END TERM REVISION

Q1: Consider the following program,

```
x = [1, 2, 3, 4, 5, 6]
y = [...x, 7, 8, 9]
z = y.filter((x) => x % 2 == 0) // 2, 4, 6, 8
     .map((i) => i * i) // 4, 16, 36, 64
     .reduce((a, i) => a + i, (a = 0)) // a= 0, i=4, a=4, 16,
console.log(z)
```

What will be the output of the program?

A. 100
B. 120
C. 140
D. 160

Answer: B

Q2: Consider the below Vue app.

index.html:

```
<div id = "app">{{message}}</div>
<script src = "app.js"> </script>
```

app.js:

```
const dataObj = {
  message: 'Welcome',
}

const optObject = {
  el: '#app',
  data: dataObj,
}
```

```
const app = new Vue(optObject)
app.newMessage = 'Welcome to iitm online degree'
```

What will be rendered by the browser?

A. Welcome
B. Welcome to iitm online degree
C. App with throw an error “property or method ‘newMessage’ is not defined on the instance” and will show a blank page.
D. None of the above

Answer: C

Q3: Which of the following is/are correct regarding the E-Tag and If-Match? // if-None-match

A. E-Tag is a response header to validate the current version of a resource.
B. E-Tag is a string of ASCII characters placed between double quotes.
C. For get and head request, If-Match request header is sent to get the resource only if E-Tag value matches.
D. For get and head request, If-Match request header is sent to get the resource only if E-Tag value does not match.

Answer: A, B and C

Q4: Consider the following Vue application with markup index.html and JavaScript file app.js

index.html

```
<div id="app" style="color: white"
v-bind:style="divStyle">{{message}}</div>
```

app.js

```
const dataObj = {
```

```
  message: 'IITM online',
  divStyle: {
    backgroundColor: 'red',
    padding: '20px',
    fontSize: '2em',
  },
}
const optObject = {
  el: '#app',
  data: dataObj,
}

const app = new Vue(optObject)
```

What will be the color, background color and font-size of the div with id “app”?

A. White, red, 2em
B. Black, white, 1em
C. white, white, 1 em
D. Black, red, 2em

Answer: A

Q5: Consider the following Vue application with markup index.html and JavaScript file app.js.

index.html

```
 <div id="app">
     <p directive1>{{message}}</p>
     <button directive2>click</button>
 </div>

app.js

const app = new Vue({
  el: '#app',
  data: {
    message: 'Hello',
    seen: true,
```

```
  },
  methods: {
    toggleSeen() {
      this.seen = !this.seen
    },
  },
})
```

If you want to toggle between the presence of p element on click of the button element, what are the possible value of directive1 and directive2 and their respective values?

A. v-if=”seen”, v-on:click=”toggleSeen”
B. v-if:”seen”, v-on:click:”toggleSeen”
C. v-if=”seen”, @click=”toggleSeen”
D. v-if:”seen”, @:click:”toggleSeen”

Answer: A and C

Q6: Consider the following Vue application with markup index.html and JavaScript file app.js,

index.html:

```
<div id="app">
      <h4>total payable amount is {{totalPayableAmount}}</h4>
</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    principal: 0,
    annualInterestRate: 0,
    duration: 0,
    totalPayableAmount: 0,
  },
  computed: {
```

```
    simpleInterest() {
      return (this.principal * this.annualInterestRate * this.duration) /
100
    },
  },
})


appData = [
  [2000, 10, 2],   // 400 = 8200
  [3000, 20, 3], // 1800 = 7800
  [5000, 30, 4], // 6000
]

let handler = setInterval(() => {
  data = appData.pop()
  app.principal = data[0]
  app.annualInterestRate = data[1]
  app.duration = data[2]
  app.totalPayableAmount += app.simpleInterest
}, 1000)
```

What will be rendered by the browser after 4 seconds?

A. total payable amount is 6000
B. total payable amount is 8200
C. total payable amount is 1800
D. total payable amount is 7800

Answer: B

Q7: Consider the following Vue application with markup index.html and JavaScript file app.js,

index.html:

```
<div id="app">y: {{y}}</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    x: 0,
    y: 0,
  },
  watch: {
// p-q = 1
    x(p, q) {
      if (p > q && p > 10) {
        this.y = 1
      }
    },
  },
})

for (let i = 0; i < 20; i++) {
  app.x++
}
```

What will be rendered by the browser?

A. 0
B. y: 1
C. 20
D. None of the above

Answer: B

Q8: Consider the following Vue application with markup index.html and JavaScript file app.js,

index.html:

```
<div id="app">y: {{y}}</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    x: 20,
    y: 40,
  },
  beforeCreate() {
    console.log(this.x)
  },
})
```

What will be logged on console and rendered on screen, respectively?

A. undefined, 40
B. undefined, 20
C. 20, 40
D. 40, 40

Answer: A

Q9: Consider the following Vue application with markup index.html and JavaScript file app.js,

index.html:

```
<div id="app">
      <comp1></comp1>
 </div>
<script src = "app.js"> </script>
```

app.js:

```
const comp1 = {
  template: '<h4> This is {{name}}</h4>',
  data: {
    name: 'component 1',
  },
}
```

```
const app = new Vue({
  el: '#app',
  components: {
    comp1,
  },
})
```

What will be rendered by the browser?

A. This is
B. This is component 1
C. No text will be shown on the browser
D. None of the above

Answer: A

Q10: Consider the following files are present inside the same directory,

index.html:

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width">
   <title>repl.it</title>
   <link href="style.css" rel="stylesheet" type="text/css" />
 </head>
 <body>
 </body>
<script src="app.js"></script>
</html>
```

app.js:

```
const a = fetch("sample.txt")

a.then(r => {
    if (!r.ok){
```

```
        throw new Error("HTTP status code: " + r.status);
    }
    return r.text();
}).then(d => {
    console.log("Got the data !!", d);
}).catch(e => {
    console.log(e);
})
```

sample.txt:

```
Hello World !!
```

What will be shown on the browser console if index.html is rendered with the help of a browser? If there is no network error.

A. Got the data !! Hello World !!
B. Hello World !!
C. Got the data !!
D. Error: HTTP status code: 404

Answer: A

Q11: Consider the following JavaScript code.

```
let x = 2,
  n = 3,
  k = 4

const Promise1 = new Promise((resolved, rejected) => {
  if (k < n) {
    resolved(x)
  } else {
    rejected('Bad Promise')
  }
})
```

```
Promise1.then((x) => {
  return x * x
}).then((x) => {
    console.log(x)
  })
  .catch((err) => {
    console.log(err)
  })

promise.then(()=>{}, ()=>{}) != promise.then(()=>{}).then(()=>{})
```

What will be logged on the console?

A. 2
B. 4
C. 8
D. Bad Promise

Answer: D

Q12: Consider the following markup index.html and JavaScript file app.js for an application,

Index.html:

```
<div id="app">
      <named-slot v-slot:header="slotProp">
        Name: {{slotProp.user.name}}, District:
        {{slotProp.user.dist}}</named-slot
      >
</div>
<script src="app.js"></script>
```

app.js:

```
const namedSlot = {
  template: `
```

```
    <div>
        <slot name = "header" :user="currentUser" city=''></slot>
    </div>
    `,
  data() {
    return {
      users: [
        {
          user_id: 1,
          name: 'Narendra',
          dist: 'Ballia',
        },
        {
          user_id: 2,
          name: 'Abhisek',
          dist: 'Delhi',
        },
      ],
      currentUser: null,
    }
  },
  props: ['current_user_id'],
  created() {
    current_user = sessionStorage.getItem('userId')
    current_user_id = current_user ? current_user : 2
    this.currentUser = this.users.find(
      (user) => user.user_id == current_user_id
    )
    sessionStorage.setItem('userId', current_user == 1 ? 2 : 1)
  },
}

const app = new Vue({
  el: '#app',
  components: {
    'named-slot': namedSlot,
  },
})
```

Suppose the application is running on port 8080 what will be rendered by the browser when the page index.html loads on the screen for the first time (without refreshing the page)?

A. Name: Narendra, District: Ballia
B. Name: Abhisek, District: Delhi
C. Nothing will be shown on the screen
D. None of the above

Answer: B

Q13: Consider the following markup index.html and JavaScript app.js for a Vue application.

app.js:

```
const player = {
  template: `<div>
              <h1> Name: {{ name }}</h1>
    <router-view />
  </div>`,
  props: ['name'],
}

const test = {
  template: '<div><h1> Test Runs: {{ runs }} </h1></div>',
  data() {
    return { runs: 5000 }
  },
}

const oneDay = { template: '<div><h1> Oneday Runs: {{ runs }}
</h1></div>',
  data() {
    return { runs: 10000 }
  },
}
const routes = [
```

```
  {
    path: '/player/:name',
    component: player,
    children: [
      { path: '', component: oneDay },
      { path: 'test', component: test },
      { path: 'onday', component: oneDay },
        ],
    props: true,
  },
  { path: '*', component: test },
]
const router = new VueRouter({
  routes,
  base: '/',
})

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

index.html:

```
<div id="app">
   <router-view></router-view>
</div>
<script src="app.js"></script>
```

Suppose the application is running on http://localhost:8080, and user visits the URL "http://localhost:8080/#/player/rohit". What will be rendered by the browser?

A. Name: Rohit
   Oneday Runs: 10000

B. Test Runs: 5000

C. Name: Rohit
Test Runs: 5000

D. Oneday Runs: 10000

Answer. A

Q14: Consider the app.js and index.html given in the Question No.13.

Suppose the application is running on http://localhost:8080, and the user visits the URL "http://localhost:8080/#/player/rohit/test". What will be rendered by the browser?

A. Name: Rohit
Oneday Runs: 10000

B. Test Runs: 5000

C. Name: Rohit
Test Runs: 5000

D. Oneday Runs: 10000

Answer: C

Q15: Consider the app.js and index.html given in the Question No.13.

Suppose the application is running on http://localhost:8080, and the user visit the URL "http://localhost:8080/#/player/rohit/t20". What will be rendered by the browser?

A. Name: Rohit
Oneday Runs: 10000

B. Test Runs: 5000

C. Name: Rohit
Test Runs: 5000

D. Oneday Runs: 10000

Answer. B

Q16: What will be the output of the following program?

```
const obj = {
  name: 'Rohit',
  arrowFunction: null,
  normalfunction: function () {
    this.arrowFunction = () => {
      console.log(this.name)
    }
  },
}

obj.normalfunction()
obj.arrowFunction()
```

A. Rohit
B. Mohit
C. undefined
D. Syntax Error

Answer: A

Global Execution Context → Function execution context (Normal) → Function execution context (Arrow)

Q17: Consider the following JavaScript program.

[‘o’, ‘r’, ‘a’, ‘n” ‘g’, ‘e’]

name.split() → [‘o’, ‘r’, ‘a’, ‘n’, ‘g’, ‘e’]
name.split().reverse() → [‘e’, ‘g’, ‘n’, ‘a’, ‘r’, ‘o’]

Last element after 1 second → o
Last element after 2 seconds → r
Last element after 3 seconds → a
Last element after 4 seconds → n

```
let startNamePrinter = (name) => {
  let x = name.split('').reverse()
  let handler = setInterval(() => {
    let y = x.pop()
    console.log(y)
```

```
  }, 1000)

  setTimeout(() => {
    clearInterval(handler)
  }, (name.length + 1) * 1000)
}

startNamePrinter('orange')
```

What will be the last value shown on the console after 4 seconds? [MCQ : 3 points]

A. r
B. a
C. n
D. o

Answer: C

Q18: Consider the following JavaScript program, and predict the output if executed,

Yet to reach
Reached
Pass

```
let start = 5;
function check() {
   return new Promise((res, rej) => {
   let a1 = setInterval(() => {
         start++;
         if (start === 7) {
             console.log("Reached");
             clearInterval(a1);
             res("Pass");
             rej("Fail");
         }
         else {
             console.log("Yet to Reach");
         }

     }, 500);
 })
}
```

```
check().then(
   pass => console.log(pass)
   ).catch(
       fail => console.log(fail)
   );
```

A. Yet to Reach
   Reached
   Pass

B. Yet to Reach
   Reached
   Pass
   Fail

C. Yet to Reach
   Reached
   Fail

D. Yet to Reach
   Yet to Reach
   Reached
   Pass
   Fail

Answer: A

Q19: In order to ensure that items entered to the toDoList persist after screen refresh by adding it to localstorage, choose the code that can be placed in *code1* as marked below.

index.html:

```
<input  v-model="newData"
              placeholder="Enter your List"/>
      <button @click="addTodoList">Add</button>
```

app.js:

```
data:{
       title:"To Do List",
       newData:'',
       todoList:new Array(),
     },
 methods:{
       addTodoList(){
         this.todoList.push(this.newData)
         code1
       }
     }
```

A. localStorage.setItem(STORAGE_KEY,JSON.stringify(this.todoList))
B. localStorage.getItem(STORAGE_KEY,JSON.stringify(this.todoList))
C. localStorage.setItem(STORAGE_KEY,JSON.parse(this.todoList))
D. localStorage.getItem(STORAGE_KEY,JSON.parse(this.todoList))

Answer: A

Q20: Consider the following markup index.html and JavaScript file app.js for an application,

index.html:

```
<div id="app" v-bind:style="mode === 'dark' ? dark: normal">
      <button @click="changeMode('dark')">Dark</button>
      <button @click="changeMode('normal')">Normal</button>
      <div style="margin-top: 20px">Hello IITM</div>
    </div>
<script src="app.js"></script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
```

```
    dark: {
      backgroundColor: 'black',
      color: 'white',
    },
    normal: {
      backgroundColor: 'red',
      color: 'white',
    },
    mode: null,
  },
  methods: {
    changeMode(mode) {
      sessionStorage.setItem('mode', mode)
      this.mode = mode
    },
  },
  created() {
    websiteMode = sessionStorage.getItem('mode')
    this.mode = websiteMode ? websiteMode : 'dark'
  },
})
```

Suppose the application is running on http://localhost:8080. If a user visit the page “index.html” and click on the button normal, and then close the tab in which the application is open and then again visit the website, what will be the background color of div with ID app?

A. black
B. red
C. white
D. None of the above

Answer: A

Q21: Fill in ***code1***, inside a mutation handler, which can be used to reduce the price of each of the products in the state by a certain discount.

app.js:

```
const store = new Vuex.Store({
    state: {
      discount:10,
      products:[
        {name:'apple',
        price:120
        },
        {
          name:'banana',
          price:60
        }
      ]
    },
    mutations: {
        reducePrice:state=>{
            Code 1

        }
      }
  })
```

A. this.$store.state.products.forEach(product => {
product.price -= (state.discount)/100*product.price;})

B. state.products.forEach(product => {
product.price -= (state.discount)/100*product.price;})

C. commit.products.forEach(product => {
product.price -= (state.discount)/100*product.price;})

D. dispatch.products.forEach(product => {
product.price -= (state.discount)/100*product.price;})

Answer: B

Q22: Which of the following HTTP request methods is cacheable in general?

A. GET
B. POST
C. PUT
D. All of the above

Answer: A

Actual server ⇒ Reverse proxy

Q23: Consider the flask application app.py and an HTML file index.html,

app.py:

```
from flask import Flask, jsonify
from datetime import datetime
import time

app = Flask(__name__)


@app.route('/')
def home():
    time.sleep(10)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


@app.route('/profile')
def profile():
    time.sleep(30)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


if __name__ == "__main__":
    app.run(threaded=False, debug=True)
```

index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function test() {
        const res1 = fetch('http://127.0.0.1:5000')
        const res2 = fetch('http://127.0.0.1:5000/profile')
        res1
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })

        res2
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })
      }
      test()
    </script>
  </body>
</html>
```

If a user visit index.html at 10:10:10 AM, what will be logged in the console(Approximately)?

A. 20, 50

B. 20, 40
C. 40, 40
D. 50, 50

Answer: A

Q24: Which of the following statements is/are true about Redis?

A. Data is always manipulated in the main computer memory(RAM).
B. It stores data on the disk by default.
C. Data cannot be reconstructed back.
D. Data is stored as key value pairs.

Answer: A and D

Q25: Which of the below is true for Celery?

A. Celery systems can consist of multiple brokers.
B. The celery will automatically retry if connection is lost.
C. Redis and RabbitMQ are the only supported brokers.
D. It has features to monitor and schedule tasks.

Answer: A, B and D

Q26: Which of the following is/are true regarding message broker?

A. They serve as intermediary between applications, which allow senders to issue a message without knowing the state of the receiver.
B. They serve as intermediary between applications, which allow senders to issue a message only after knowing the state of the receiver.
C. Inter application communication is often asynchronous.
D. Inter application communication is often synchronous.

Answer: A and C

Q27: Which of the following statement(s) is/are correct, regarding Webhooks and Web sockets?

A. A Webhook uses HTTP protocol in general.
B. A web socket is used to achieve 2-way communication.
C. A Webhook is primarily used for server-to-server communication.
D. A Webhook generally keeps the connection long alive.

Answer: A, B and C

SysA —> SysB

Q28: Which of the following statement(s) is false regarding polling and long poll?

A. In Polling, a client make repetitive calls to the server at fixed time intervals.
B. Polling requires a persistent connection to the server.
C. In long poll mechanism, the connection is kept alive till the server respond.

Answer: B

Q29: Which of the following statement(s) is/are correct?

A. A Webhook pushes the messages to an application.
B. Webhook pulls the messages from an application, but an API pushes the messages to an application.
C. Webhooks are generally synchronous in nature.
D. API can pull and push the messages from/to an application.

Answer: A, C and D

Q30: What is the role of the message broker?

A. It ensures that the message gets delivered to the correct recipient.
B. In general, if the message is undelivered in the first try, it fails and doesn't retry.
C. It ensures that the messages are processed in LIFO manner.
D. It can send messages to multiple recipients, if required.

Answer: A and D

// Compiles the templates in a javascript code
// Start observing the states of the component
// Created hook
// Vue will mount the component on the desired place
// Mounted hook will run