```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>Calculator</title>
 <style>
  :root{
    --bg:#0f1220;
    --panel:#181c33;
    --panel-2:#1f2442;
    --accent:#7c6cff;
    --text:#e8eaf6;
    --sub:#aab1d1;
    --danger:#ff5d5d;
    --ok:#62d487;
    --shadow: 0 10px 25px rgba(0,0,0,.35);
    --radius: 16px;
  }
  *{box-sizing:border-box}
  body{
   margin:0; min-height:100svh; display:grid; place-items:center; background:radial-gradient(1200px 600px at 20% 10%, #1b2040 0%, #0f1220 60%), #0f1220; font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, "Helvetica Neue", Arial, "Noto Sans", "Apple Color Emoji", "Segoe UI Emoji"; color:var(--text);
  }
  .app{
```

```
  width:min(420px, 92vw); background:linear-gradient(180deg, var(--panel) 0%, var(--panel-2) 100%); border-radius:var(--radius); box-shadow:var(--shadow), inset 0 1px 0 rgba(255,255,255,.04); overflow:hidden; border:1px solid rgba(255,255,255,.06)

 }

 header{ padding:18px 20px; display:flex; align-items:center; justify-content:space-between; border-bottom:1px solid rgba(255,255,255,.06); background:linear-gradient(180deg, rgba(255,255,255,.03), transparent); }

 header .title{font-weight:700; letter-spacing:.3px}

 header .actions{display:flex; gap:8px}

 .chip{font-size:12px; padding:6px 10px; border-radius:999px; background:rgba(124,108,255,.15); color:#c7c1ff; border:1px solid rgba(124,108,255,.35)}


 .display{ padding:18px 20px 6px; }

 .expr{ min-height:28px; color:var(--sub); text-align:right; font-size:14px; letter-spacing:.4px; word-break:break-all; }

 .screen{ margin-top:6px; font-size:38px; text-align:right; font-weight:700; letter-spacing:1px; word-break:break-all; }


 .keypad{ display:grid; grid-template-columns: repeat(4, 1fr); gap:10px; padding:16px; }

 button.key{ appearance:none; border:0; padding:16px 12px; border-radius:14px; font-size:18px; font-weight:700; color:var(--text); background:#202647; box-shadow: 0 6px 14px rgba(0,0,0,.25), inset 0 -2px 0 rgba(255,255,255,.03); cursor:pointer; transition: transform .05s ease, filter .2s ease, background .2s ease; user-select:none; }

 button.key:active{ transform: translateY(1px) scale(.995); }

 .key.op{ background:#242a55; color:#d7dcff; }

 .key.eq{ background: linear-gradient(180deg, #8a7bff, #6b5aff); color:white; box-shadow: 0 8px 18px rgba(108,96,255,.45); }

 .key.wide{ grid-column: span 2; }

 .key.alt{ background:#2a315e; color:#d0e6ff; }
```

```
   .key.danger{ background:#3a2240; color:#ffd2e1; }


   .footer{ display:flex; align-items:center; justify-content:space-between; padding:8px
14px 14px; }
   .history{
    max-height: 120px; overflow:auto; padding:8px; margin:0; list-style:none; display:flex;
gap:8px; flex-wrap:wrap; align-items:flex-start
   }
   .history li{
    font-size:12px; background:rgba(255,255,255,.06); padding:6px 8px; border-radius:8px;
color:#d2d6f2; border:1px solid rgba(255,255,255,.08); cursor:pointer
   }
   .muted{ color:var(--sub); font-size:12px }


   /* Scrollbar (for Windows users) */
   ::-webkit-scrollbar{height:8px;width:10px}
   ::-webkit-scrollbar-thumb{background:rgba(255,255,255,.12);border-radius:20px}
   ::-webkit-scrollbar-track{background:transparent}
  </style>
</head>
<body>
  <div class="app" id="app" aria-label="Calculator">
   <header>
    <div class="title">Calculator</div>
    <div class="actions">
     <span class="chip" title="Keyboard shortcuts: digits, + - * / ( ) .  Backspace, Enter
=">Keyboard ✓</span>
```

```
  </div>
</header>

<div class="display">
  <div class="expr" id="expression" aria-live="polite"></div>
  <div class="screen" id="screen" aria-live="polite">0</div>
</div>

<div class="keypad" role="group" aria-label="Keypad">
  <button class="key alt" data-key="C" title="Clear">C</button>
  <button class="key alt" data-key="DEL" title="Delete">DEL</button>
  <button class="key op" data-key="%" title="Percent">%</button>
  <button class="key op" data-key="/" title="Divide">÷</button>

  <button class="key" data-key="7">7</button>
  <button class="key" data-key="8">8</button>
  <button class="key" data-key="9">9</button>
  <button class="key op" data-key="*" title="Multiply">×</button>

  <button class="key" data-key="4">4</button>
  <button class="key" data-key="5">5</button>
  <button class="key" data-key="6">6</button>
  <button class="key op" data-key="-" title="Minus">−</button>

  <button class="key" data-key="1">1</button>
  <button class="key" data-key="2">2</button>
```

```
    <button class="key" data-key="3">3</button>
    <button class="key op" data-key="+" title="Plus">+</button>

    <button class="key" data-key="(">(</button>
    <button class="key" data-key=")">)</button>
    <button class="key" data-key="0">0</button>
    <button class="key" data-key=".">.</button>

    <button class="key wide" data-key="±" title="Toggle sign">±</button>
    <button class="key eq wide" data-key="=" title="Equals">=</button>
  </div>

  <div class="footer">
    <div class="muted">History</div>
    <ul class="history" id="history" aria-label="Previous results"></ul>
  </div>
</div>

<script>
  (function(){
    const screen = document.getElementById('screen');
    const exprEl = document.getElementById('expression');
    const keypad = document.querySelector('.keypad');
    const historyEl = document.getElementById('history');

    let expr = '';
```

```
let justEvaluated = false;

const operators = new Set(['+','-','*','/']);

function update(){
  exprEl.textContent = expr || '';
}
function setScreen(val){
  screen.textContent = val;
}

function insert(token){
  if(justEvaluated && /[\d.(]/.test(token)){
    // Start a new expression if user types number/decimal/(
    expr = '';
    justEvaluated = false;
  } else if (justEvaluated && operators.has(token)){
    // continue with result if an operator is pressed after equals
    justEvaluated = false;
  }
  expr += token;
  update();
}

function clearAll(){ expr = ''; setScreen('0'); update(); }
```

```
function del(){
  if (justEvaluated){ justEvaluated = false; }
  expr = expr.slice(0,-1); update();
}

function toggleSign(){
  // Toggle sign of the last number (basic heuristic)
  const m = /(-?\d*\.?\d+)(?!.*\d)/.exec(expr);
  if(!m) return;
  const num = m[1];
  const start = m.index;
  const isNeg = num.startsWith('-');
  const replacement = isNeg ? num.slice(1) : '-' + num;
  expr = expr.slice(0,start) + replacement + expr.slice(start + num.length);
  update();
}

function toSafeExpression(raw){
  let e = raw.replace(/×/g,'*').replace(/÷/g,'/');
  // Convert 50% -> (50/100)
  e = e.replace(/(\d+(?:\.\d+)?)%/g,'($1/100)');
  // Disallow anything except digits operators parentheses dot and whitespace
  if(!/^[-+*/().\d\s]+$/.test(e)) throw new Error('Invalid characters');
  // Basic sanitization against malformed operator chains (allow unary minus)
  e = e.replace(/\s+/g,'');
  return e;
```

```
}

function evaluate(){
 if(!expr) return;
 try{
  const safe = toSafeExpression(expr);
  // Disallow ending with operator
  if(/[+\-*/.]$/.test(safe)) throw new Error('Incomplete');
  // Evaluate safely using Function after sanitization
  const result = Function('return (' + safe + ')')();
  const display = Number.isFinite(result) ? formatNumber(result) : 'Error';
  setScreen(display);
  pushHistory(expr + ' = ' + display);
  expr = String(result);
  justEvaluated = true;
  update();
 }catch(err){
  setScreen('Error');
 }
}

function formatNumber(n){
 // Avoid scientific notation for reasonable ranges
 if (Math.abs(n) < 1e12 && Math.abs(n) > 1e-8) {
  return parseFloat(n.toFixed(10)).toString();
 }
```

```
  return n.toString();
}

function pushHistory(item){
  const li = document.createElement('li');
  li.textContent = item;
  li.title = 'Click to reuse result';
  li.addEventListener('click',()=>{
    const m = /=\s*(-?\d+(?:\.\d+)?)$/.exec(item);
    if (m) {
      expr = m[1]; justEvaluated = false; update(); setScreen(expr);
    }
  });
  historyEl.prepend(li);
  // keep the last 10 items
  while(historyEl.children.length > 10){ historyEl.removeChild(historyEl.lastChild); }
}

// Click handlers
keypad.addEventListener('click', (e)=>{
  const b = e.target.closest('button.key'); if(!b) return;
  const k = b.dataset.key;
  switch(k){
    case 'C': clearAll(); break;
    case 'DEL': del(); break;
    case '=': evaluate(); break;
```

```
      case '±': toggleSign(); break;
      default: insert(k);
    }
  });

  // Keyboard support
  window.addEventListener('keydown', (e)=>{
    const k = e.key;
    if ((k >= '0' && k <= '9') || ['+','-','*','/','(',')','.'].includes(k)){
      insert(k);
    } else if (k === 'Enter' || k === '='){
      e.preventDefault(); evaluate();
    } else if (k === 'Backspace'){
      del();
    } else if (k.toLowerCase() === 'c'){
      // Ctrl+L often clears address bar; keep C to clear
      clearAll();
    } else if (k === '%'){
      insert('%');
    }
  });

  update();
 })();
</script>
</body>
```

</html>