Practice Problem Workshop 12:
Part I - Individual Practice Questions

INFO 2310, Spring 2026

Student ID: Solution **NetID:**

Overview

Retrieval practice is the act of actively recalling information from memory, like quizzing oneself.

This workshop is designed to help you prepare and study for upcoming exams by engaging in retrieval practice.

Part I: Individual Practice Questions

Credit: 10 points. (completion)

1. Open this booklet and try your best to answer the practice problems **individually**.

   If you need to refer to your notes, you can. But you should try to answer the questions on your own first, without looking at your notes. (This will help you prepare for the upcoming exam.)

2. Once a majority of the class has answered the practice questions on their own, **form groups of 3-4 people and go over the questions together**.

   Discuss the questions and share your answers with the group.

   You should update your answers in this booklet if necessary.

   Discussing the answers will help you practice retrieving the information from memory and will also help you learn from your peers.

Submit this booklet for completion credit.

Practice Problem 1

Modify the code below to submit an HTTP request to the `POST /api/events/EVENT_ID/like` endpoint when the like button is clicked.

While the request is being sent to the server, the like button should be disabled to prevent multiple clicks. Use `<button>`'s `disabled` attribute to disable the button while the request is in progress.

Do not render a rejected user experience. Because no data is rendered in the SPA in response to this request, silently failing is a **reasonable** (albeit not ideal) user experience.

```
import { useState } from "react"
use axios from "axios"


export default function LikeButton({ eventId }) {
    const [isLoading, setIsLoading] = useState(false)

    function postLike() {
      setIsLoading(true)
      axios.post(`/api/events/${eventId}/like`)
        .catch(() => {})
        .finally(() => {
          setIsLoading(false)
        })
    }

  return (
    <button onClick={handleClick} aria-label="Like">
      <img src="/images/heart.svg" alt="" />
    </button>
  );
}
```

postLike

disabled={isLoading}

Practice Problem 2

The `MarkdownPreviewEditor` component has a left pane with a `<textarea>` for entering Markdown text, and a right pane that renders the markdown as HTML as a **live preview**.

The algorithm to convert markdown to HTML is an **expensive** and slow operation; we only want to run the markdown-to-HTML conversion when the user stops typing for a short period of time (e.g. 500ms). Modify the code below to implement to **debounce** the markdown-to-HTML conversion, so that it only runs when the user stops typing for 500ms. Use the `marked` library to convert Markdown to HTML: `marked.parse("# Example Markdown")` will return the HTML string `<h1>Example Markdown</h1>`.

Note: `dangerouslySetInnerHTML` renders raw HTML in React. Change the `__html` property to the HTML string that you want to render in the preview pane.

```
import { useState, useEffect } from 'react'
import { useDebounce } from 'use-debounce'
import { marked } from 'marked'

export default function MarkdownPreviewEditor({}) {
  const [markdown, setMarkdown] = useState('')
  const [html, setHtml] = useState('')
  Const [debouncedMarkdown] = useDebounce(markdown, 500)

  useEffect(() => {
      Const newHtml = marked.parse(debouncedMarkdown)
      SetHtml(newHtml)
  }, [debouncedMarkdown])


  return (
    <div className="flex flex-col gap-4">
      <textarea placeholder="Enter markdown here..." value={markdown}
        onChange={(e) => setMarkdown(e.target.value)}
      />
      <div className="markdown-preview"
        dangerouslySetInnerHTML={{
          __html: html
        }}
      ></div>
    </div>
  )
}
```