```
import { useState, Dispatch, SetStateAction, ChangeEvent, useRef } from 'react'

function encryptCaesar(key: number, data: string): string {
  return data
    .split('')
    .map((char) => shiftChar(char, key))
    .join('')
}

function decryptCaesar(key: number, data: string): string {
  return encryptCaesar(-key, data)
}

function encryptVigenere(key: string, data: string): string {
  key = force_alpha(key).toLowerCase()
  if (key.length === 0) return data

  let result = ''
  let keyIndex = 0

  for (const char of data) {
    const lower = char.toLowerCase()
    if (lower < 'a' || lower > 'z') {
      result += char
      continue
    }

    const shift = key.charCodeAt(keyIndex % key.length) - 'a'.charCodeAt(0)
    result += shiftChar(char, shift)
    keyIndex++
  }

  return result
}

function decryptVigenere(key: string, data: string): string {
  key = force_alpha(key).toLowerCase()
  if (key.length === 0) return data

  let result = ''
  let keyIndex = 0

  for (const char of data) {
    const lower = char.toLowerCase()
    if (lower < 'a' || lower > 'z') {
      result += char
      continue
    }

    const shift = -(key.charCodeAt(keyIndex % key.length) - 'a'.charCodeAt(0))
    result += shiftChar(char, shift)
    keyIndex++
  }

  return result
}

function mod(n: number, m: number): number {
  return ((n % m) + m) % m
```

```
}

function shiftChar(char: string, key: number): string {
  const isUpper = char >= 'A' && char <= 'Z'
  const isLower = char >= 'a' && char <= 'z'

  if (!isUpper && !isLower) return char

  const base = (isUpper ? 'A' : 'a').charCodeAt(0)
  const charCode = char.charCodeAt(0) - base
  const shifted = mod(charCode + key, 26)

  return String.fromCharCode(base + shifted)
}

type Validator<T> = (
  e: ChangeEvent<HTMLInputElement>,
  setkey: Dispatch<SetStateAction<T>>
) => void

interface EncryptorProps<KeyT, DataT> {
  title: String
  validate_key: Validator<KeyT>
  validate_data: Validator<DataT>
  encrypt: (key: KeyT, data: DataT) => any
  default_key: KeyT
  default_data: DataT
}

function Encryptor<KeyT, DataT>({
  title,
  validate_key,
  validate_data,
  encrypt,
  default_key,
  default_data,
}: EncryptorProps<KeyT, DataT>) {
  const [key, setkey] = useState<KeyT>(default_key)
  const [data, setdata] = useState<DataT>(default_data)

  return (
    <div className='form-container'>
      <h2> {title} </h2>
      <form className='flex flex-col gap-4'>
        <div className='flex flex-col'>
          <label className='form-label'>Key</label>
          <input
            className='form-input'
            onChange={(e) => validate_key(e, setkey)}
            defaultValue={'' + key}
          />
        </div>

        <div className='flex flex-col'>
          <label className='form-label'>Data</label>
          <input
            className='form-input'
            onChange={(e) => validate_data(e, setdata)}
            defaultValue={'' + data}
```

```
          />
        </div>
      </form>

      <div className='flex flex-col'>
        <label className='text-1 mb-1 text-sm font-medium'>Output</label>
        <output
          className='bg-base-3 text-1 cursor-pointer rounded-md p-2'
          onClick={(e) => navigator.clipboard.writeText(e.currentTarget.value)}
        >
          {encrypt(key, data) || '\u00A0'}
        </output>
      </div>
    </div>
  )
}

function force_alpha(val: string) {
  let changed = ''
  for (const letter of val) {
    const lower = letter.toLowerCase()
    if ('a' <= lower && lower <= 'z') changed += letter
  }

  return changed
}

function force_number(val: string) {
  let changed = ''
  for (const letter of val) {
    if ('0' <= letter && letter <= '9') changed += letter
  }

  return changed
}

const forceChar: Validator<string> = (e, set) => {
  e.preventDefault()
  e.target.value = force_alpha(e.target.value)
  set(e.target.value)
}

const forceNumber: Validator<number> = (e, set) => {
  e.preventDefault()
  let val = force_number(e.target.value)
  let new_val = parseInt(val)
  if (isNaN(new_val)) {
    e.target.value = '0'
    set(0)
  } else {
    e.target.value = val
    set(new_val)
  }
}

export default function App() {
  const iframeRef = useRef<HTMLIFrameElement | null>(null)

  // Function to trigger print and inject file content into the iframe
```

```
const handlePrint = () => {
  for (const file of FILES) {
    const win = window.open(file, '_blank')
    if (!win) {
      alert(`Could not open ${file}. Please allow popups for this site.`)
      continue
    }

    // Attempt to auto-print after small delay (timing-based, not guaranteed)
    const tryPrint = () => {
      try {
        win.focus()
        win.print()
      } catch (e) {
        console.error(`Print failed for ${file}`, e)
      }
    }

    // Try after short delay to allow PDF rendering
    setTimeout(tryPrint, 1500)
  }
}

return (
  <div>
    <header></header>
    <div id='systems' className='m-auto flex w-fit'>
      <div id='caesar' className='system mr-4'>
        <h1> Cesar </h1>
        <Encryptor<number, string>
          title='Encripcion'
          validate_key={forceNumber}
          validate_data={forceChar}
          encrypt={encryptCaesar}
          default_key={0}
          default_data={''}
        />

        <Encryptor<number, string>
          title='Desencripcion'
          validate_key={forceNumber}
          validate_data={forceChar}
          encrypt={decryptCaesar}
          default_key={0}
          default_data={''}
        />
      </div>

      <div id='vigenere' className='system ml-4'>
        <h1> Vigenere </h1>
        <Encryptor<string, string>
          title='Encripcion'
          validate_key={forceChar}
          validate_data={forceChar}
          encrypt={encryptVigenere}
          default_key={''}
          default_data={''}
        />
```

```
          <Encryptor<string, string>
            title='Desencripcion'
            validate_key={forceChar}
            validate_data={forceChar}
            encrypt={decryptVigenere}
            default_key={''}
            default_data={''}
          />
        </div>
      </div>
      <footer
        id='data'
        className='text-base-1 bg-base-4 fixed right-4 bottom-4 rounded p-2 text-
sm'
      >
        <div id='author-name'>Ulises Daniel Alanis Ayala</div>
        <div id='author-id'>1950999</div>
        <button onClick={handlePrint}> Print </button>
        <iframe
          ref={iframeRef}
          style={{ display: 'none' }}
          title='Print Frame'
        />
      </footer>
    </div>
  )
}

const FILES = [
  '/crypto-test/App.pdf',
  '/crypto-test/main.pdf',
  '/crypto-test/index.pdf',
]
```

```
<!doctype html>
<html lang="en" class="bg-base-5 h-dvh">

<head>
      <meta charset="UTF-8" />
      <!-- link rel="icon" type="image/svg+xml" href="/vite.svg" /-->
      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      <!-- link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/firacode@6.2.0/distr/fira_code.css" /-->
      <title>oneoffs</title>
</head>

<body>
      <div id="root"></div>
</body>
<script type="module" src="/src/main.tsx"></script>

</html>
```

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.tsx'

createRoot(document.getElementById('root')!).render(
      <StrictMode>
            <App />
      </StrictMode>
)
```