Software License Details

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 120 s **Maximum memory used**: 1 GiB **Maximum execution file size**: 128 MiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, Operators and Control Statements are covered.

**Problem Description:**

The application manages the details pertaining to Software License. As a part of the entire application, one of the business logic done through Javascript is to validate the serial key of a software - by verifying if it follows a certain pattern.

**Following are the files that contains code snippets.**

|  |  |
| --- | --- |
| **index.html** | HTML for webpage (complete implementation is given for you). You only have to run this. No change needs to be done in this file. |
| **script.js** | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**1.** Required function with partially filled body is already available in the javascript file  
*Hint : Do NOT change the function names.*  
Fill the following functionswith required code

|  |  |
| --- | --- |
| **validate()** | On Clicking 'Validate' button, this function is invoked. Part of this function implementation is given in code template.  Fill your code appropriately for variable initialization, as instructed in code template |
| **validateSerialKey(serialKey)** | This function must check if the serial key of a software passed as parameter matches with the pattern : /^[0-9a-zA-Z]{12}$/.  If so, return **true**. Else, return **false.** |

Sample screenshot 1:  
  
  
  
Sample screenshot 2:

Find Highest Enrollment of Policies

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 120 s **Maximum memory used**: 1 GiB **Maximum execution file size**: 128 MiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, Operators and Control Statements are covered.

**Problem Description:**

The application manages the details pertaining to enrollment of policies. As a part of the entire application, one of the business logic done through Javascript is to validate the policy number - by verifying if it follows a certain pattern.

**Following are the files that contains code snippets.**

|  |  |
| --- | --- |
| **index.html** | HTML for webpage (complete implementation is given for you). You only have to run this. No change needs to be done in this file. |
| **script.js** | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**1.** Required function with partially filled body is already available in the javascript file  
*Hint : Do NOT change the function names.*  
Fill the following functionswith required code

|  |  |
| --- | --- |
| **validate()** | On Clicking 'Submit' button, this function is invoked. Part of this function implementation is given in code template.  Fill your code appropriately for variable initialization, as instructed in code template |
| **validatePolicyNumber(policyNumber)** | This function must check if the 'policyNumber' passed as parameter matches with the specified pattern. If so, return **true**. Else, return **false.** |

Sample screenshot 1:  
  
  
Sample screenshot 2:  
  
  
  
Sample screenshot 3:

Email Validation

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 120 s **Maximum memory used**: 1 GiB **Maximum execution file size**: 128 MiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, Operators and Control Statements are covered.

**Problem Description:**

The application manages the registration for a training program . As a part of the entire application, one of the business logic done through Javascript is to validate the email id of trainees - before going ahead with the registration process.

**Following are the files that contains code snippets.**

|  |  |
| --- | --- |
| **index.html** | HTML for webpage (complete implementation is given for you). You only have to run this. No change needs to be done in this file. |
| **script.js** | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**1.** Required function with partially filled body is already available in the javascript file  
*Hint : Do NOT change the function names.*  
Fill the following functionswith required code

|  |  |
| --- | --- |
| **validate()** | On Clicking 'Register' button, this function is invoked. Part of this function implementation is given in code template.  Fill your code appropriately for variable initialization, as instructed in code template. |
| **validateEmailId(emailId)** | This function must check if the email id of a trainee passed as parameter  has '**@**' and **'.'** .  Example (true): [**test@gmail.com**](mailto:test@gmail.com) Example (false): **sam@gmailcom**   Return **true**if the email has '@' and '.' symbols. Else, return **false**. |

Number Of Days

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 120 s **Maximum memory used**: 1 GiB **Maximum execution file size**: 128 MiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, Operators and Control Statements are covered.

**Problem Description:**

The application manages the details pertaining to Recharge Packs. As a part of the entire application, one of the business logic done through Javascript is to validate the name of the recharge pack - by verifying if it follows a certain pattern.

**Following are the files that contains code snippets.**

|  |  |
| --- | --- |
| **index.html** | HTML for webpage (complete implementation is given for you). You only have to run this. No change needs to be done in this file. |
| **script.js** | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**1.** Required function with partially filled body is already available in the javascript file  
*Hint : Do NOT change the function names.*  
Fill the following functionswith required code

|  |  |
| --- | --- |
| **validate()** | On Clicking 'Submit' button, this function is invoked. Part of this function implementation is given in code template.  Fill your code appropriately for variable initialization, as instructed in code template. |
| **validateRechargePackName** **(rechargePackName)** | This function must check if the recharge pack name passed as parameter matches with the pattern : /^[A-Z]{2}[0-9]{3}$/.  If so, return **true**. Else, return **false.** |

Frequency Calculation

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 120 s **Maximum memory used**: 1 GiB **Maximum execution file size**: 128 MiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, Operators and Control Statements are covered.

**Problem Description:**

The application manages the details pertaining to Satellite Frequency Calculation. As a part of the entire application, one of the business logic done through Javascript is to validate the frequency band name - by verifying if it is any one of the valid bands among H, M, L, U, S, C, X and K.

**Following are the files that contains code snippets.**

|  |  |
| --- | --- |
| **index.html** | HTML for webpage (complete implementation is given for you). You only have to run this. No change needs to be done in this file. |
| **script.js** | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**1.** Required function with partially filled body is already available in the javascript file  
*Hint : Do NOT change the function names.*  
Fill the following functionswith required code

|  |  |
| --- | --- |
| **validate()** | On Clicking 'Submit' button, this function is invoked. Part of this function implementation is given in code template.  Fill your code appropriately for variable initialization, as instructed in code template. |
| **validateFrequencyBand(band)** | This function must check if the frequency band passed as parameter matches with the pattern : /^[H|M|L|U|S|C|X|K]$/.  If so, return **true**. Else, return **false.** |

AC Maintenance Service-V1

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 120 s **Maximum memory used**: 2 GiB **Maximum execution file size**: 2 GiB **Maximum number of processes**: 100000

**AC Maintenance Service-V1**

**Objective:**

To add scripting to existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, conditional Statements and control statements are covered.

**Problem Description:**

Sanjay is an AC mechanic who runs a service center for AC maintenance and repair servicing in the city. His customers will call him and book him for the AC maintenance service. Due to more number phone calls in between the busy working hours, he forgets few bookings and so his customers were disappointed. To solve this inconvenience, Sanjay wants an application to book the appointment for AC Servicing. So that he could have a record over the service bookings.

 As an initiative to this application, develop a web page to get the customer details and the appointment details for AC servicing. On submitting, the appointment details have to be displayed on the same page along with the estimated service amount as shown in the screenshots below

**Following are the files that contain code snippets.**

|  |  |
| --- | --- |
| AcMaintenanceService.html | HTML for webpage (complete implementation is already provided). Do not change this file |
| script.js | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

In script.js, Provide the implementation for the functions as per the requirements mentioned below

|  |  |
| --- | --- |
| **getTotalService ()** | This function should return the number of services selected by the customer |
| **getServiceCost ()** | This function should calculate the service cost ( Refer the table service cost)  For eg: if the customer needs service for Cleaning and Repair, calculate the service cost as below.  Cleaning=500, Repair=2500,   service cost= (500+2500) =>3000   Return the service cost |
| **calculateDiscount (servicecost)** | This function should calculate the discounted service cost.  Provide a discount of 15% on service cost.  eg: If service cost =3000;  discounted service cost= 3000 -(3000\*15/100)  Return the discounted service cost |
| **getYearlyMaintenanceCost()** | This function should return  the yearly maintenance cost based on the yearly maintenance field  If the customer requires a yearly maintenance then return 1500, otherwise return 0 |
| **bookAppointment ()** | On Clicking submit button, this function is invoked.  Invoke getTotalService() function and get the number of services. Then invoke the getServiceCost() function and get the service cost.  If the customer requires more than 2 services, then calculate the discounted service cost by using calculateDiscount(servicecost) function  Get the yearly maintenance cost by invoking **getYearlyMaintenanceCost** function and add that amount along with the service cost.  Display the output (Refer screenshot) in <div> tag with id as “result”  Note: User Math.round() method to format the service cost |

We-Host Server Resellers - Purchase Entry-V1

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 240 s **Maximum memory used**: 2 GiB **Maximum number of processes**: 100000

**We-Host Server Re-sellers - Purchase Entry-V1**

**Objective:**

To add scripting to existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, String, conditional Statements are covered.

**Problem Description:**

We-Host - A new server re-seller in the city wants to design a Purchase Entry form to enter the details of the servers that have been purchased from them. Design a form which will require the person in-charge to make an entry of every purchase made with ease.

The following are the screen-shots pertaining to We-Host Server Re-sellers - Purchase Entry

**Following are the files that contain code snippets.**

|  |  |
| --- | --- |
| WEHOST.html | HTML for webpage (complete implementation is already provided). Do not change this file |
| script.js | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

In script.js, Provide the implementation for the functions as per the requirements mentioned below

|  |  |
| --- | --- |
| getCoreCost (core) | This function should return the core cost based on the chosen core ( Refer the table CPU (core) cost)   Return the core cost |
| getConfigurationCost(config) | This function should return the configuration cost based on the chosen configuration ( Refer the table Configuration cost)   Return the configuration cost |
| calculateTax(totalcost,ptype) | This function should calculate the GST and extra tax for payment using card.  GST should be 12% of total cost(core cost+configuration cost)  . If the customer wish to make the payment through a Debit card / Credit card, then along with GST calculate an extra tax of 2% on the total cost.  For eg:  tax=totalcost+(totalcost\*12/100)  If thepayment type is "card" then calculate 2% extra tax  extraTax=totalcost+(totalcost\*2/100)  totalTax=tax+extraTax  Return the totalTax |
| calculatePurchaseCost () | On Clicking CONFIRM PURCHASE button, this function is invoked.  In this function invoke the getCoreCost and getConfigurationCost functions and capture the core cost and configuration cost. Then add those values to get the total cost.  That is: totalcost =core cost+configuration cost  Then calculate the tax for this total cost by invoking calculateTax(totalcost , type)  Add this tax amount to the total cost.   Display the output (Refer screenshot) in <div> tag with id as “result”  Note: User Math.round() method to format the purchase cost |

Cpu core cost:

|  |  |
| --- | --- |
| CPU (core) | cost |
| 2 cores | 20000 |
| 4 cores | 25000 |
| 6 cores | 30000 |
| 8 cores | 40000 |

Configuration cost:

|  |  |
| --- | --- |
| Configuration | Cost |
| *4 GB RAM , 300 GB SSD-boosted Disk Storage* | 5000 |
| *8 GB RAM , 700 GB SSD-boosted Disk Storage* | 10000 |
| *12 GB RAM , 1 TB  SSD-boosted Disk Storage* | 15000 |

Boat Ride Bill Automation-V1

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1 GiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to the existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, loops, Operators and Control Statements are covered.

**Problem Description:**

SMS Lake View wants to create a web application to calculate the bill amount for a boat ride on SMS Lake. So it has decided to create an application that calculates the bill amount from the given data.

**Following are the files that contain code snippets.**

|  |  |
| --- | --- |
| index.html | HTML for webpage (complete implementation is given for you). You only have to run this. No change needs to be done in this file. |
| script.js | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**In script.js, Provide the implementation for the functions as per the requirements mentioned below**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| getBoatCount(btype,noOfPersons) | This function used to find the boat count based on the boat type and noOfPersons  **Formula:**  boatCount= noOfPersons/seatCount **(refer the table for seatCount based on btype)**  **if (noOfPersons/seatCount) have a remainder greater than 0 then increment boatCount value with 1.**   |  |  | | --- | --- | | **Boat Type** | **Seat Count** | | 2 Seater Boat | 2 | | 4 Seater Boat | 4 | | 8 Seater Boat | 8 | | 15 Seater Boat | 15 |     **Example:**  if btype is “4 Seater Boat” and noOfPersons is 10 then,  boatCount=10/4=2.5  Use parseInt(boatCount) to get the quotient part alone as 2.  In this (10/4) value, remainder is 2. So add 1 with boat count.  boatCount=2+1=3.  Return the boatCount. |
| getBoatPrice(btype,boatCount) | This function calculates the boat price based on the boat type and boatCount  **Formula:**  boatPrice= boatCount\*chargePerHour **(refer the table for chargePerHour based on btype)**   |  |  | | --- | --- | | **Boat Type** | **Charges Per Hour** | | 2 Seater Boat | 240 | | 4 Seater Boat | 260 | | 8 Seater Boat | 560 | | 15 Seater Motor Boat | 990 |     2. Return the boatPrice. |
| calculateBill(boatPrice,duration) | This function calculates the bill for the boat ride based on the boatPrice and duration.  **Formula:**  totalPrice= boatPrice\* duration.  Return the totalPrice |
| bookRide() | On Clicking submit button, this function is invoked.  Invoke getBoatCount function by passing the boat type and number of persons. Get the returned count of the boat.  Then Invoke getBoatPrice by passing the boat type, and boat count. Get the returned boat price value.  Invoke the calculateBill function by passing the boat price, and duration. Get the returned totalPrice.  Display the output (Refer screenshot) in <div> tag with id as “result”. |

Singapore Tourism-V1

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1 GiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to the existing web page and work with JavaScript basic activities. The basic Javascript concepts like DOM, Strings, Arrays and Control Statements are covered.

**Problem Description:**

Singapore Tourism conducts a special tour program in October every year. This year they are celebrating their 30th anniversary. So, they decide to celebrate their anniversary by providing a discount for passenger who selects more than 3 places to visit. They plan to take the passengers on a big tour around Singapore as per their wish. Create a web application and help them to calculate the bill amount based on the places to visit and the number of tickets

**Following are the files that contain code snippets.**

|  |  |
| --- | --- |
| index.html | HTML for webpage (complete implementation is given for you). You only have to run this. No change needs to be done in this file. |
| script.js | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**In script.js, Provide the implementation for the functions as per the requirements mentioned below**

|  |  |
| --- | --- |
| getCount() | This function should return the number of places the passenger selects  Return the count to **calculateCost**function |
| getTotalCost(noOfpersons) | This function should calculate the total cost  **Formula:**  totalCost=costForSelectedPlaces\* noOfpersons **(Refer the cost table)**   |  |  | | --- | --- | | **Place to visit** | **Cost for one ticket per day (Dollars)** | | Places Of Pilgrimage | 350$ | | Places Of Heritage | 430$ | | Hills | 780$ | | Falls | 1200$ | | Beach | 270$ | | Places Of Adventures | 4500$ |     For eg: if the passenger selects the Places Of Pilgrimage and Beach for 2 persons, then it should return the total cost of 1240,  Places Of Pilgrimage=350, Beach=270,  totalCost= (350+270)\*2 =1240  Return the totalCost to **calculateCost**function |
| calculateDiscount(cost) | This function should calculate the discounted cost.  If the passenger selects the more than 3 places, then he gets a discount of 15% on total cost.  Calculate the discount and return the discounted service cost to **calculateCost**function |
| getStayCost(noOfPersons) | This function should return the stay cost  Stay cost for each person per day is 150$  **Formula:**  stayCost=noOfDaysStay\* noOfPersons\*150  If the passenger prefers to stay then get the value of noOfDaysStay, calculate the stayCost and return the stayCost, otherwise return 0 to the **calculateCost**function. |
| disableNoOfDaysStay() | This function should disable or enable the noOfDaysStay field based on passenger's stay preference.  If the passenger prefers not to stay then this method should disable the noOfDaysStay field. Otherwise, this method should enable the noOfDaysStay field. |
| calculateCost() | On Clicking submit button, this function is invoked.  Get the number of selected places by calling the getCount() function. Then find the total cost by using getTotalCost(noOfpersons) function.  If the passenger selects more than 2 places, then calculate the discount for the total cost by using calculateDiscount(cost) function  If the passenger prefers to stay, add the stay cost and the discounted cost by using getStayCost(noOfPersons) function.  Display the output (Refer screenshot) in <div> tag with id as “result” |

Monthly Instalment Estimator-V1

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1 GiB **Maximum number of processes**: 10000

**Objective:**

To add scripting to the existing web page and to work with JavaScript basic activities. Javascript concepts like DOM, Operators and Math object are covered.

**Problem Description**

The prime bank needs a web application for its customers to calculate the equated monthly instalment (EMI) amount based on their loan type, expected loan amount and salary. Help them by developing the emi calculation page.

**Following are the files that contain code snippets.**

|  |  |
| --- | --- |
| index.html | HTML for webpage (complete implementation is given for you). Do not change this file. |
| script.js | Add your code in this file for the functions given. |

**Procedure to complete the exercise**

**In script.js, Provide the implementation for the functions as per the requirements mentioned below**

|  |  |
| --- | --- |
| calculateEMI(income,expectedAmt, tenure,interestRatePerAnnum) | This function should calculate the EMI amount based on the salary, expected loan amount, tenure and interest rate per annum  The EMI is calculated using the following formula:  **EMI = L \* R \*  ((1 + R)^N) / (((1 + R)^N) - 1)**  where L - Expected Loan amount  r - interestRatePerAnnum R - rate of interest per month = (r/100) / 12 N - tenure in months.  Use **Math.pow()**to implement ((1 + R)^N).  **Example:**  If Monthly Income = Rs. 10000 r = 10% and R = (10/100) / 12 N = 24 months Expected Loan Amount = Rs.600000 emi= Rs.27686.96  Use Math.round(emi) to get a rounded emi amount.  Return the emi amount |
| getInterestRate(loanType) | This function should return the rate of interest per annum based on the loan type. (Refer to the table for the rate of interest based on loan type)   |  |  | | --- | --- | | **Loan Type** | **Interest Rate Per Annum** | | Home Loan | 7% | | Personal Loan | 7.8% | | Vehicle Loan | 15% |   Return the interest rate per annum |
| checkEligibility(income,emi) | This function should check the loan eligibility based on the income and emi amount  If 60% of the customer's income is greater than or equal to the calculated emi, then return true. Otherwise, return false.  For eg: Income=20000, Emi=12500  Then 60% of income is 20000\*60/100 =>12000  But emi > 60% income , so return false. |
| availLoan() | On Clicking submit button, this function is invoked.  Invoke getInterestRate function by passing the loan type and get the interest rate per annum.  Then Invoke calculateEMI by passing the expected loan amount, income, tenure and rate of interest rate per annum. Get the returned emi value.  Invoke the checkEligibility function and find whether he is eligible to avail of the expected loan amount.  Based on the returned boolean value, display the output in <div> tag with id as "result" If the return value is true then display "You are eligible to get a loan amount as <<expectedAmt>> and emi per month is <<emi>>" Otherwise, display the message as "You are not eligible"  Use **toFixed()**function to display only two digits after the decimal point in the calculated EMI. |