```
students.stream()
    .filter(s -> s.gpa >= 3.0)
    .map(s -> s.gpa <= 3.8)
    .forEach(s -> System.out.println(s.name));
```

# AppDev2

| | |
|---|---|
| **Section Id :** | 64065353267 |
| **Section Number :** | 11 |
| **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 |
| **Section Negative Marks :** | 0 |
| **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 :** | 640653112624 |
| **Question Shuffling Allowed :** | No |
| **Is Section Default? :** | null |

**Question Number : 172 Question Id : 640653770591 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 II (COMPUTER BASED EXAM)"**

**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 :**

6406532577644. ✔ YES

6406532577645. ✖ NO

| | |
|---|---|
| **Sub-Section Number :** | 2 |
| **Sub-Section Id :** | 640653112625 |
| **Question Shuffling Allowed :** | Yes |
| **Is Section Default? :** | null |

**Question Number : 173 Question Id : 640653770592 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.

```
var first = 30
const obj1 = {
    first: 50,
    second: () => {
        console.log("Value:", this.first);
    }
}

const obj2 = {
    first: 80,
    second: function () {
        console.log("Value:", this.first);
        this.second();
    }
}

obj2.second.call(obj1);
```

What will be the output of the program, if executed in a REPL environment?

**Options :**

6406532577646. ✖
Value: 80
Value: 50

6406532577647. ✖
Value: 80
Value: 30

6406532577648. ✖
Value: 50
Value: 50

6406532577649. ✔
Value: 50
Value: 30

6406532577650. ✖
Value: 30
Value: 30

**Question Number : 174 Question Id : 640653770593 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 code.

```
function isValidAge(age) {
    return !(age > 18)
}

const validAgeParams = {
    voter_id: 123,
    constituency: 'New Delhi'
}

const teenageParams = {
    age_criteria: '<18',
}

let some_value;

console.log(
    isValidAge(some_value) ? {
        'name': 'ABC',
        ...(!isValidAge(19) && validAgeParams)
    } : {
        'name': 'XYZ',
        ...(isValidAge(19) && teenageParams)
    }
);
```

What will be the output of the above program, assuming the value of the variable “some_value” is less than 18?

**Options :**

6406532577651. ✖

```
{
   name: 'ABC'
}
```

6406532577652. ✔

```
{
   name: 'ABC',
   voter_id: 123,
   constituency: 'New Delhi'
}
```

6406532577653. ✖

```
{
   name: 'XYZ'
}
```

6406532577654. ✖

```
{
   name: 'ABC',
   age_criteria: '<18'
}
```

**Question Number : 175 Question Id : 640653770594 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.

```
const promise1 = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve('First promise resolved');
    }, 1000);
});

promise1
    .then((result) => {
        console.log(result);
        return new Promise((resolve, reject) => {
            setTimeout(() => {
                resolve('Second promise resolved');
            }, 3000);
        });
    })
    .then((result) => {
        console.log(result);
        return new Promise((resolve, reject) => {
            setTimeout(() => {
                reject('Third promise resolved');
            }, 3000);
        });
    })
    .then((result) => {
        console.log(result);
    })
    .catch((error) => {
        console.log("Promise chain interrupted. Reason:", error);
    });
```

What will be the output of the above program, if executed?

**Options :**

6406532577655. ✖ First promise resolved
Second promise resolved
Third promise Resolved

6406532577656. ✖ The program will crash without showing any output

6406532577657. ✔ First promise resolved
Second promise resolved
Promise chain interrupted. Reason: Third promise resolved

6406532577658. ✖

First promise resolved
Second promise resolved
Promise chain interrupted. Reason: Third promise resolved
Third promise Resolved

6406532577659. ✖

First promise resolved
Second promise resolved

**Question Number : 176 Question Id : 640653770601 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 Script embedded in an HTML document.

```
let Obj1 = {model:'AIR101', brand:'AIRBUS', type: 'Passenger'}
let Obj2 = {__proto__: Obj1, wheels: '36', engines: 4}

console.log(Obj2.model)
console.log(Object.keys(Obj2).length)
console.log(Obj1.engines)
```

What will be the output on console, if the HTML document is rendered using a browser?

**Options :**

6406532577688. ✔

```
AIR101
2
undefined
```

6406532577689. ✖

```
undefined
2
4
```

6406532577690. ✖

```
AIR101
5
undefined
```

6406532577691. ✖

```
AIR101
undefined
undefined
```

**Question Number : 177 Question Id : 640653770604 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

Fill in property_name & definition, which can be used in Vuex Store to update the "best_food" state variable with the objects of those food items which have their calorific value less than 300.

```
const store = new Vuex.Store({
    state:{
        foods:[
                {"name":"Sandwich", "calories":100},
                {"name":"Pizza", "calories": 1250},
                {"name":"Chips", "calories": 700}
              ],
        best_food: []
    },
    =property_name=:{
        get_best_food: function(state) {

            ========== definition ============

        }
    },

})
```

**Options :**

property_name : actions,
definition:

```
state.foods.forEach(food => {
                if (food.calories < 300){
                    context.best_food.push(food)
                }
        });
```

6406532577700. ✖

property_name : actions,
definition:

```
state.foods.forEach(food => {
                if (food.calories < 300){
                    state.best_food.push(food)
                }
            });
```

6406532577701. ✖

property_name : mutations,
definition:

```
state.foods.forEach(food => {
                if (food.calories < 300){
                    context.best_food.push(element)
                }
            });
```

6406532577702. ✖

property_name : mutations,
definition:

```
state.foods.forEach(food => {
                if (food.calories < 300){
                    state.best_food.push(element)
                }
            });
```

6406532577703. ✔

**Sub-Section Number :** 3

**Sub-Section Id :** 640653112626

**Question Shuffling Allowed :** Yes

**Is Section Default? :** null

**Question Number : 178 Question Id : 640653770602 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 following JavaScript code snippet.

```
// Code Snippet 1
sessionStorage.setItem('username', 'course_user');
let storedUsername = sessionStorage.getItem('username');

// Code Snippet 2
sessionStorage.removeItem('username');
let removedUsername = sessionStorage.getItem('username');

// Code Snippet 3
sessionStorage.clear();
let clearedStorage = sessionStorage.username;
```

What will be the values of 'storedUsername', 'removedUsername', and 'clearedStorage' after the execution of the above code snippets?

**Options :**

6406532577692. ✖ storedUsername: 'course_user', removedUsername: null, clearedStorage: null

6406532577693. ✔ storedUsername: 'course_user', removedUsername: null, clearedStorage: undefined

6406532577694. ✖ storedUsername: 'course_user', removedUsername: undefined, clearedStorage: null

6406532577695. ✖ storedUsername: 'course_user', removedUsername: undefined, clearedStorage: undefined

**Sub-Section Number :** 4

**Sub-Section Id :** 640653112627

**Question Shuffling Allowed :** Yes

**Is Section Default? :** null

**Question Number : 179 Question Id : 640653770595 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.

```
const promise1 = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve('First promise resolved');
    }, 1000);
});

promise1
    .then((result) => {
        console.log(result);
        return new Promise((resolve, reject) => {
            setTimeout(() => {
                resolve('Second promise resolved');
            }, 3000);
        });
    })
    .then((result) => {
        console.log(result);
        return new Promise((resolve, reject) => {
            setTimeout(() => {
                resolve('Third promise resolved');
            }, 3000);
        });
    })
    .then((result) => {
        console.log(result);
    })
    .catch((error) => {
        console.log("Promise chain interrupted. Reason:", error);
    });
```

Considering the asynchronous nature of promises in the above JavaScript program, what is the minimum time (in seconds) it will take for the entire execution to complete?

**Options :**

6406532577660. ✖ The program will never end

6406532577661. ✖ 0 second

6406532577662. ✖ 1 second

6406532577663. ✔ 7 seconds

6406532577664. ✖ 9 seconds

**Question Number : 180 Question Id : 640653770607 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 Vue application with markup “index.html” and JavaScript file “app.js”.

index.html:

```
<div id = "app">
    <input v-model = "course" @input = "do_something">
    <p> {{role}} </p>
</div>
```

app.js:

```
new Vue({
    el : "#app",
    data : {
        course : "#app",
        role : "user",
    },

    mounted () {
        try {
            this.course = localStorage.getItem("course").split(" ")[0];
            this.role = localStorage.getItem("course").split(" ")[1];
            localStorage.setItem("course",
                localStorage.getItem("course").split(" ")[0] + " " +
this.course);
        }
        catch {
            this.course = "MAD_II";
            this.role = "admin";
        }
    },

    methods : {
        do_something() {
            localStorage.setItem("course", this.course);
            localStorage.setItem("role", this.role);
        }
    }
})
```

Suppose you open “index.html” file in a browser, and type the text “App Dev” in the text box shown (after removing the previous text, if any), and hard refresh the page twice, without clicking anywhere. What will be the value shown in the text box, and the “age” placeholder, respectively?

**Options :**

6406532577712. ✖ MAD_II, admin

6406532577713. ✖ App, Dev

6406532577714. ✔ App, App

6406532577715. ✖ Dev, App

**Sub-Section Number :** 5

**Sub-Section Id :** 640653112628

**Question Shuffling Allowed :** Yes

**Is Section Default? :** null

**Question Number : 181 Question Id : 640653770596 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 2 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Which of the following statement(s) is/are correct regarding web storage APIs?

**Options :**

6406532577665. ✔ Web Storage APIs provide a way to store key-value pairs persistently on the client side.

6406532577666. ✔ Web Storage APIs include “localStorage” and “sessionStorage”.

6406532577667. ✖ Data stored in “localStorage” is automatically cleared when the browser session ends.

6406532577668. ✖ The “sessionStorage” allows data to persist across browser sessions.

**Question Number : 182 Question Id : 640653770598 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 2 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Which of the following is a valid function signature for an action function in Vuex?

**Options :**

6406532577673. ✔ async actionFunction ( context ) { }

6406532577674. ✔ async actionFunction ( { state, commit }, payload ) { }

6406532577675. ✖ async actionFunction ( context, payload1, payload2 ) { }

6406532577676. ✖ async actionFunction ( { context, commit } ) { }

**Question Number : 183 Question Id : 640653770600 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 2 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Select the statements that incorrectly describe characteristics or practices related to RESTful APIs.

**Options :**

6406532577683. ✖ Stateless communication

6406532577684. ✔ Use of SOAP for data exchange

6406532577685. ✖ Emphasis on nouns (resources) in URIs

6406532577686. ✔ Strict requirement for XML as the data format

6406532577687. ✖ Utilization of standard HTTP methods

| | |
|---|---|
| **Sub-Section Number :** | 6 |
| **Sub-Section Id :** | 640653112629 |
| **Question Shuffling Allowed :** | Yes |
| **Is Section Default? :** | null |

**Question Number : 184 Question Id : 640653770599 Question Type : MSQ Is Question**

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

**Correct Marks : 3 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Which of the following statement(s) is/are true regarding Single Page Applications (SPA) and Progressive Web Apps (PWA)?

**Options :**

6406532577677. ✖ SPAs load entire web pages from the server for each user interaction.

6406532577678. ✔ PWAs can be installed on a user's device and accessed from the home screen.

6406532577679. ✔ SPAs often rely on client-side routing to update the content dynamically.

6406532577680. ✖ PWAs always require an internet connection to function.

6406532577681. ✔ SPAs may result in faster user experiences as they avoid full-page reloads.

6406532577682. ✔ PWAs use service workers to enable offline capabilities.

**Question Number : 185 Question Id : 640653770603 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Identify the correct statement(s) about the behavior of promise chains in JavaScript.

**Options :**

6406532577696. ✖ A "finally" block always comes at the end of the promise chain.

6406532577697. ✖ Every "catch" block must always be preceded by a "then" block.

6406532577698. ✔ A promise chain may consist of a number of "then" blocks.

6406532577699. ✔ The "finally" block always gets executed, irrespective of the promise outcome.

**Question Number : 186 Question Id : 640653770605 Question Type : MSQ Is Question**

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

**Correct Marks : 3 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Consider the below Vue class binding.

Script.js:

```
var app = new Vue({
    el: '#app',
    data : {
        classObj : {
            is_active: true,
            completed: false
        },
        myClass: 'is_active',
        status: true
    }
})
```

Code 1:

```
<div :class="classObj"></div>
```

Code 2:

```
<div :class="[status ? myClass : '']"></div>
```

Which of the following statements is/are true, assuming both code 1 and code 2 are part of app object?

**Options :**

6406532577704. ✔ The code snippet 1 will render the div element with class “completed”, if the “completed” property of “classObj” is set to true.

6406532577705. ✖ The code snippet 2 will render the div element with class “completed”, if the data variable “status” is set to true.

6406532577706. ✔ Both the code snippets will render same HTML

6406532577707. ✖ Both the code snippets will render different HTML

| | |
|---|---|
| **Sub-Section Number :** | 7 |
| **Sub-Section Id :** | 640653112630 |
| **Question Shuffling Allowed :** | Yes |
| **Is Section Default? :** | null |

**Question Number : 187 Question Id : 640653770597 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 4.5 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Consider the below Vue component template and script definitions that use Vue router.

Template:

```
<template>
  <div>
    <h1>{{ pageTitle }}</h1>
    <router-link to="/home" v-if="showHomeLink">Home</router-link>
    <router-link to="/about" v-if="showAboutLink">About</router-link>
    <router-view></router-view>
  </div>
</template>
```

Script:

```
<script>
export default {
  name: 'App',
  data() {
    return {
      pageTitle: 'Vue Router Demo',
      showHomeLink: true,
      showAboutLink: false,
    };
  },
  watch: {
  '$route.path'() {
    if (this.$route.path === '/home') {
      this.showAboutLink = true;
      this.showHomeLink = false;
      this.pageTitle = 'Home Page';
    } else if (this.$route.path === '/about') {
      this.showAboutLink = false;
      this.showHomeLink = true;
      this.pageTitle = 'About Page';
    } else {
      this.pageTitle = 'Vue Router Demo';
      }
    },
  },
};
</script>
```

Assuming that the corresponding routes are properly configured, what does this component structure accomplish?

**Options :**

6406532577669. ✖ It displays a web page with a heading title "Vue Router Demo" and two navigation links to "Home" and "About."

6406532577670. ✔ It dynamically updates the heading title based on the route and conditionally shows navigation links to "Home" and "About."

6406532577671. ✖ It has a bug and will throw a runtime error.

6406532577672. ✔ The heading title is populated with the value “Vue Router Demo” when the user opens the app for the first time.

**Question Number : 188 Question Id : 640653770606 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 4.5 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Consider the below application with markup “index.html” and JavaScript file “app.js”.

Filename: index.html

```
<body>
    <div id="app"></div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script src="https://unpkg.com/vue-router@3.0.0/dist/vue-router.js"></script>
    <script src="app.js"></script>
</body>
```

Filename: app.js

```
const records = {
                  1: {albums: "Heartrisers", songs: 20, genre: 4},
                  2: {albums: "Pianobind", songs: 13, genre: 5}
                }

const notFound = { template: `<h1>Unknown record!</h1>` }
const myRecords = {
    template: `<h1>{{record.albums}} has {{record.songs}} songs across
               {{record.genre}} genres.</h1>`,
    data() {
        return {record: records[this.$route.params.id]}
        },
    }

const router = new VueRouter({
routes: [
    { path: '/records/:id', component: myRecords },
    { path: '*', component: notFound },
],
})

new Vue({
el: '#app',
template: '<div><router-view /></div>',
router,
})
```

Suppose the application is running on “http://127.0.0.1:8080”, select the correct option(s)?

**Options :**

6406532577708. ✖ For the endpoint, http://127.0.0.1:8080/records/1, The browser will render:

**Heartrisers has 20 songs across 4 genres.**

6406532577709. ✔ For the endpoint, http://127.0.0.1:8080/#/records/2, The browser will render:

**Pianobind has 13 songs across 5 genres.**

6406532577710. ✔ For the endpoint, http://127.0.0.1:8080/#/records, The browser will render:

**Unknown record!**

6406532577711. ✖ For the endpoint, http://127.0.0.1:8080/#/records/3, The browser will render:

**Unknown record!**

## MLT

| | |
|---|---|
| **Section Id :** | 64065353268 |
| **Section Number :** | 12 |
| **Section type :** | Online |
| **Mandatory or Optional :** | Mandatory |
| **Number of Questions :** | 12 |
| **Number of Questions to be attempted :** | 12 |
| **Section Marks :** | 50 |
| **Display Number Panel :** | Yes |
| **Section Negative Marks :** | 0 |
| **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 :** | 640653112631 |
| **Question Shuffling Allowed :** | No |
| **Is Section Default? :** | null |