```
Queue2<String> q1, q2;
q1 = new QueueImpl1<String>();
q2 = new QueueImpl2<String>();
```

6406531484784. ✔

```
Queue1<String> q1, q2;
q1 = new QueueImpl1<String>();
q2 = new QueueImpl2<String>();
```

6406531484785. ✖

```
QueueImpl1<String> q1, q2;
q1 = new QueueImpl1<String>();
q2 = new QueueImpl2<String>();
```

6406531484786. ✖

```
QueueImpl2<String> q1, q2;
q1 = new QueueImpl1<String>();
q2 = new QueueImpl2<String>();
```

AppDev2

Section Id :	64065328984
Section Number :	10
Section type :	Online
Mandatory or Optional :	Mandatory
Number of Questions :	17
Number of Questions to be attempted :	17
Section Marks :	50
Display Number Panel :	Yes
Group All Questions :	No
Enable Mark as Answered Mark for Review and Clear Response :	Yes
Maximum Instruction Time :	0
Sub-Section Number :	1
Sub-Section Id :	64065363337

Question Shuffling Allowed : No

Is Section Default? : null

Question Number : 169 Question Id : 640653445612 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 0

Question Label : Multiple Choice Question

THIS IS QUESTION PAPER FOR THE SUBJECT "DIPLOMA LEVEL: MODERN APPLICATION DEVELOPMENT 2"

ARE YOU SURE YOU HAVE TO WRITE EXAM FOR THIS SUBJECT?
CROSS CHECK YOUR HALL TICKET TO CONFIRM THE SUBJECTS TO BE WRITTEN.

(IF IT IS NOT THE CORRECT SUBJECT, PLS CHECK THE SECTION AT THE TOP FOR THE SUBJECTS REGISTERED BY YOU)

Options :

6406531484795. ✔ Yes

6406531484796. ✖ No

Sub-Section Number : 2

Sub-Section Id : 64065363338

Question Shuffling Allowed : Yes

Is Section Default? : null

Question Number : 170 Question Id : 640653445613 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4.5

Question Label : Multiple Choice Question

Consider the below javascript program.

```
async function result (x) {
    const y = 3;
    return new Promise ((reject, resolve) => {
        if (((x * 15) / (5 * x) - 2) == 1)
            reject(y ** 2)
        console.log("Inside Promise")
        resolve(y ** 3)
    })
}

result(x).then(
rej => console.log("Promise rejected with the value", rej),
res => console.log("Promise resolved with the value", res)
).then(data => {
    console.log("Data received is", data);
    return "5"
}).finally(data => {
    console.log("Data received is", data);
    return "15"
}).then(data => console.log("Data received is", data))
```

Assuming the variable "x" passed to the "result" function is a whole number between 1 and 99 (including 1 and 99), what will be shown on the console, if the above program is executed?

Options :

6406531484797. ✖
Promise rejected with the value 9
Data received is undefined
Data received is 5
Data received is 15

6406531484798. ✖
Inside Promise
Promise rejected with the value 8
Data received is undefined
Data received is undefined
Data received is 15

6406531484799. ✔
Inside Promise
Promise rejected with the value 9
Data received is undefined
Data received is undefined
Data received is 5

6406531484800. ✖

Promise rejected with the value 8
Data received is undefined
Data received is undefined
Data received is 5

Question Number : 171 Question Id : 640653445617 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4.5

Question Label : Multiple Choice Question

Consider the below javascript program.

```
var num = 15;

const obj = {
    num : 8,

    func : function () {
        console.log("Number 1:", this.num, "Number 2:", num)
    }
}
obj.func.call()
```

What will be the output of the above program, if it is executed as a script and in a REPL (interactive mode), respectively?

Options :

6406531484813. ✔ For Script: Number 1: undefined Number 2: 15

For REPL: Number 1: 15 Number 2: 15

6406531484814. ✖ For Script: Number 1: 8 Number 2: 15

For REPL: Number 1: 15 Number 2: 15

6406531484815. ✖ For Script: Number 1: undefined Number 2: 15

For REPL: Number 1: 8 Number 2: 15

6406531484816. ✖ For Script: Number 1: 8 Number 2: 15

For REPL: Number 1: 8 Number 2: 15

Question Number : 172 Question Id : 640653445626 Question Type : MCQ Is Question

Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4.5

Question Label : Multiple Choice Question

Consider the following application with markup index.html and script app.js.

index.html:

```
<body>
    <div id="app"><router-view></router-view></div>
    <script
src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script
src="https://unpkg.com/vue-router@2.0.0/dist/vue-router.js"></script>
    <script src="app.js"></script>
  </body>
```

app.js:

```
const Header = {
  template: `<div>This is header <router-view /> </div>`,
}
const Error = {
  template: '<div> 404 Page not found</div>',
}
const Profile = {
  template: `<div> Welcome to profile page</div>`,
}
const routes = [
  {
    path: '/',
    component: Header,
    children: [
      { path: 'profile', component: Profile },
      { path: '*', component: Error },
    ],
  },
]
const router = new VueRouter({
  routes,
})
new Vue({
  el: '#app',
  router,
})
```

Suppose the application is running on http://127.0.0.1:8080. What will be rendered for the router-view component in Header component if the user visit the URL http://127.0.0.1:8080/#/profile?

Options :

6406531484849. ✖ This is header

6406531484850. ✖ 404 Page not found

6406531484851. ✔ Welcome to profile page

6406531484852. ✖ None of these

Question Number : 173 Question Id : 640653445628 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4.5

Question Label : Multiple Choice Question

Consider an application with markup index.html and script app.js.

index.html:

```
<body>
    <div id="app">
      <router-view></router-view>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script
src="https://unpkg.com/vue-router@2.0.0/dist/vue-router.js"></script>
    <script src="app.js"></script>
  </body>
```

app.js:

```
const Player = {
  props: {
    run: { type: Number, default: 50 },
  },
  computed: {
    playerName() {
      return this.$route.params.name
    },
  },
  template: `<div>Name: {{playerName}}, Run: {{run}}</div>`,
}

const router = new VueRouter({
  routes: [{ path: '/player/:name', component: Player }],
})

new Vue({
  el: '#app',
  router,
})
```

Suppose the application is running on http://127.0.0.1:8080. What will be rendered inside the router-view for http://127.0.0.1:8080/#/player/rohit?

Options :

6406531484857. ✖ Name: rohit, Run: 0

6406531484858. ✖ Name: rohit, Run:

6406531484859. ✖ Name: , Run: 0

6406531484860. ✔ Name: rohit, Run: 50

Sub-Section Number : 3

Sub-Section Id :	64065363339
Question Shuffling Allowed :	Yes
Is Section Default? :	null

Question Number : 174 Question Id : 640653445614 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

Consider the below Vue application with markup file "index.html" and javascript file "app.js".

index.html:

```
<div id = "app">
     <input v-model = "data" />
    <p> Random Number : {{random_number}} </p>
    <button @click = "do_something1"> Click Me</button>
</div>
<script src = "app.js"> </script>
```

app.js:

```
const a = new Vue({
    el : '#app',
    data : {
        random_number : 15,
        hits : 0,
        data : "",
    },
    methods: {
        do_something1 : function () {
            this.hits+= 1
            if (this.hits % 2) {

                sessionStorage.data = this.data + this.data
            }
            localStorage.data = sessionStorage.data
        }
    },
    mounted : function () {
        try {
            this.random_number = localStorage.data.split("iitm").length+1;
        }
        catch {

        }

        this.data = "suffix" + localStorage.data + "prefix"
    }
})
```

Suppose a user of the application opens the "index.html" file in a browser, and enters the text "iitm" in the input box (after removing the old text, if any). After entering the text, the user clicks the button with the text "Click Me" thrice, and closes the browser, and reopens the app in a new browser window. What will be shown in the input text box, and the placeholder "random_number", respectively?

Options :

6406531484801. ✖ suffixiitmiitmprefix, 3

6406531484802. ✔ suffixiitmiitmprefix, 4

6406531484803. ✖ suffixiitmiitmiitmprefix, 3

6406531484804. ✖ suffixiitmiitmiitmprefix, 4

Question Number : 175 Question Id : 640653445615 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

Consider the below Vuex store definition.

```
const store = new Vuex.Store({
    state : {
        students : [],
        // other state properties
    },
    mutations : {
        filter_rows : function (state, data) {
            // filtering logic
            }
    },

    code1 : {
        fetch_data : function (context) {
            let data_to_be_sent = "nothing";
            fetch(some_url).then(response => response.json()
                            ).then(data => {
                                  data_to_be_sent = data;
                            })
            code2
        }
    }
})
```

Suppose the function named "fetch_data" makes an API call to get the data from the backend, and invokes another function named "filter_rows" after receiving the data to filter the results based on some criteria. Which of the following is the best option to fill the placeholders "code1" and "code2"?

Options :

6406531484805. ✖ Code 1: mutations

Code 2: context.dispatch("filter_rows", data)

6406531484806. ✖ Code 1: actions

Code 2: context.dispatch("filter_rows", data_to_be_sent)

6406531484807. ✖ Code 1: mutations

Code 2: context.commit(“filter_rows”, data)

6406531484808. ✔ Code 1: actions

Code 2: context.commit(“filter_rows”, data_to_be_sent)

Question Number : 176 Question Id : 640653445619 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

Consider the below javascript program, and predict the output, if executed.

```
async function quiz () {
    let x = await new Promise((res, rej) => res(6 || 9));
    console.log("Statement 1")
    let y = await new Promise((res, rej) => res(6 && 9));
    console.log("Statement 2", "gives result as", x + y)
}
quiz()
console.log("Statement 3")
```

Options :

6406531484821. ✔ Statement 3

Statement 1

Statement 2 gives result as 15

6406531484822. ✖ Statement 3

Statement 1

Statement 2 gives result as 18

6406531484823. ✖ Statement 1

Statement 3

Statement 2 gives result as 15

6406531484824. ✖ Statement 1

Statement 3

Statement 2 gives result as 12

Question Number : 177 Question Id : 640653445622 Question Type : MCQ Is Question

Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

Suppose you want to store some data in browser which is relevant only till the browser restarts. Which of the following is most suited for this purpose?

Options :

6406531484833. ✔ Session Storage

6406531484834. ✖ Local Storage

6406531484835. ✖ Both localstorage and sessionstorage

6406531484836. ✖ None of these

Question Number : 178 Question Id : 640653445623 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

What will be the output of the below given JavaScript program, if executed?

```
const createPromise = (condition) => {
  return new Promise((reject, resolve) => {
    if (condition) {
      setTimeout(() => {
        resolve('Promise Resolved')
      }, 2000)
    } else {
      setTimeout(() => {
        reject('Promise Rejected')
      }, 1000)
    }
  })
}

createPromise(true)
  .then((val) => {
    console.log(val)
  })
  .catch(() => {
    console.log('Promise was not resolved')
  })
```

Options :

6406531484837. ✖ Promise Resolved

6406531484838. ✖ Promise Rejected

6406531484839. ✔ Promise was not resolved

6406531484840. ✖ None of these

Question Number : 179 Question Id : 640653445624 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

What will be the output of the below written JavaScript program, if executed?

```
async function Demo(str) {
  let r = await new Promise((res, rej) => {
    res(str)
  })
  return r
}

Demo('Hello').then((val) => {
  console.log(val)
})

console.log('World')
```

Options :

6406531484841. ✖ Hello

World

6406531484842. ✔ World

Hello

6406531484843. ✖ Hello

6406531484844. ✖ World

Question Number : 180 Question Id : 640653445625 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

Consider the following application with markup index.html and script app.js.

index.html:

```
<body>
    <div id="app"><router-view></router-view></div>
    <script
src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script
src="https://unpkg.com/vue-router@2.0.0/dist/vue-router.js"></script>
    <script src="app.js"></script>
  </body>
```

app.js:

```
const Header = {
  template: `<div>This is header <router-view /> </div>`,
}
const Error = {
  template: '<div> 404 Page not found</div>',
}
const Profile = {
  template: `<div> Welcome to profile page</div>`,
}
const routes = [
  {
    path: '/',
    component: Header,
    children: [
      { path: 'profile', component: Profile },
      { path: '*', component: Error },
    ],
  },
]
const router = new VueRouter({
  routes,
})
new Vue({
  el: '#app',
  router,
})
```

Suppose the application is running on http://127.0.0.1:8080. What will be rendered for the router-view component used inside Header component, "if the user visits the URL 'http:127.0.0.1:8080/#/home'?

Options :

6406531484845. ✖ This is header

6406531484846. ✔ 404 Page not found

6406531484847. ✖ Welcome to profile page

6406531484848. ✖ None of these

Question Number : 181 Question Id : 640653445627 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

Consider the application with markup index.html and script app.js.

index.html:

```
<body>
    <div id="app">
      <h1>Welcome to the application</h1>
      <button @click="getMovies">Get Movies</button>
      <router-view></router-view>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script
src="https://unpkg.com/vue-router@2.0.0/dist/vue-router.js"></script>
    <script src="app.js"></script>
</body>
```

app.js:

```
const Movies = {
  template: `<div>
   Movie from {{this.$route.query.startIndex}} to
{{this.$route.query.endIndex}}
  </div>`,
}

const router = new VueRouter({
  routes: [{ path: '/movies', name: 'movies', component: Movies }],
})

new Vue({
  el: '#app',
  router,
  methods: {
    getMovies() {
      this.$router.push({
        name: 'movies',
        query: { startIndex: 1, endIndex: 10 },
      })
    },
  },
})
```

Suppose the application is running on http://127.0.0.1:8080 and let user clicks on the button "Get Movies". What will be rendered inside the "router-view" component?

Options :

6406531484853. ✔ Movie from 1 to 10

6406531484854. ✖ Movie from

6406531484855. ✖ Movie from 0 to 10

6406531484856. ✖ None of these

Sub-Section Number : 4

Sub-Section Id : 64065363340

Question Shuffling Allowed : Yes

Is Section Default? : null

Question Number : 182 Question Id : 640653445618 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 2

Question Label : Multiple Choice Question

Consider the below javascript program, and predict the output of the statements mentioned in the program, if executed in a non-strict environment?

```
function quiz () {
    var a = 10;
    let b = c = 20;
    console.log(b); // Statement 1
}
quiz();
console.log(c); // Statement 2
console.log(a); // Statement 3
```

Options :

6406531484817. ✖ Statement 1: 20

Statement 2: 20

Statement 3: 10

6406531484818. ✔ Statement 1: 20

Statement 2: 20

Statement 3: Reference Error

6406531484819. ✖ Statement 1: 20

Statement 2: Error

Statement 3: Control will not reach till statement 3

6406531484820. ✖ Statement 1: Error

Statement 2: Control will not reach till statement 2

Statement 3: Control will not reach till statement 3

Sub-Section Number :	5
Sub-Section Id :	64065363341
Question Shuffling Allowed :	Yes
Is Section Default? :	null

Question Number : 183 Question Id : 640653445616 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 2 Selectable Option : 0

Question Label : Multiple Select Question

Which of the following statement(s) is/are true regarding REST and GraphQL?

Options :

6406531484809. ✖ GraphQL is always preferred over REST, when it comes to fetching the data.

6406531484810. ✔ GraphQL is helpful in solving 2 general problems - “overfetching” and “underfetching”.

6406531484811. ✔ REST defines GET, PUT and DELETE methods to be idempotent, if implemented correctly.

6406531484812. ✖ All of these

Question Number : 184 Question Id : 640653445620 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 2 Selectable Option : 0

Question Label : Multiple Select Question

Which of the following statements describes the best practices with respect to Vuex?

Options :

6406531484825. ✔ The “getters” property in a Vuex store should be used for read only operations.

6406531484826. ✖ The “actions” in a Vuex store should not be used for any asynchronous operations.

6406531484827. ✖ The “mutations” in a Vuex store should be used to perform asynchronous operations, if required.

6406531484828. ✔ All the components should use “this.$store” to access the properties of a Vuex store, instead of directly using the Vuex store object name.

Question Number : 185 Question Id : 640653445621 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 2 Selectable Option : 0

Question Label : Multiple Select Question

Which of the following statements is/are true regarding Vuex?

Options :

6406531484829. ✖ A component which uses store state cannot have local state.

6406531484830. ✔ A component can have local state, even if it uses store state.

6406531484831. ✔ Getters in vuex is used to create derived state.

6406531484832. ✖ All of these

MLT

Section Id :	64065328985
Section Number :	11
Section type :	Online
Mandatory or Optional :	Mandatory
Number of Questions :	17
Number of Questions to be attempted :	17
Section Marks :	100
Display Number Panel :	Yes
Group All Questions :	No