CODE FIRST GIRLS

CFGDEGREE

FULLSTACK ASSESSMENT - 2 HOURS

SECTION TYPE	TOTAL MARKS AVAILABLE	NOTES
Redux (pseudocode code/ reasoning explanation)	25	Multiple questions, all comprising 25 total
Algorithms 1 (Coding)	15	1 question only
Algorithms 2 (Coding)	20	1 question only
60 marks available total		

Questions begin on the next page

Redux	**25 MARKS**

This question involves using built-in React Hook for a simple attendance app.

Here are notes to help: -

- *useState() : This allows you to have states variables in functional components. It helps to set and retrieve the state.*
- *A reducer: This is a function that returns some state data, triggered by an action type.*
- *An action: This is dispatched by components and is represented as one object that contains type property and sometimes payload property. It tells the reducer how to change the state. Here is an example of the shape of an action -> { type: 'GREETINGS', payload: 'Hello'}*
- *Dispatch: this helps update the state by sending the type of action to the reducer function for it to perform its job. To invoke a dispatch function, you need to pass action as an argument to the dispatch function, e.g. dispatch ({type: "SOMETHING", payload: "SOMETHING" })}*

*** Remember: Submit pseudocode or simply describe the solution.*

Part 1 (5 marks):

```
function countReducer(state = initialState, action) {
          if (action.type === 'increment') {
            return {
              value: state.value + 1
            }
          }
    return state
}
```

Figure 1

1. Can you provide a brief summary of what is happening in this function code?
   - from function above countReducer it takes the current state and an action. If action is increment it will return new state with value increased by 1 if not it will return current state

2. Add one action that tells the reducer to reduce the state value by 1
   - can reduce by using Decrement action add like this

```
function countReducer(state = initialState, action){
        if (action.type === 'increment'){
                return {value: state.value + 1}

                else if (action.type === 'decrement'){
                return {value: state.value -1}
                }

                return state
        }
```

3. Add one action that tells the reducer to reset the state

```
        if (action.type === 'increment'){
                return {value: state.value + 1}
                else if (action.type === 'decrement'){
                return {value: state.value -1}
                }
                else if (action.type === 'reset'){
                return initialState}
                return state
        }
```

Part 2 (10 marks):

This section involves handling state l ocally.In the code above the useState hook is used to set the state of a variable inside the component.

```
33   const classInfo = () => {
34      let [studentsCount, setStudentsCount] = useState(0);
35
36       return(
37         <div>
38           <p>Number of students in class room: {studentsCount}</p>
39           <button onClick={?????} >Add Student</button>
40         </div>
41       )
42   }
```

1. Can you provide a brief summary on what is happening on line 34, 39?

   - On line 34 useState hook is used to initialize studentCount to 0

   and setStudentCount is a function can be update on this state using setState

   - On line 39 This line created a button 'Add Student' and

   currently it's no function call when it clicked because onClick handler is ???

2. When a user clicks on the "Add student" button update the state (studentsCount) to include only the total number of students who are present. Using the data provided below:

```
const students =  [

          {name: "Nrupul", present: false},
          {name: "Prateek", present: true },
          {name: "Jane", present: true },
          {name: "Paul", present: false },
          {name: "Luke", present: true }

        ]
```

Figure 3

   a. Write a *pseudocode* of how your function would look.

```
// add handler function and call it
const classInfo= () => {
  let [studentCount, setStudentsCount] = useState(0)
const handlerAddStudent = () => {
// using filter loop through array
const presentStudent = students.filter(student => student.present).length
setStudentsCount(presentStudent)
        }

  return (
        <div>
        <p>num of students in class: {studentCount}</p>
        <button onClick={handlerAddStudent}>Add student</button>
        </div>
          )
  }
```

b. How do you ensure that the function is triggered when the button is clicked?

- <button onClick={handlerAddStudent}>Add student</button>

button in JSX element using onClick function is provided to onClick to executed

c. How will you update the state with the result of your function?

- I updated using setState from code above
  setStudentsCount(presentStudent)

Part 3 (10 marks):

Now let's use dispatch to update the state on button click

```
166  const initialState = { value: 0 }
167
168  function countReducer(state = initialState, action) {
169
170          if (action.type === 'increment') {
171
172            return {
173
174              value: state.value + action.payload
175            }
176          }
177      return state
178  }
```

Figure 4

1. A change of code was made on line 174 (figure 4), can you briefly explain what that would do?
   - The change that made is the increment to rely on action.payload now rather than just increment by 1 each time
2. Let's say we don't want to set the state locally anymore and want to use dispatch. How would you ensure that an "*increment*" action that also contains the result of the studentCount is dispatched on button click? According to your answer in part 2.2b what would need to be changed?

- first need to modify classInfo something similar to this
  add using dispatch const dispatch = useDispatch(); useSelector ?
  const studentCount = useSelector(state => state.studentCount);
  also set up Redux store, action and reducer
  also set up provider in index.js
  make changes in handlerAddttudent to use dispatch

3. Which code do you think is best suited to ensure that the "increment" action updates the state with the *correct* total number of students who are present. *Is it Figure 4? Or Figure 5? Explain the code difference and your reasoning*

```
166  const initialState = { value: 0 }
167
168  function countReducer(state = initialState, action) {
169
170          if (action.type === 'increment') {
171
172            return {
173
174              value: action.payload
175            }
176          }
177      return state
178  }
```

Figure 5

- Depends what the situation if It need to be use in multiple components I would prefer using dispatch

Algorithms 1 (Coding)	15 MARKS (1 question)

Write an algorithm that returns true if the given string is a palindrome. Otherwise, return false.

Note: A String is said to be a palindrome if the string is spelled the same way forward and backwards.

For example, some sample input and outputs would be:

	stringA value	Output value
Sample Input 1	radar	True
Sample Input 2	level	True
Sample Input 3	Pencil	False
Sample Input 4	a	True

In your answer, please discuss your solution - what is its Big O Time & Space complexity? Why have you chosen this approach? Could there be a more efficient way (and if so, how)?

// start looking at the first letter and the end if they're the same move to next one if not return false

```
const isPalindrome = (word) => {
  let start = 0
  let end = word.length - 1
  while (start < end) {
    if (word[start] !== word[end]) {
      return false
    }
    start++
    end--
  }
  return true
}
```

Time complexity of O(n) operation run in linear time have to check every letter (I'm not sure about this)
but Space Complexity is O(1) constant space because It doesn't need to use any extra space to work with only need to check half of the word

Algorithms 2 (Coding)	**20 MARKS (1 question)**

If you are short on time, you can also submit pseudocode or simply describe what solution you'd write in code (just describe what you have in your mind) - this cannot attain full marks, but it is still a perfectly acceptable answer and can get partial marks.

In essence, just submit what you have even if you don't know the answer!

Write a function that takes in an unsorted array of any size. These elements are in the range of 1 to n. In the input array one number is missing. Your function should return the missing number.

If the input array contains a negative number or non-numeric value then return an error with the correct error message.

For example, some sample input and outputs would be:

	Array input	**Output**
Sample Input 1	[4,5,1,3, 5]	Missing = 2
Sample Input 2	[4, 3,5, 6, 8, 2, 1, 3]	Missing = 7,
Sample Input 3	[1,2,3,4]	"Nothing is missing"
Sample Input 4	[4,5, -1,3, 5]	"Invalid input, negative number detected"
Sample Input 5	[3, 4, 5, 6, 'cfg']	"Invalid input, non-numeric value detected"

In your answer, please discuss your solution - what is its Big O Time & Space complexity? Why have you chosen this approach? Could there be a more e cient way (and if so, how)?

If you are short on time, you can also submit pseudocode or simply describe what solution you'd write in code (just describe what you have in your mind) - this cannot attain full marks, but it is still a perfectly acceptable answer and can get partial marks.

In essence, just submit what you have even if you don't know the answer!

The code not running :(

```
// check negative number first
// looks for the biggest number in the array.
// If the biggest number is not bigger than the number of items in
the array means no numbers are missing
// so  "nothing is missing".
// If the biggest number is bigger than the number of items it
means there might be a number missing. So starts check from 1 and
checks every number up to the biggest number to see if it's in the
array.
// The first number it finds that's not in the array is the missing
number, so it says that number "is missing".
// If it checks all the numbers up to the biggest number and
doesn't find any missing numbers, it says "nothing is missing".
```

```
- const isMissing = (nums) => {
  let max = 0
  for (let missingNum = 0; missingNum < nums.length; missingNum++)
{
    if (nums[missingNum] !== 'number' || nums[missingNum] <= 0) {
      throw new Error('Invalid input, non-numeric value detected')
    }
    if (nums[missingNum] > max) {
      max = nums[missingNum]
    }
  }

  if (max <= nums.length) {
    return 'nothing is missing'
  }

  for (let missingNum = 1; missingNum <= max; missingNum++) {
    if (!nums.includes(missingNum)) {
      return `${missingNum} is missing`
    }
  }

  return 'nothing is missing'
}


console.log(isMissing([1, 2, 3, 4, 5])) // nothing is missing
console.log(isMissing([4,5,1,3, 5, 6])) // 2 is missing
console.log(isMissing([ 1, 3, 7, 5, 6, 2 ])) // 4 is missing
console.log(isMissing([4,5,-1,3, 5])) // THROW ERROR Invalid input,
non-numeric value detected
```

```
console.log(isMissing([ 1, 3, 7, 5, 6, 2 ])) // THROW ERROR Invalid input, non-numeric value detected
The space complexity is O(1)
```