!, vector? vector->list

# Vectors vs. Lists

Try to list out the answers to these questions. (You can use the answers from the previous question to remember the operations)

1. **Which list operations are efficient?**
2. **Which list operations are inefficient?**
3. **Which vector operations are efficient?**
4. **Which vector operations are inefficient?**

# Vectors vs. Lists

| | Efficient | Inefficient |
|---|---|---|
| **Lists** | car, take, drop, cons, append | list-ref, length (Both have to traverse the list.) |
| **Vectors** | vector-ref, vector-length, changing the value in the middle or end of a vector | Adding values to the front of a vector. Appending vectors. |

# Vectors vs. Lists

Do you notice any patterns in the table from the previous page?

# Recursion with Vectors

Without looking at the reading, try to fill in the blanks:

```
(define number-vector-increment!
  (lambda (vec)
    (let ([len ( …  )]
          (let kernel! ([pos … ])
            (when ( …  )
              (… vec pos (… ))
              (… (+ 1 pos)))))))
```

# Recursion with Vectors

- The recursion on the previous slide used "when" (rather than cond or if)
- when is used fairly frequently when dealing with vectors
- Remember that it allows for multiple statements to be executed when the guard passes, which is useful when we want to both make a change to a vector and also make our recursive call. It also has the benefit of not returning anything if the guard doesn't pass, which lets us have a procedure that doesn't return anything. This way of programming is not a very 'Scheme-y' way, which is why we save it for later in the semester.

# Recursion with Vectors

- This brings up the question though: when you are writing a recursive vector procedure, when should you use `when` and when should you use `if`?
- `when` is used by procedures that make changes to the vector and then don't return anything.
- However, some of our procedures on vectors don't make changes to the vector, like number-vector-largest, and instead always return something, and therefore use `if`.
- So if you are working on a procedure and run into a situation where you don't know what should go in the third part of the `if`, you might need to use `when` instead.



# Lab Time

Yay! Friday!

P.S. this picture was not taken recently. Stay home!

# Upcoming Work

Please submit all problems before the extras by Monday 4/6 at 10:30 pm Grinnell time. Include 6P Documentation for problems 4 and 6.

**To:** csc-151-02-grader@grinnell.edu
**Subject:** CSC 151-02 Lab 4/3

---

Exam 2 Revisions (Optional) Due April 10

Assignment 6 posted today.