The Apex Codes I've written in my console to complete my trailhead modules are given below:

ContactsTodayController Apex Class:

```
public class ContactsTodayController {

    @AuraEnabled
    public static List<Contact> getContactsForToday() {

        List<Task> my_tasks = [SELECT Id, Subject, WhoId FROM Task WHERE OwnerId =
:UserInfo.getUserId() AND IsClosed = false AND WhoId != null];
        List<Event> my_events = [SELECT Id, Subject, WhoId FROM Event WHERE OwnerId =
:UserInfo.getUserId() AND StartDateTime >= :Date.today() AND WhoId != null];
        List<Case> my_cases = [SELECT ID, ContactId, Status, Subject FROM Case WHERE OwnerId
= :UserInfo.getUserId() AND IsClosed = false AND ContactId != null];

        Set<Id> contactIds = new Set<Id>();
        for(Task tsk : my_tasks) {
            contactIds.add(tsk.WhoId);
        }
        for(Event evt : my_events) {
            contactIds.add(evt.WhoId);
        }
        for(Case cse : my_cases) {
            contactIds.add(cse.ContactId);
        }

        List<Contact> contacts = [SELECT Id, Name, Phone, Description FROM Contact WHERE Id
IN :contactIds];

        for(Contact c : contacts) {
            c.Description = '';
            for(Task tsk : my_tasks) {
                if(tsk.WhoId == c.Id) {
                    c.Description += 'Because of Task "'+tsk.Subject+'"\n';
                }
            }
            for(Event evt : my_events) {
                if(evt.WhoId == c.Id) {
```

```
                c.Description += 'Because of Event "'+evt.Subject+'"\n';
            }
        }
        for(Case cse : my_cases) {
            if(cse.ContactId == c.Id) {
                c.Description += 'Because of Case "'+cse.Subject+'"\n';
            }
        }
    }

    return contacts;
}

}
```

ContactsTodayControllerTest Apex Class:

```
@IsTest
public class ContactsTodayControllerTest {

    @IsTest
    public static void testGetContactsForToday() {

        Account acct = new Account(
            Name = 'Test Account'
        );
        insert acct;

        Contact c = new Contact(
            AccountId = acct.Id,
            FirstName = 'Test',
            LastName = 'Contact'
        );
        insert c;

        Task tsk = new Task(
            Subject = 'Test Task',
            WhoId = c.Id,
            Status = 'Not Started'
        );
        insert tsk;
        Event evt = new Event(
            Subject = 'Test Event',
            WhoId = c.Id,
            StartDateTime = Date.today().addDays(5),
            EndDateTime = Date.today().addDays(6)
        );
        insert evt;

        Case cse = new Case(
            Subject = 'Test Case',
            ContactId = c.Id
        );
        insert cse;

        List<Contact> contacts = ContactsTodayController.getContactsForToday();
        System.assertEquals(1, contacts.size());
        System.assert(contacts[0].Description.containsIgnoreCase(tsk.Subject));
        System.assert(contacts[0].Description.containsIgnoreCase(evt.Subject));
        System.assert(contacts[0].Description.containsIgnoreCase(cse.Subject));

    }

    @IsTest
    public static void testGetNoContactsForToday() {

        Account acct = new Account(
            Name = 'Test Account'
        );
        insert acct;

        Contact c = new Contact(
            AccountId = acct.Id,
            FirstName = 'Test',
            LastName = 'Contact'
        );
        insert c;

        Task tsk = new Task(
            Subject = 'Test Task',
            WhoId = c.Id,
            Status = 'Completed'
        );
        insert tsk;

        Event evt = new Event(
            Subject = 'Test Event',
            WhoId = c.Id,
            StartDateTime = Date.today().addDays(-6),
            EndDateTime = Date.today().addDays(-5)
        );
        insert evt;

        Case cse = new Case(
            Subject = 'Test Case',
            ContactId = c.Id,
            Status = 'Closed'
        );
        insert cse;

        List<Contact> contacts = ContactsTodayController.getContactsForToday();
        System.assertEquals(0, contacts.size());

    }

}
```

ContactAndLeadSearch Apex Class:

```
public class ContactAndLeadSearch {

    public static List<List<sObject>> searchContactsAndLeads(String name){
         List<List<sObject>> ContactLeadList = [Find :name IN ALL FIELDS RETURNING
Contact(LastName), Lead(LastName)];

        return ContactLeadList;

    }

}
```

StringArrayTest Apex Class:

```
public class StringArrayTest {

    public static List<String> generateStringArray(Integer N){
        List<String> TestList = new List<String>();
        for(Integer i=0;i<N;i++){
            TestList.add('Test ' + i);
            system.debug(TestList[i]);
        }
        return TestList;

    }

}
```

EmailManager Apex Class:

```
public class EmailManager {

    public void sendMail(String address, String subject, String body){
        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
        String[] toAddresses = new String[] {address};
        mail.setToAddresses(toAddresses);
        mail.setSubject(subject);
        mail.setPlainTextBody(body);

        Messaging.SendEmailResult[] results = Messaging.sendEmail(new
Messaging.SingleEmailMessage[] { mail });
        inspectResults(results);
    }

    private static Boolean inspectResults(Messaging.SendEmailResult[] results){
        Boolean sendResult = true;

        for(Messaging.SendEmailResult res : results){
         if(res.isSuccess()){
            System.debug('Email sent successfully');
         }
         else{
            sendResult = false;
            System.debug('The following errors occured: '+res.getErrors());
         }
      }
      return sendResult;
   }

}
```

AccountHandler Apex Class:

```
public class AccountHandler {
   public static Account insertNewAccount(String AccountName){
      try {
         Account newacct = new Account(Name=AccountName);
         insert newacct;
         return newacct;
         } catch (DmlException e) {
            System.debug('A DML exception has occurred: ' + e.getMessage());
            return null;

      }

   }

}
```

ContactSearch Apex Class:

```
public class ContactSearch {
   public static List<Contact> searchForContacts(String lastName, String postalCode){

      List<Contact> Contacts = [Select Id,Name from Contact where LastName =:lastName and
MailingPostalCode =:postalcode];

      return Contacts;
```

```
    }

}
```

NewCaseListController Apex Class:

```
public class NewCaseListController {

    public List<Case> getNewCases(){
        List<Case> filterList = [Select Id, CaseNumber from Case where status = 'New'];
        return filterList;
    }

}
```

RandomContactFactory Apex Class:

```
public class RandomContactFactory {

    public static List<Contact> generateRandomContacts(Integer numcnt, string lastname){
        List<Contact> contacts = new List<Contact>();
        for(Integer i=0;i<numcnt;i++){
            Contact cnt = new Contact(FirstName = 'Test '+i, LastName = lastname);
            contacts.add(cnt);
        }
        return contacts;
    }

}
```

TestRestrictContactByName Apex Class:

```
@isTest
public class TestRestrictContactByName {
    @isTest static void Test_insertupdateContact(){
        Contact cnt = new Contact();
        cnt.LastName = 'INVALIDNAME';
        Test.startTest();
        Database.SaveResult result = Database.insert(cnt, false);
        Test.stopTest();

        System.assert(!result.isSuccess());
        System.assert(result.getErrors().size()>0);
        System.assertEquals('The Last Name "INVALIDNAME" is not allowed for
DML',result.getErrors()[0].getMessage());
    }

}
```

TestVerifyDate Apex Class:

```
@isTest
private class TestVerifyDate {

    @isTest static void Test_CheckDates_case1(){
        Date D = VerifyDate.CheckDates(date.parse('01/01/2020'),date.parse('01/05/2020'));
        System.assertEquals(date.parse('01/05/2020'), D);
    }

    @isTest static void Test_CheckDates_case2(){
        Date D = VerifyDate.CheckDates(date.parse('01/01/2020'),date.parse('05/05/2020'));
        System.assertEquals(date.parse('01/31/2020'), D);

    }

    @isTest static void Test_DateWithin30Days_case1(){
        Boolean flag =
VerifyDate.DateWithin30Days(date.parse('01/01/2020'),date.parse('12/30/2019'));
        System.assertEquals(false, flag);
    }
    @isTest static void Test_DateWithin30Days_case2(){
        Boolean flag =
VerifyDate.DateWithin30Days(date.parse('01/01/2020'),date.parse('02/02/2019'));
        System.assertEquals(false, flag);
    }
    @isTest static void Test_DateWithin30Days_case3(){
        Boolean flag =
VerifyDate.DateWithin30Days(date.parse('01/01/2020'),date.parse('01/15/2020'));
        System.assertEquals(true, flag);
    }

    @isTest static void Test_SetEndOfMonthDate(){
        Date returndate = VerifyDate.SetEndOfMonthDate(date.parse('01/01/2020'));
    }

}
```

VerifyDate Apex Class:

```
public class VerifyDate {

        //method to handle potential checks against two dates
        public static Date CheckDates(Date date1, Date date2) {
                //if date2 is within the next 30 days of date1, use date2.  Otherwise use the end
of the month
                if(DateWithin30Days(date1,date2)) {
                        return date2;
                } else {
                        return SetEndOfMonthDate(date1);
                }
        }

        //method to check if date2 is within the next 30 days of date1
        @TestVisible private static Boolean DateWithin30Days(Date date1, Date date2) {
                //check for date2 being in the past
        if( date2 < date1) { return false; }

        //check that date2 is within (>=) 30 days of date1
        Date date30Days = date1.addDays(30); //create a date 30 days away from date1
                if( date2 >= date30Days ) { return false; }
                else { return true; }
        }

        //method to return the end of the month of a given date
        @TestVisible private static Date SetEndOfMonthDate(Date date1) {
                Integer totalDays = Date.daysInMonth(date1.year(), date1.month());
                Date lastDay = Date.newInstance(date1.year(), date1.month(), totalDays);
                return lastDay;
        }

}
```

EmailMissionSpecialist Apex Class:

```
public class EmailMissionSpecialist {
  // Public method
  public void sendMail(String address, String subject, String body) {
    // Create an email message object
    Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
    String[] toAddresses = new String[] {address};
    mail.setToAddresses(toAddresses);
    mail.setSubject(subject);
    mail.setPlainTextBody(body);
    // Pass this email message to the built-in sendEmail method
    // of the Messaging class
    Messaging.SendEmailResult[] results = Messaging.sendEmail(
                        new Messaging.SingleEmailMessage[] { mail });
    // Call a helper method to inspect the returned results
    inspectResults(results);
  }
  // Helper method
  private static Boolean inspectResults(Messaging.SendEmailResult[] results) {
    Boolean sendResult = true;
    // sendEmail returns an array of result objects.
    // Iterate through the list to inspect results.
    // In this class, the methods send only one email,
    // so we should have only one result.
    for (Messaging.SendEmailResult res : results) {
      if (res.isSuccess()) {
        System.debug('Email sent successfully');
      }
      else {
        sendResult = false;
        System.debug('The following errors occurred: ' + res.getErrors());
      }
    }
    return sendResult;
  }
}
```

AnimalsHttpCalloutMock Apex Class:

```
@isTest
global class AnimalsHttpCalloutMock implements HttpCalloutMock {
    // Implement this interface method
    global HTTPResponse respond(HTTPRequest request) {
        // Create a fake response
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
        response.setBody('{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken",
"mighty moose"]}');
        response.setStatusCode(200);
        return response;
    }
}
```

AnimalsCallouts Apex Class:

```
public class AnimalsCallouts {
    public static HttpResponse makeGetCallout() {
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals');
        request.setMethod('GET');
        HttpResponse response = http.send(request);
        // If the request is successful, parse the JSON response.
        if(response.getStatusCode() == 200) {
            // Deserializes the JSON string into collections of primitive data types.
            Map<String, Object> results = (Map<String, Object>)
JSON.deserializeUntyped(response.getBody());
            // Cast the values in the 'animals' key as a list
            List<Object> animals = (List<Object>) results.get('animals');
            System.debug('Received the following animals:');
            for(Object animal: animals) {
                System.debug(animal);
            }
        }
        return response;
    }
    public static HttpResponse makePostCallout() {
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals');
        request.setMethod('POST');
        request.setHeader('Content-Type', 'application/json;charset=UTF-8');
        request.setBody('{"name":"mighty moose"}');
        HttpResponse response = http.send(request);
        // Parse the JSON response
        if(response.getStatusCode() != 201) {
            System.debug('The status code returned was not expected: ' +
                response.getStatusCode() + ' ' + response.getStatus());
        } else {
            System.debug(response.getBody());
        }
        return response;
    }
}
```

AnimalsCalloutsTest Apex Class:

```
@isTest
private class AnimalsCalloutsTest {
    @isTest static  void testGetCallout() {
        // Create the mock response based on a static resource
        StaticResourceCalloutMock mock = new StaticResourceCalloutMock();
        mock.setStaticResource('GetAnimalResource');
        mock.setStatusCode(200);
        mock.setHeader('Content-Type', 'application/json;charset=UTF-8');
        // Associate the callout with a mock response
        Test.setMock(HttpCalloutMock.class, mock);
        // Call method to test
        HttpResponse result = AnimalsCallouts.makeGetCallout();
        // Verify mock response is not null
        System.assertNotEquals(null,result, 'The callout returned a null response.');
        // Verify status code
        System.assertEquals(200,result.getStatusCode(), 'The status code is not 200.');
        // Verify content type
        System.assertEquals('application/json;charset=UTF-8',
          result.getHeader('Content-Type'),
          'The content type value is not expected.');
        // Verify the array contains 3 items
        Map<String, Object> results = (Map<String, Object>)
            JSON.deserializeUntyped(result.getBody());
        List<Object> animals = (List<Object>) results.get('animals');
        System.assertEquals(3, animals.size(), 'The array should only contain 3 items.');

    }
    @isTest
    static void testPostCallout() {
    // Set mock callout class
    Test.setMock(HttpCalloutMock.class, new AnimalsHttpCalloutMock());
    // This causes a fake response to be sent
    // from the class that implements HttpCalloutMock.
    HttpResponse response = AnimalsCallouts.makePostCallout();
    // Verify that the response received contains fake values
    String contentType = response.getHeader('Content-Type');
    System.assert(contentType == 'application/json');
    String actualValue = response.getBody();
    System.debug(response.getBody());
    String expectedValue = '{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken",
"mighty moose"]}';
    System.assertEquals(expectedValue, actualValue);
    System.assertEquals(200, response.getStatusCode());
}

}
```

LeadProcessorTest Apex Class:

```
@isTest
public class LeadProcessorTest {

        @testSetup
    static void setup() {
        List<Lead> leads = new List<Lead>();
        for(Integer counter=0 ;counter <200;counter++){
            Lead lead = new Lead();
            lead.FirstName ='FirstName';
            lead.LastName ='LastName'+counter;
            lead.Company
 ='demo'+counter;
            leads.add(lead);
        }
        insert leads;
    }

    @isTest static void test() {
        Test.startTest();
        LeadProcessor leadProcessor = new LeadProcessor();
        Id batchId = Database.executeBatch(leadProcessor);
        Test.stopTest();
    }

}
```

LeadProcessor Apex Class:

```
public class LeadProcessor implements Database.Batchable<sObject> {

     public Database.QueryLocator start(Database.BatchableContext bc) {
        // collect the batches of records or objects to be passed to execute
          return Database.getQueryLocator([Select LeadSource From Lead ]);
    }
    public void execute(Database.BatchableContext bc, List<Lead> leads){
         // process each batch of records
            for (Lead Lead : leads) {
                lead.LeadSource = 'Dreamforce';
            }
        update leads;
    }
    public void finish(Database.BatchableContext bc){
      }

}
```

AnimalLocator Apex Class:

```
public class AnimalLocator{
    public static String getAnimalNameById(Integer x){
        Http http = new Http();
        HttpRequest req = new HttpRequest();
        req.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/' + x);
        req.setMethod('GET');
        Map<String, Object> animal= new Map<String, Object>();
        HttpResponse res = http.send(req);
            if (res.getStatusCode() == 200) {
        Map<String, Object> results = (Map<String,
Object>)JSON.deserializeUntyped(res.getBody());
      animal = (Map<String, Object>) results.get('animal');
        }
return (String)animal.get('name');
    }
}
```

AnimalLocatorTest Apex Class:

```
@isTest
private class AnimalLocatorTest{
    @isTest static void AnimalLocatorMock1() {
        Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
        string result = AnimalLocator.getAnimalNameById(3);
        String expectedResult = 'chicken';
        System.assertEquals(result,expectedResult );
    }
}
```

AnimalLocatorMock Apex Class:

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
     // Implement this interface method
    global HTTPResponse respond(HTTPRequest request) {
        // Create a fake response
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
        response.setBody('{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken",
"mighty moose"]}');
        response.setStatusCode(200);
        return response;
    }
}
```

AccountProcessor Apex Class:

```
public class AccountProcessor {

    @future
    public static void countContacts(List<Id> accountIds){
        List<Account> accList = [Select Id, Number_Of_Contacts__c,(Select Id from Contacts) from
Account where Id in :accountIds];
        For(Account acc: accList){

            acc.Number_Of_Contacts__c = acc.Contacts.size();
        }
        update accList;

    }

}
```

AccountProcessorTest Apex Class:

```
@isTest
public class AccountProcessorTest {

    public static testmethod void testAccountProcessor(){
        Account a = new Account();
        a.Name = 'Test Account';
        insert a;

        Contact con = new Contact();
        con.FirstName = 'Binary';
        con.LastName = 'Programming';
        con.AccountId = a.Id;
        insert con;
        List<Id> accListId = new List<Id>();
        accListId.add(a.Id);
        Test.startTest();
        AccountProcessor.countContacts(accListId);
        Test.stopTest();

        Account acc = [Select Number_Of_Contacts__c from Account where Id =: a.Id];
        System.assertEquals(Integer.valueOf(acc.Number_Of_Contacts__c),1);
    }

}
```

DailyLeadProcessor Apex Class:

```
public class DailyLeadProcessor implements Schedulable  {
    Public void execute(SchedulableContext SC){
       List<Lead> LeadObj=[SELECT Id from Lead where LeadSource=null limit 200];
        for(Lead l:LeadObj){
            l.LeadSource='Dreamforce';
            update l;
        }
    }
}
```

DailyLeadProcessorTest Apex Class:

```
@isTest
private class DailyLeadProcessorTest {
            static testMethod void testDailyLeadProcessor() {
                        String CRON_EXP = '0 0 1 * * ?';
                        List<Lead> lList = new List<Lead>();
                for (Integer i = 0; i < 200; i++) {
                                    lList.add(new Lead(LastName='Dreamforce'+i, Company='Test1 Inc.',
Status='Open - Not Contacted'));
                        }
                        insert lList;

                        Test.startTest();
```

```
                    String jobId = System.schedule('DailyLeadProcessor', CRON_EXP, new
DailyLeadProcessor());
          }
}
```

AddPrimaryContact Apex Class:

```
public class AddPrimaryContact implements Queueable
{
   private Contact c;
   private String state;
   public  AddPrimaryContact(Contact c, String state)
   {
      this.c = c;
      this.state = state;
   }
   public void execute(QueueableContext context)
   {
       List<Account> ListAccount = [SELECT ID, Name ,(Select id,FirstName,LastName from
contacts ) FROM ACCOUNT WHERE BillingState = :state LIMIT 200];
       List<Contact> lstContact = new List<Contact>();
       for (Account acc:ListAccount)
       {
             Contact cont = c.clone(false,false,false,false);
             cont.AccountId =  acc.id
;
             lstContact.add( cont );
       }

       if(lstContact.size() >0 )
       {
          insert lstContact;
       }

   }
}
```

AddPrimaryContactTest Apex Class:

```
@isTest
```

```
public class AddPrimaryContactTest
{
    @isTest static void TestList()
    {
        List<Account> Teste = new List <Account>();
        for(Integer i=0;i<50;i++)
        {
            Teste.add(new Account(BillingState = 'CA', name = 'Test'+i));
        }
        for(Integer j=0;j<50;j++)
        {
            Teste.add(new Account(BillingState = 'NY', name = 'Test'+j));
        }
        insert Teste;

        Contact co = new Contact();
        co.FirstName='demo';
        co.LastName ='demo';
        insert co;
        String state = 'CA';

         AddPrimaryContact apc = new AddPrimaryContact(co, state);
         Test.startTest();
           System.enqueueJob(apc);
         Test.stopTest();
     }
 }
```

AwesomeCalculator Apex Class:

```
public class AwesomeCalculator {
    public static Double add(Double x, Double y) {
        calculatorServices.CalculatorImplPort calculator =
            new calculatorServices.CalculatorImplPort();
        return calculator.doAdd(x,y);
    }
}
```

AwesomeCalculatorTest Apex Class:

```
@isTest
private class AwesomeCalculatorTest {
   @isTest static void testCallout() {
      // This causes a fake response to be generated
      Test.setMock(WebServiceMock.class, new CalculatorCalloutMock());
      // Call the method that invokes a callout
      Double x = 1.0;
      Double y = 2.0;
      Double result = AwesomeCalculator.add(x, y);
      // Verify that a fake result is returned
      System.assertEquals(3.0, result);
   }
}
```

ParkService Apex Class:

```
public class ParkService {
   public class byCountryResponse {
      public String[] return_x;
      private String[] return_x_type_info = new String[]{'return','http://parks.services/',null,'0','-
1','false'};
      private String[] apex_schema_type_info = new String[]{'http://parks.services/','false','false'};
      private String[] field_order_type_info = new String[]{'return_x'};
   }
   public class byCountry {
      public String arg0;
      private String[] arg0_type_info = new String[]{'arg0','http://parks.services/',null,'0','1','false'};
      private String[] apex_schema_type_info = new String[]{'http://parks.services/','false','false'};
      private String[] field_order_type_info = new String[]{'arg0'};
   }
   public class ParksImplPort {
      public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/parks';
      public Map<String,String> inputHttpHeaders_x;
      public Map<String,String> outputHttpHeaders_x;
      public String clientCertName_x;
      public String clientCert_x;
      public String clientCertPasswd_x;
      public Integer timeout_x;
      private String[] ns_map_type_info = new String[]{'http://parks.services/', 'ParkService'};
```

```
        public String[] byCountry(String arg0) {
            ParkService.byCountry request_x = new ParkService.byCountry();
            request_x.arg0 = arg0;
            ParkService.byCountryResponse response_x;
            Map<String, ParkService.byCountryResponse> response_map_x = new Map<String,
ParkService.byCountryResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
              '',
              'http://parks.services/',
              'byCountry',
              'http://parks.services/',
              'byCountryResponse',
              'ParkService.byCountryResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
    }
}
```

ParkServiceMock Apex Class:

```
@isTest
global class ParkServiceMock implements WebServiceMock {
   global void doInvoke(
           Object stub,
           Object request,
           Map<String, Object> response,
           String endpoint,
           String soapAction,
           String requestName,
           String responseNS,
           String responseName,
           String responseType) {
        // start - specify the response you want to send
        ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
        response_x.return_x = new List<String>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};
        // end
        response.put('response_x', response_x);
    }
}
```

AsyncParkService Apex Class:

```
public class AsyncParkService {
    public class byCountryResponseFuture extends System.WebServiceCalloutFuture {
        public String[] getValue() {
            ParkService.byCountryResponse response =
(ParkService.byCountryResponse)System.WebServiceCallout.endInvoke(this);
            return response.return_x;
        }
    }
    public class AsyncParksImplPort {
        public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/parks';
        public Map<String,String> inputHttpHeaders_x;
        public String clientCertName_x;
        public Integer timeout_x;
        private String[] ns_map_type_info = new String[]{'http://parks.services/', 'ParkService'};
        public AsyncParkService.byCountryResponseFuture beginByCountry(System.Continuation
continuation,String arg0) {
            ParkService.byCountry request_x = new ParkService.byCountry();
            request_x.arg0 = arg0;
            return (AsyncParkService.byCountryResponseFuture)
System.WebServiceCallout.beginInvoke(
              this,
              request_x,
              AsyncParkService.byCountryResponseFuture.class,
              continuation,
              new String[]{endpoint_x,
              '',
              'http://parks.services/',
              'byCountry',
              'http://parks.services/',
              'byCountryResponse',
              'ParkService.byCountryResponse'}
            );
        }
    }
}
```

ParkServices Apex Class:

```
public class ParkServices {
    public class byCountryResponse {
        public String[] return_x;
        private String[] return_x_type_info = new String[]{'return','http://parks.services/',null,'0','-
1','false'};
        private String[] apex_schema_type_info = new String[]{'http://parks.services/','false','false'};
        private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class byCountry {
        public String arg0;
        private String[] arg0_type_info = new String[]{'arg0','http://parks.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new String[]{'http://parks.services/','false','false'};
        private String[] field_order_type_info = new String[]{'arg0'};
    }
    public class ParksImplPort {
        public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/parks';
        public Map<String,String> inputHttpHeaders_x;
        public Map<String,String> outputHttpHeaders_x;
        public String clientCertName_x;
        public String clientCert_x;
        public String clientCertPasswd_x;
        public Integer timeout_x;
        private String[] ns_map_type_info = new String[]{'http://parks.services/', 'ParkServices'};
        public String[] byCountry(String arg0) {
            ParkServices.byCountry request_x = new ParkServices.byCountry();
            request_x.arg0 = arg0;
            ParkServices.byCountryResponse response_x;
            Map<String, ParkServices.byCountryResponse> response_map_x = new Map<String,
ParkServices.byCountryResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
       '',
       'http://parks.services/',
       'byCountry',
       'http://parks.services/',
       'byCountryResponse',
       'ParkServices.byCountryResponse'}
      );
      response_x = response_map_x.get('response_x');
      return response_x.return_x;
    }
  }
}
```

CaseManager Apex Class:

```
@RestResource(urlMapping='/Cases/*')
global with sharing class CaseManager {
  @HttpGet
  global static Case getCaseById() {
    RestRequest request = RestContext.request;
    // grab the caseId from the end of the URL
    String caseId = request.requestURI.substring(
     request.requestURI.lastIndexOf('/')+1);
    Case result =  [SELECT CaseNumber,Subject,Status,Origin,Priority
            FROM Case
            WHERE Id = :caseId];
    return result;
  }
  @HttpPost
  global static ID createCase(String subject, String status,
    String origin, String priority) {
    Case thisCase = new Case(
      Subject=subject,
      Status=status,
      Origin=origin,
      Priority=priority);
    insert thisCase;
    return thisCase.Id;
  }
  @HttpDelete
  global static void deleteCase() {
    RestRequest request = RestContext.request;
    String caseId = request.requestURI.substring(
      request.requestURI.lastIndexOf('/')+1);
    Case thisCase = [SELECT Id FROM Case WHERE Id = :caseId];
    delete thisCase;
  }
  @HttpPut
  global static ID upsertCase(String subject, String status,
    String origin, String priority, String id) {
    Case thisCase = new Case(
        Id=id,
        Subject=subject,
        Status=status,
        Origin=origin,
        Priority=priority);
    // Match case by Id, if present.
    // Otherwise, create new case.
    upsert thisCase;
    // Return the case ID.
    return thisCase.Id;
  }
  @HttpPatch
  global static ID updateCaseFields() {
    RestRequest request = RestContext.request;
    String caseId = request.requestURI.substring(
      request.requestURI.lastIndexOf('/')+1);
    Case thisCase = [SELECT Id FROM Case WHERE Id = :caseId];
    // Deserialize the JSON string into name-value pairs
    Map<String, Object> params = (Map<String,
Object>)JSON.deserializeUntyped(request.requestbody.tostring());
    // Iterate through each parameter field and value
    for(String fieldName : params.keySet()) {
      // Set the field and value on the Case sObject
      thisCase.put(fieldName, params.get(fieldName));
    }
    update thisCase;
    return thisCase.Id;
  }
}
```

CaseManagerTest Apex Class:

```
@IsTest
private class CaseManagerTest {
  @isTest static void testGetCaseById() {
    Id recordId = createTestRecord();
    // Set up a test request
    RestRequest request = new RestRequest();
    request.requestUri =
      'https://yourInstance.my.salesforce.com/services/apexrest/Cases/'
      + recordId;
    request.httpMethod = 'GET';
    RestContext.request = request;
    // Call the method to test
    Case thisCase = CaseManager.getCaseById();
    // Verify results
    System.assert(thisCase != null);
    System.assertEquals('Test record', thisCase.Subject);
  }
  @isTest static void testCreateCase() {
    // Call the method to test
    ID thisCaseId = CaseManager.createCase(
      'Ferocious chipmunk', 'New', 'Phone', 'Low');
    // Verify results
    System.assert(thisCaseId != null);
    Case thisCase = [SELECT Id,Subject FROM Case WHERE Id=:thisCaseId];
    System.assert(thisCase != null);
    System.assertEquals(thisCase.Subject, 'Ferocious chipmunk');
  }
  @isTest static void testDeleteCase() {
    Id recordId = createTestRecord();
    // Set up a test request
    RestRequest request = new RestRequest();
    request.requestUri =
      'https://yourInstance.my.salesforce.com/services/apexrest/Cases/'
      + recordId;
    request.httpMethod = 'DELETE';
    RestContext.request = request;
    // Call the method to test
    CaseManager.deleteCase();
    // Verify record is deleted
    List<Case> cases = [SELECT Id FROM Case WHERE Id=:recordId];
    System.assert(cases.size() == 0);
}
@isTest static void testUpsertCase() {
    // 1. Insert new record
    ID case1Id = CaseManager.upsertCase(
            'Ferocious chipmunk', 'New', 'Phone', 'Low', null);
    // Verify new record was created
    System.assert(Case1Id != null);
    Case case1 = [SELECT Id,Subject FROM Case WHERE Id=:case1Id];
    System.assert(case1 != null);
    System.assertEquals(case1.Subject, 'Ferocious chipmunk');
    // 2. Update status of existing record to Working
    ID case2Id = CaseManager.upsertCase(
            'Ferocious chipmunk', 'Working', 'Phone', 'Low', case1Id);
    // Verify record was updated
    System.assertEquals(case1Id, case2Id);
    Case case2 = [SELECT Id,Status FROM Case WHERE Id=:case2Id];
    System.assert(case2 != null);
    System.assertEquals(case2.Status, 'Working');
}
@isTest static void testUpdateCaseFields() {
    Id recordId = createTestRecord();
    RestRequest request = new RestRequest();
    request.requestUri =
        'https://yourInstance.my.salesforce.com/services/apexrest/Cases/'
        + recordId;
    request.httpMethod = 'PATCH';
    request.addHeader('Content-Type', 'application/json');
    request.requestBody = Blob.valueOf('{"status": "Working"}');
    RestContext.request = request;
    // Update status of existing record to Working
    ID thisCaseId = CaseManager.updateCaseFields();
    // Verify record was updated
    System.assert(thisCaseId != null);
    Case thisCase = [SELECT Id,Status FROM Case WHERE Id=:thisCaseId];
    System.assert(thisCase != null);
    System.assertEquals(thisCase.Status, 'Working');
}
// Helper method
```

```
    static Id createTestRecord() {
        // Create test record
        Case caseTest = new Case(
            Subject='Test record',
            Status='New',
            Origin='Phone',
            Priority='Medium');
        insert caseTest;
        return caseTest.Id;
    }
}
```

calculatorServices Apex Class:

```
public class calculatorServices {
    public class doDivideResponse {
        public Double return_x;
        private String[] return_x_type_info = new
String[]{'return','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class doMultiply {
        public Double arg0;
        public Double arg1;
        private String[] arg0_type_info = new
String[]{'arg0','http://calculator.services/',null,'0','1','false'};
        private String[] arg1_type_info = new
String[]{'arg1','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'arg0','arg1'};
    }
    public class doAdd {
        public Double arg0;
        public Double arg1;
        private String[] arg0_type_info = new
```

```
String[]{'arg0','http://calculator.services/',null,'0','1','false'};
        private String[] arg1_type_info = new
String[]{'arg1','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'arg0','arg1'};
    }
    public class doAddResponse {
        public Double return_x;
        private String[] return_x_type_info = new
String[]{'return','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class doDivide {
        public Double arg0;
        public Double arg1;
        private String[] arg0_type_info = new
String[]{'arg0','http://calculator.services/',null,'0','1','false'};
        private String[] arg1_type_info = new
String[]{'arg1','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'arg0','arg1'};
    }
    public class doSubtract {
        public Double arg0;
        public Double arg1;
        private String[] arg0_type_info = new
String[]{'arg0','http://calculator.services/',null,'0','1','false'};
        private String[] arg1_type_info = new
String[]{'arg1','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'arg0','arg1'};
    }
    public class doSubtractResponse {
        public Double return_x;
        private String[] return_x_type_info = new
String[]{'return','http://calculator.services/',null,'0','1','false'};
```

```
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class doMultiplyResponse {
        public Double return_x;
        private String[] return_x_type_info = new
String[]{'return','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class CalculatorImplPort {
        public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/calculator';
        public Map<String,String> inputHttpHeaders_x;
        public Map<String,String> outputHttpHeaders_x;
        public String clientCertName_x;
        public String clientCert_x;
        public String clientCertPasswd_x;
        public Integer timeout_x;
        private String[] ns_map_type_info = new String[]{'http://calculator.services/',
'calculatorServices'};
        public Double doDivide(Double arg0,Double arg1) {
            calculatorServices.doDivide request_x = new calculatorServices.doDivide();
            request_x.arg0 = arg0;
            request_x.arg1 = arg1;
            calculatorServices.doDivideResponse response_x;
            Map<String, calculatorServices.doDivideResponse> response_map_x = new Map<String,
calculatorServices.doDivideResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
              '',
              'http://calculator.services/',
              'doDivide',
              'http://calculator.services/',
              'doDivideResponse',
              'calculatorServices.doDivideResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
        public Double doSubtract(Double arg0,Double arg1) {
            calculatorServices.doSubtract request_x = new calculatorServices.doSubtract();
            request_x.arg0 = arg0;
            request_x.arg1 = arg1;
            calculatorServices.doSubtractResponse response_x;
            Map<String, calculatorServices.doSubtractResponse> response_map_x = new
Map<String, calculatorServices.doSubtractResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
              '',
              'http://calculator.services/',
              'doSubtract',
              'http://calculator.services/',
              'doSubtractResponse',
              'calculatorServices.doSubtractResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
        public Double doMultiply(Double arg0,Double arg1) {
            calculatorServices.doMultiply request_x = new calculatorServices.doMultiply();
            request_x.arg0 = arg0;
            request_x.arg1 = arg1;
            calculatorServices.doMultiplyResponse response_x;
            Map<String, calculatorServices.doMultiplyResponse> response_map_x = new
Map<String, calculatorServices.doMultiplyResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
              '',
              'http://calculator.services/',
              'doMultiply',
              'http://calculator.services/',
              'doMultiplyResponse',
              'calculatorServices.doMultiplyResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
        public Double doAdd(Double arg0,Double arg1) {
            calculatorServices.doAdd request_x = new calculatorServices.doAdd();
            request_x.arg0 = arg0;
            request_x.arg1 = arg1;
            calculatorServices.doAddResponse response_x;
            Map<String, calculatorServices.doAddResponse> response_map_x = new Map<String,
calculatorServices.doAddResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
              '',
              'http://calculator.services/',
              'doAdd',
              'http://calculator.services/',
              'doAddResponse',
              'calculatorServices.doAddResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
    }
}
```

AsyncCalculatorServices Apex Class:

```
public class AsyncCalculatorServices {
   public class doDivideResponseFuture extends System.WebServiceCalloutFuture {
       public Double getValue() {
           calculatorServices.doDivideResponse response =
(calculatorServices.doDivideResponse)System.WebServiceCallout.endInvoke(this);
         return response.return_x;
      }
   }
   public class doSubtractResponseFuture extends System.WebServiceCalloutFuture {
      public Double getValue() {
         calculatorServices.doSubtractResponse response =
(calculatorServices.doSubtractResponse)System.WebServiceCallout.endInvoke(this);
         return response.return_x;
      }
   }
   public class doMultiplyResponseFuture extends System.WebServiceCalloutFuture {
      public Double getValue() {
         calculatorServices.doMultiplyResponse response =
(calculatorServices.doMultiplyResponse)System.WebServiceCallout.endInvoke(this);
         return response.return_x;
      }
   }
   public class doAddResponseFuture extends System.WebServiceCalloutFuture {
      public Double getValue() {
         calculatorServices.doAddResponse response =
(calculatorServices.doAddResponse)System.WebServiceCallout.endInvoke(this);
         return response.return_x;
      }
   }
   public class AsyncCalculatorImplPort {
      public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/calculator';
      public Map<String,String> inputHttpHeaders_x;
      public String clientCertName_x;
      public Integer timeout_x;
      private String[] ns_map_type_info = new String[]{'http://calculator.services/',
'calculatorServices'};
      public AsyncCalculatorServices.doDivideResponseFuture
beginDoDivide(System.Continuation continuation,Double arg0,Double arg1) {
         calculatorServices.doDivide request_x = new calculatorServices.doDivide();
         request_x.arg0 = arg0;
         request_x.arg1 = arg1;
         return (AsyncCalculatorServices.doDivideResponseFuture)
System.WebServiceCallout.beginInvoke(
           this,
           request_x,
           AsyncCalculatorServices.doDivideResponseFuture.class,
           continuation,
           new String[]{endpoint_x,
           '',
           'http://calculator.services/',
           'doDivide',
           'http://calculator.services/',
           'doDivideResponse',
           'calculatorServices.doDivideResponse'}
         );
      }
      public AsyncCalculatorServices.doSubtractResponseFuture
beginDoSubtract(System.Continuation continuation,Double arg0,Double arg1) {
         calculatorServices.doSubtract request_x = new calculatorServices.doSubtract();
         request_x.arg0 = arg0;
         request_x.arg1 = arg1;
         return (AsyncCalculatorServices.doSubtractResponseFuture)
System.WebServiceCallout.beginInvoke(
           this,
           request_x,
           AsyncCalculatorServices.doSubtractResponseFuture.class,
           continuation,
           new String[]{endpoint_x,
           '',
           'http://calculator.services/',
           'doSubtract',
           'http://calculator.services/',
           'doSubtractResponse',
           'calculatorServices.doSubtractResponse'}
         );
      }
      public AsyncCalculatorServices.doMultiplyResponseFuture
beginDoMultiply(System.Continuation continuation,Double arg0,Double arg1) {
         calculatorServices.doMultiply request_x = new calculatorServices.doMultiply();
         request_x.arg0 = arg0;
         request_x.arg1 = arg1;
         return (AsyncCalculatorServices.doMultiplyResponseFuture)
System.WebServiceCallout.beginInvoke(
           this,
           request_x,
           AsyncCalculatorServices.doMultiplyResponseFuture.class,
           continuation,
           new String[]{endpoint_x,
           '',
           'http://calculator.services/',
           'doMultiply',
           'http://calculator.services/',
           'doMultiplyResponse',
           'calculatorServices.doMultiplyResponse'}
         );
      }
      public AsyncCalculatorServices.doAddResponseFuture beginDoAdd(System.Continuation
continuation,Double arg0,Double arg1) {
         calculatorServices.doAdd request_x = new calculatorServices.doAdd();
         request_x.arg0 = arg0;
         request_x.arg1 = arg1;
         return (AsyncCalculatorServices.doAddResponseFuture)
System.WebServiceCallout.beginInvoke(
           this,
           request_x,
           AsyncCalculatorServices.doAddResponseFuture.class,
           continuation,
           new String[]{endpoint_x,
           '',
           'http://calculator.services/',
           'doAdd',
           'http://calculator.services/',
           'doAddResponse',
           'calculatorServices.doAddResponse'}
         );
      }
  }
}
```

ParkLocator Apex Class:

```
public class ParkLocator {
    public static string[] country(string theCountry) {
        ParkService.ParksImplPort  parkSvc = new  ParkService.ParksImplPort(); // remove space
        return parkSvc.byCountry(theCountry);
    }
}
```

ParkLocatorTest Apex Class:

```
@isTest
private class ParkLocatorTest {
    @isTest static void testCallout() {
        Test.setMock(WebServiceMock.class, new ParkServiceMock ());
        String country = 'United States';
        List<String> result = ParkLocator.country(country);
        List<String> parks = new List<String>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};
         System.assertEquals(parks, result);
    }
}
```

AccountManager Apex Class:

```
@RestResource(urlMapping='/Accounts/*/contacts')
global class AccountManager {
    @HttpGet
    global static Account getAccount() {
        RestRequest req = RestContext.request;
        String accId = req.requestURI.substringBetween('Accounts/', '/contacts');
        Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)
                       FROM Account WHERE Id = :accId];
        return acc;
    }
}
```

AccountManagerTest Apex Class:

```
@isTest
private class AccountManagerTest {

    private static testMethod void getAccountTest1() {
        Id recordId = createTestRecord();
        // Set up a test request
        RestRequest request = new RestRequest();
        request.requestUri = 'https://na1.salesforce.com/services/apexrest/Accounts/'+ recordId
+'/contacts' ;
        request.httpMethod = 'GET';
        RestContext.request = request;
        // Call the method to test
        Account thisAccount = AccountManager.getAccount();
        // Verify results
        System.assert(thisAccount != null);
        System.assertEquals('Test record', thisAccount.Name);

    }

    // Helper method
        static Id createTestRecord() {
        // Create test record
        Account TestAcc = new Account(
          Name='Test record');
        insert TestAcc;
        Contact TestCon= new Contact(
        LastName='Test',
        AccountId = TestAcc.id);
        return TestAcc.Id;
    }
}
```

CalculatorCalloutMock Apex Class:

```
@isTest
global class CalculatorCalloutMock implements WebServiceMock {
   global void doInvoke(
           Object stub,
           Object request,
           Map<String, Object> response,
           String endpoint,
           String soapAction,
           String requestName,
           String responseNS,
           String responseName,
           String responseType) {
        // start - specify the response you want to send
        calculatorServices.doAddResponse response_x =
            new calculatorServices.doAddResponse();
        response_x.return_x = 3.0;
        // end
        response.put('response_x', response_x);
    }
}
```