```
// All code is written by me
//==========================================================================
// Name badge assignment (App.js)
//==========================================================================
import * as ScreenOrientation from 'expo-screen-orientation';
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, SafeAreaView } from 'react-native';
import { useEffect } from 'react';

export default function App() {
 useEffect(() => {
  ScreenOrientation.lockAsync(ScreenOrientation.OrientationLock.LANDSCAPE_LEFT);
 }, []);

 return (
  <SafeAreaView style={styles.container}>
   <View>
    <Text style={styles.welcomeText}>Hello</Text>
    <Text style={styles.subtitleText}>my name is</Text>
   </View>
   <View style={styles.nameTagContainer}>
    <Text style={styles.nameText}>Bryan 💻 (he/him)</Text>
   </View>

   <StatusBar style="auto" />
  </SafeAreaView>
 );
}

const styles = StyleSheet.create({
 container: {
  flex: 1,
  width: '100%',
  height: '100%',
  backgroundColor: 'blue',
  alignItems: 'center',
  justifyContent: 'center',
 },
 welcomeText: {
  fontSize: 90,
  textTransform: 'uppercase',
  fontWeight: 'bold',
  color: 'white',
  textAlign: 'center'
 },
 subtitleText: {
  fontSize: 30,
  textTransform: 'uppercase',
  fontWeight: 'bold',
  color: 'white',
  marginBottom: 20,
  textAlign: 'center'
 },
 nameTagContainer:{
  width: '90%',
  height: '55%',
  backgroundColor: 'white',
  borderRadius: 10,
  justifyContent: 'center'
 },
 nameText: {
  fontSize: 60,
  textAlign: 'center',
  fontWeight: 'bold',
  color: 'purple'
 },
});

//============================================================================
// Tic tac toe (App.js)
//============================================================================
import { Text, View, SafeAreaView, StyleSheet } from 'react-native';
import * as ScreenOrientation from 'expo-screen-orientation';
import { useEffect } from 'react';

const Symbol = ({ value }) => (
 <Text style={[styles.symbol, value === 'O' ? styles.circleColor : styles.crossColor]}>
  {value}
 </Text>
);

export default function App() {

 useEffect(() => {
   ScreenOrientation.lockAsync(ScreenOrientation.OrientationLock.PORTRAIT);
  }, []);

 return (
  <SafeAreaView style={ styles.container }>

   <View style={styles.row}>
    <View style={[styles.cell, styles.borderBottom, styles.borderRight]}>
     <Symbol value="O" />
    </View>
    <View style={[styles.cell, styles.borderBottom, styles.borderRight]}>
     <Symbol value="O" />
    </View>
    <View style={[styles.cell, styles.borderBottom]}>
     <Symbol value="X" />
    </View>
   </View>

   <View style={styles.row}>
    <View style={[styles.cell, styles.borderBottom, styles.borderRight]}>
```

```
        <Symbol value="X" />
       </View>
       <View style={[styles.cell, styles.borderBottom, styles.borderRight]}>
        <Symbol value="O" />
       </View>
       <View style={[styles.cell, styles.borderBottom]}>
        <Symbol value="O" />
       </View>
      </View>

      <View style={styles.row}>
       <View style={[styles.cell, styles.borderRight]}>
        <Symbol value="X" />
       </View>
       <View style={[styles.cell, styles.borderRight]}>
        <Symbol value="X" />
       </View>
       <View style={styles.cell}>
        <Symbol value="O" />
       </View>
      </View>
     </SafeAreaView>
   );
  }

  const styles = StyleSheet.create({
```

```
container: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: 'black',
  padding: 8
},
symbol:{
   fontSize:100,
  textAlign:'center'
},
row: {
  flexDirection: 'row',
  width:'80%',
  justifyContent:'center',
  alignItems:'center',
  marginLeft: 10,
  marginRight: 10


},
borderBottom:{
  borderBottomColor:'blue',
  borderBottomWidth:4
},
borderRight:{
  borderRightColor:'blue',
  borderRightWidth:4
 },
 circleColor: {
  color: 'green'
 },
 crossColor: {
  color: 'red'
 },
 cell:{
  height: '100%',
  flex:1
 }
});

//=============================================================================
// Calculator (App.js)
//=============================================================================

import { StatusBar } from 'expo-status-bar';
import { StyleSheet,
    Text,
    View,
    TouchableOpacity,
    Dimensions,
    SafeAreaView } from 'react-native';
```

```
import * as ScreenOrientation from 'expo-screen-orientation';


import { useState } from 'react';

import { useEffect } from 'react';


const buttonWidth = Dimensions.get('window').width / 4;

const toRadians = (deg) => deg * (Math.PI / 100);


export default function App() {


useEffect(() => {

   ScreenOrientation.lockAsync(ScreenOrientation.OrientationLock.PORTRAIT);

 }, []);


 //  Declare variables

 const [answerValue, setAnswerValue] = useState('0');

 const [memoryValue, setMemoryValue] = useState(null);

 const [operatorValue, setOperatorValue] = useState(null);

 const [readyToReplace, setReadyToReplace] = useState(true);


 const buttonPressed = (value) => {

  if(!isNaN(value)) {

    handleNumber(value);


  } else if (value === 'C') {

    setAnswerValue('0');
    setMemoryValue(null);
    setOperatorValue(null);
    setReadyToReplace(true);

  } else if (value === '+/-') {
    setAnswerValue((prev) => (-parseFloat(prev) * -1).toFixed(2).toString());

  } else if (value === '%') {
    setAnswerValue((prev) => (parseFloat(prev) * 0.01).toFixed(2),toString());

  } else if (value=== '=') {
      if (operatorValue && memoryValue !== null) {
        const result = calculateEquals();
        setAnswerValue(result.toString());
        setMemoryValue(null);
        setOperatorValue(null);
        setReadyToReplace(true);
      }
  }else if (value === 'sin')  {
    try{
      const result = Math.sin(toRadians(parseFloat(answerValue)));
      setAnswerValue(isFinite(result) ? result.toFixed(2).toString() : 'Error');
      setReadyToReplace(true);
    } catch {
      setAnswerValue('Error');
    }
```

```
}else if (value === 'cos') {
 try{
  const result = Math.cos(toRadians(parseFloat(answerValue)));
  setAnswerValue(isFinite(result) ? result.toFixed(2).toString() : 'Error');
  setReadyToReplace(true);
 } catch {
  setAnswerValue('Error');
 }
}else if (value === 'tan') {
 try{
  const result = Math.tan(toRadians(parseFloat(answerValue)));
  setAnswerValue(isFinite(result) ? result.toFixed(2).toString() : 'Error');
  setReadyToReplace(true);
 } catch {
  setAnswerValue('Error');
 }
}else if(value === 'π') {
 setAnswerValue(Math.PI.toFixed(2),toString());
 setReadyToReplace(true);
}else if(value === '√') {
 setAnswerValue(Math.sqrt(parseFloat(answerValue)).toFixed(2).toString());
 setReadyToReplace(true);
}else if(value === 'x²') {
 try{
  const result = Math.pow(parseFloat(answerValue), 2);
  setAnswerValue(isFinite(result) ? result.toFixed(2).toString() : 'Error');
```

```
    setReadyToReplace(true);
   } catch {
    setAnswerValue('Error');
   }
 }else if(value === 'log') {
   try{
    const result = Math.log10(parseFloat(answerValue));
    setAnswerValue(isFinite(result) ? result.toFixed(2).toString() : 'Error');
    setReadyToReplace(true);
   } catch {
    setAnswerValue('Error');
   }
 }else if(value === 'ln')  {
   try{
    const result = Math.log(parseFloat(answerValue));
    setAnswerValue(isFinite(result) ? result.toFixed(2).toString() : 'Error');
    setReadyToReplace(true);
   } catch {
    setAnswerValue('Error');
   }
 }else if (value === '.')  {
   if(readyToReplace) {
    setAnswerValue('0.');
    setReadyToReplace(false);
   } else if (!answerValue.includes('.')){
    setAnswerValue((prev) => prev + '.');
   }
  }

  else  {
    if (operatorValue !== null && memoryValue !== null && !readyToReplace) {
     const result = calculateEquals();
     setMemoryValue(result);
     setAnswerValue(result.toString());

    }else{
     setMemoryValue(parseFloat(answerValue));
    }
    setOperatorValue(value);
    setReadyToReplace(true);
  }
};

const handleNumber = (num) => {
  if(readyToReplace || answerValue === '0') {
   setAnswerValue(num.toString());
   setReadyToReplace(false);

  }else{
   setAnswerValue((prev) => prev + num.toString());

  }
```

```
};

const calculateEquals = () => {
  const prev = parseFloat(memoryValue);
  const current = parseFloat(answerValue);
  let result = 0;

  switch(operatorValue) {
    case '+':
      result = prev + current;
      break;
    case '-':
      result = prev - current;
      break;
    case 'x':
      result = prev * current;
      break;
    case '/':
      result = current !== 0 ? prev / current : 'Error';
      break;
  }
  return result;
};

const renderButton = (value, type = 'default', extraStyle = {}) => (
  <TouchableOpacity
```

```
    onPress={() => buttonPressed(value)}
    style={[
      styles.button,
      type === 'gray' && styles.grayButton,
      type === 'blue' &&  styles.blueButton,
      type === 'sci' && styles.sciButton,
      value === '0' && styles.longButton,
      extraStyle,
    ]}
  >
    <Text style={styles.buttonText}>{value}</Text>
  </TouchableOpacity>
);

return (
  <SafeAreaView style={styles.container}>
    <StatusBar barStyle="light-content"/>
    <View style={styles.resultContainer}>
      <Text style={styles.resultText}>{answerValue}</Text>
    </View>
    <View style={styles.buttonRow}>
      {renderButton(answerValue==='0'?'AC':'C','gray')}
      {renderButton('+/-', 'gray')}
      {renderButton('%', 'gray')}
      {renderButton('/', 'blue')}
    </View>
```

```
<View style={styles.buttonRow}>
 {renderButton('sin', 'sci')}
 {renderButton('cos', 'sci')}
 {renderButton('tan', 'sci')}
 {renderButton('π', 'sci')}
</View>
<View style={styles.buttonRow}>
 {renderButton('log', 'sci')}
 {renderButton('ln', 'sci')}
 {renderButton('√', 'sci')}
 {renderButton('x²', 'sci')}
</View>
<View style={styles.buttonRow}>
 {renderButton('7')}
 {renderButton('8')}
 {renderButton('9')}
 {renderButton('x', 'blue')}
</View>
<View style={styles.buttonRow}>
 {renderButton('4')}
 {renderButton('5')}
 {renderButton('6')}
 {renderButton('-', 'blue')}
</View>
<View style={styles.buttonRow}>
 {renderButton('1')}
```

```
      {renderButton('2')}
      {renderButton('3')}
      {renderButton('+', 'blue')}
     </View>
     <View style={styles.buttonRow}>
      {renderButton('0', 'default', styles.longButton)}
      {renderButton('.')}
      {renderButton('=', 'blue')}
     </View>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  container: {
   flex: 1,
   backgroundColor: 'black',
   justifyContent: 'flex-end',
  },
  resultContainer: {
   alignItems: 'flex-end',
   padding: 20,
  },
  resultText: {
   fontSize: 80,
   color: 'white',
},
buttonRow: {
  flexDirection: 'row',
  justifyContent: 'space-between',
  marginBottom: 10,
  paddingHorizontal: 10,
},
button: {
  backgroundColor: '#333333',
  width: buttonWidth - 15,
  height: buttonWidth - 15,
  borderRadius: (buttonWidth - 15) / 2,
  justifyContent: 'center',
  alignItems: 'center',
},
longButton: {
  width: buttonWidth * 2 - 25,
  alignItems: 'flex-start',
  paddingLeft: 30,
},
grayButton: {
  backgroundColor: '#a5a5a5',
},
blueButton: {
  backgroundColor: '#007AFF',
},
 buttonText: {
  fontSize: 30,
  color: 'white',
 },
 sciButton: {
  backgroundColor: '#555555',
 },
});

//=========================================================================
// Food Ordering App (App.js)
//=========================================================================
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { Cell, Section, TableView } from 'react-native-tableview-simple';
import { Text, View, StyleSheet, ScrollView, Image } from 'react-native';
import { useNavigation, useRoute } from '@react-navigation/native';
import * as ScreenOrientation from 'expo-screen-orientation';
import { useEffect } from 'react';

function FeaturedEateriesScreen() {
 const navigate = useNavigation();

 const eateryList = [
  {
   rows: [
    {
      name: "Bryan's BBQ",
      description: "Grill, Meats, $$$",
      deliveryTime: "20-40",
      thumbnail: require('./assets/barbequepic.png')
    },
    {
      name: "Noodles",
      description: "Asian, Noodles, $",
      deliveryTime: "15-25",
      thumbnail: require('./assets/noodlespic.png')
    }
   ]
  }
];

const EateryCard = (props) => (
  <Cell
   backgroundColor="transparent"
   onPress={() =>
    navigate.navigate("Details", {
     restaurantName: props.title
    })
   }
   cellContentView={
    <View style={props.cardContainer}>
```

```
        <Image style={props.imageStyle} source={props.imageSource} />
        <View style={styles.deliveryTimeContainer}>
          <Text style={styles.deliveryText}>{props.deliveryDuration}</Text>
          <Text style={styles.minsText}>mins</Text>
        </View>
        <Text style={props.titleStyle}>{props.title}</Text>
        <Text style={props.subtitleStyle}>{props.subtitle}</Text>
      </View>
    }
  />
);

return (
  <View style={styles.pageWrapper}>
    <Text style={styles.title}>Jack's Picks</Text>
    <ScrollView>
      <TableView>
        {eateryList.map((block, i) => (
          <Section key={i} hideSeparator={false}>
            {block.rows.map((item, j) => (
              <EateryCard
                key={j}
                title={item.name}
                subtitle={item.description}
                deliveryDuration={item.deliveryTime}
                imageSource={item.thumbnail}
        titleStyle={{
          marginTop: 5,
          fontSize: 20,
          fontWeight: 'bold'
        }}
        subtitleStyle={{
          marginTop: 5,
          fontSize: 14
        }}
        cardContainer={{
          width: '100%',
          height: 250,
          backgroundColor: 'transparent',
          marginLeft: 2
        }}
        imageStyle={{
          width: 100,
          height: 100,
          borderRadius: 20,
          marginBottom: 10
        }}
      />
    ))}
  </Section>
 ))}
</TableView>
```

```
    </ScrollView>
   </View>
  );
 }

function RestaurantDetailScreen() {
  const params = useRoute();
  const restaurantTitle = params?.params?.restaurantName || 'Unknown';

  return (
   <View style={styles.detailContainer}>
    <Text style={styles.detailText}>{restaurantTitle}</Text>
   </View>
  );
}

const NavigatorStack = createNativeStackNavigator();

export default function App() {

  useEffect(() => {
      ScreenOrientation.lockAsync(ScreenOrientation.OrientationLock.PORTRAIT);
    }, []);

  return (
   <NavigationContainer>
   <NavigatorStack.Navigator>
    <NavigatorStack.Screen
     name="Home"
     component={FeaturedEateriesScreen}
     options={{ title: 'Restaurants', headerTitleAlign: 'center' }}
    />
    <NavigatorStack.Screen
     name="Details"
     component={RestaurantDetailScreen}
     options={{ title: 'Menu', headerTitleAlign: 'center' }}
    />
   </NavigatorStack.Navigator>
  </NavigationContainer>
 );
}

const styles = StyleSheet.create({
 pageWrapper: {
  flex: 1,
  justifyContent: 'center',
  backgroundColor: 'lightblue',
  padding: 8
 },
 title: {
  fontSize: 26,
  fontWeight: 'bold',
  textAlign: 'center',
  marginTop: 10
 },
 detailContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center'
 },
 detailText: {
  fontSize: 22,
  fontWeight: 'bold'
 },
 deliveryTimeContainer: {
  width: 100,
  height: 60,
  backgroundColor: 'white',
  borderRadius: 50,
  right: 0,
  position: 'absolute',
  marginTop: 170,
  marginRight: 20,
  justifyContent: 'center'
 },
 deliveryText: {
  fontSize: 17,
  fontWeight: 'bold',
  textAlign: 'center'
 },
 minsText: {
  fontSize: 17,
  fontWeight: 'bold',
  textAlign: 'center',
  marginTop: -5
 }
});

//=========================================================================
// End of script, all code is written by me
//=========================================================================
```