Practice Problem Workshop 6:
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.

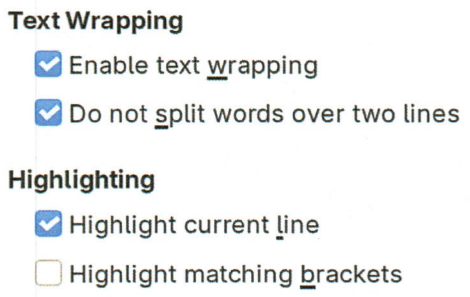
   Discussing the answers will help you practice retrieving the information from memory and will also help you learn from your peers.

Submit this booklet for completion credit.

Practice Problem 1

Code an `EditorSettings` component that has the following four options. Use **exactly one** state for the entire implementation. Set the default state value to current checked items in the design.

Your state should have all four settings. However, you need only implement the checkboxes for "Enable text wrapping" and "Do not split words over two lines".



Text Wrapping

- ✓ Enable text wrapping
- ✓ Do not split words over two lines

Highlighting

- ✓ Highlight current line
- ☐ Highlight matching brackets

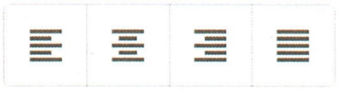
```
import { useState } from "react"
export default function EditorSettings() {
    const [settings, setSettings] = useState({
                                   textWrap: true,
                                   doNotSplit: true,
                                   highlightLine: true,
                                   highlightBrackets: false })

    return <>
        <h2>Text Wrapping</h2>
        <label><input type="checkbox"
         checked={settings.textWrap}
         onChange={(e)=>setSettings((prev) => ({
                 ...prev,
                 textWrap: e.target.checked }))} />

         Enable text wrapping

        </label>
      </>
}
```

Practice Problem 2



Render the `SegmentedButtonGroup` component by rendering a list of of `SegmentedButton` components. The code for `SegmentedButton` is correct:

```
import { useState } from "react"
export default function SegmentedButton({iconUri, alt, variant,
  isPressed, onPressed}) {
  return (<button aria-label={alt}
      className={`segmented-btn--${variant} ${isPressed ? "pressed" : ""}`}
      onClick={onPressed}>
      <img src={iconUri} alt="" />
    </button>)
}
```

This question is only asking you to render a list of `SegmentedButton` components inside the `SegmentedButtonGroup` component. You need not implement the state for the current alignment.

Create an array of button data for left, center, right, and justify alignments.

```
export default function default App() {

  const alignmentBtns = [
     { iconUri: "left.svg", alt: "Align Left" },
     { iconUri: "center.svg", alt: "Align Center" },
     { iconUri: "right.svg", alt: "Align Right" },
     { iconUri: "justify.svg", alt: "Align Justify" }
  ]

  return <SegmentedButtonGroup btnData={alignmentBtns} />
}
```

Turn the page over and implement the `SegmentedButtonGroup` component by rendering a list of `SegmentedButton` components using the `alignmentBtns` data.

Use the `left` variant if button index is 0 and `right` variant if the button index is `btnData.length - 1`. Otherwise, use the `center` variant (or if `btnData.length === 1`).

```
import SegmentedButton from "./components/SegmentedButton"
export default function default SegmentedButtonGroup({ btnData =[] }) {

  return <div>
    {btnData.map((btn, index) => {
      let variant = "center"
      if (index === 0 && btnData.length !== 1){
        variant = "left"
      } else if (index === btnData.length - 1
            && btnData.length !== 1) {
        variant = "right"
      }
      return <SegmentedButton
        key={index}
        iconUri={btn.iconUri}
        alt ={btn.alt}
        variant={variant} />
    })}
  </div>

}
```