CS602 Final Exam

Q1. (30 points)

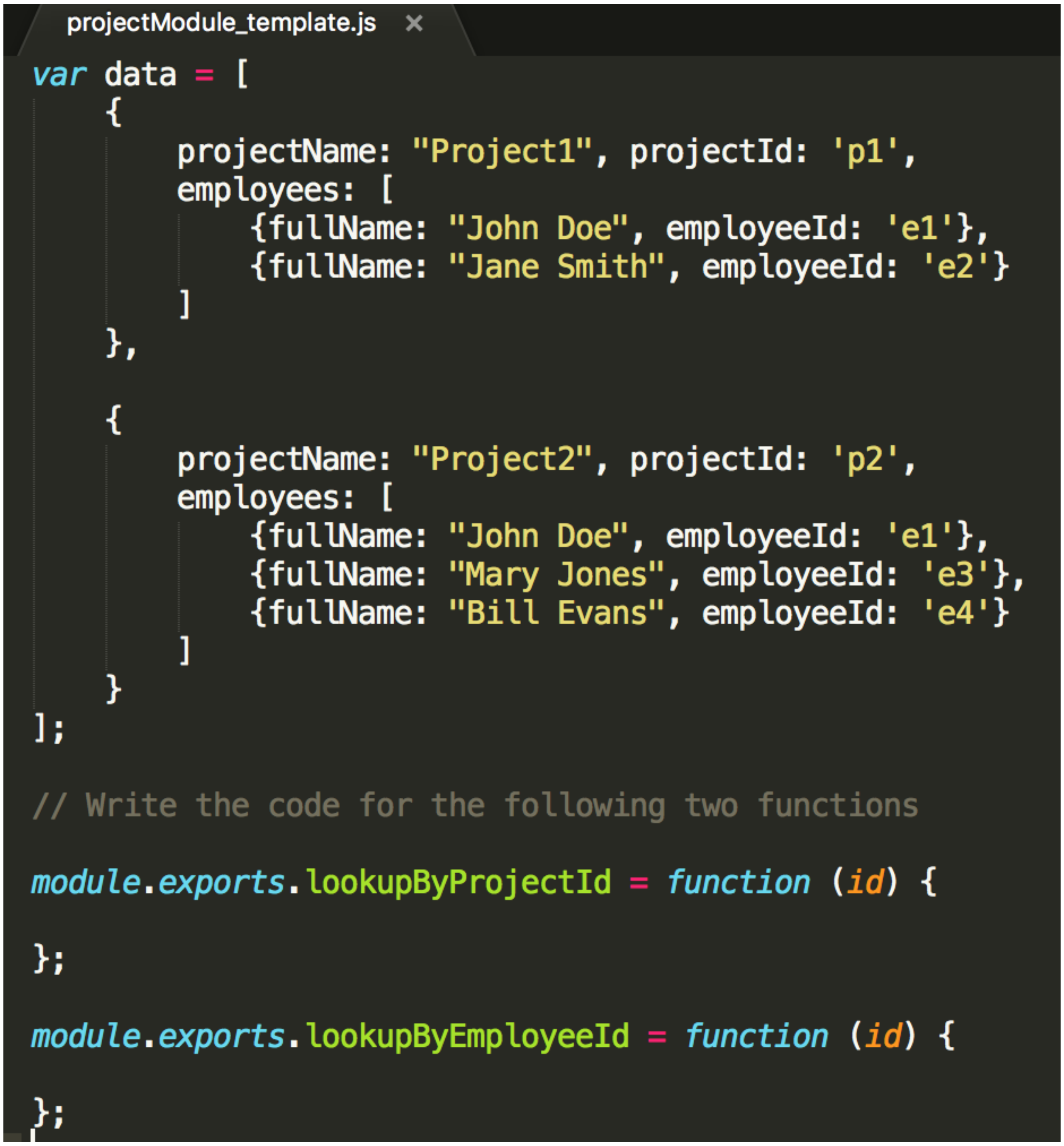
Write a Node.js module**, projectModule.js**, with the following functionality. The module maintains an array of JavaScript objects, each with the property *projectName*, and a unique project*Id* value. Each project, in turn, has an array of employee JavaScript objects. An employee has the properties, fullName, and a unique employeeId.

The module should export the functions lookupByProjectId and lookupByEmployeeId.

The function lookupByProjectId should return the JavaScript object from the data whose projectId matches the specified argument. If the specified id is not present, undefined is returned.

The function lookupByEmployeeId should return the **array** of JavaScript objects from the data that has all the projects in which the specified employee is working. If the specified employeeId is not assigned to any project, [] is returned.

Write the code for the two module functions using JavaScript. The template file is shown below. Please note that there can be any number of projects and any number of employees for each project in the data.



```
var data = [
    {
        projectName: "Project1", projectId: 'p1',
        employees: [
            {fullName: "John Doe", employeeId: 'e1'},
            {fullName: "Jane Smith", employeeId: 'e2'}
        ]
    },

    {
        projectName: "Project2", projectId: 'p2',
        employees: [
            {fullName: "John Doe", employeeId: 'e1'},
            {fullName: "Mary Jones", employeeId: 'e3'},
            {fullName: "Bill Evans", employeeId: 'e4'}
        ]
    }
];

// Write the code for the following two functions

module.exports.lookupByProjectId = function (id) {

};

module.exports.lookupByEmployeeId = function (id) {

};
```

Sample Output for lookupByProjectId('p2'):

```
{ projectName: 'Project2',
  projectId: 'p2',
  employees:
   [ { fullName: 'John Doe', employeeId: 'e1' },
     { fullName: 'Mary Jones', employeeId: 'e3' },
     { fullName: 'Bill Evans', employeeId: 'e4' } ] }
```

Sample Output for lookupByEmployeeId ('e3'):

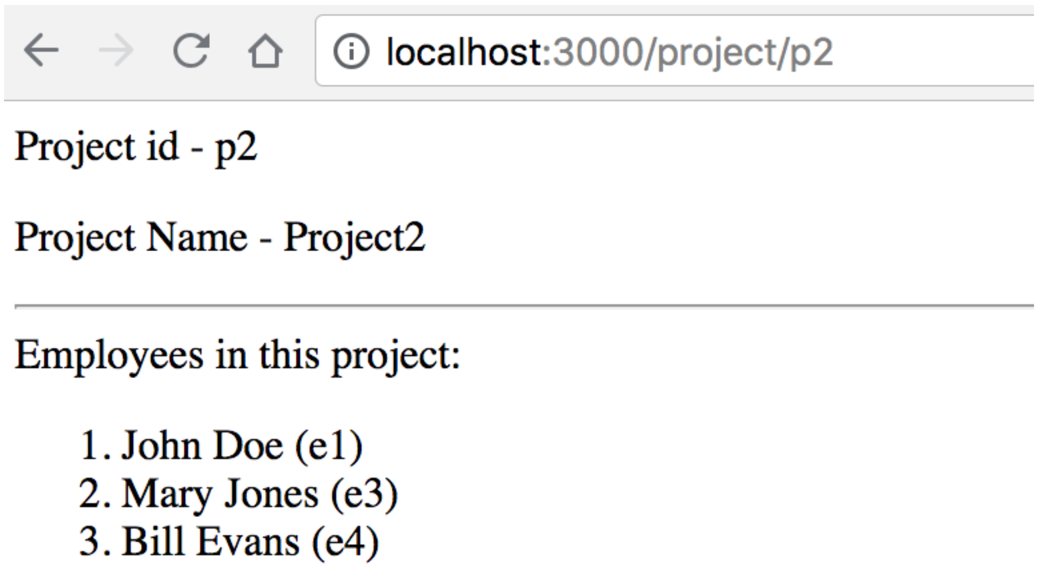
```
[ { projectName: 'Project2',
    projectId: 'p2',
    employees:
     [ { fullName: 'John Doe', employeeId: 'e1' },
       { fullName: 'Mary Jones', employeeId: 'e3' },
       { fullName: 'Bill Evans', employeeId: 'e4' } ] } ]
```

Sample Output for lookupByEmployeeId ('e1'):

```
[ { projectName: 'Project1',
    projectId: 'p1',
    employees:
     [ { fullName: 'John Doe', employeeId: 'e1' },
       { fullName: 'Jane Smith', employeeId: 'e2' } ] },
  { projectName: 'Project2',
    projectId: 'p2',
    employees:
     [ { fullName: 'John Doe', employeeId: 'e1' },
       { fullName: 'Mary Jones', employeeId: 'e3' },
       { fullName: 'Bill Evans', employeeId: 'e4' } ] } ]
```

Q2. (20 points)

Using the **projectModule** functionality from Question1, show how you would develop Node web application that can display the information about the requested project. The web application should handle GET requests for the form /project/<projectId>



localhost:3000/project/p2

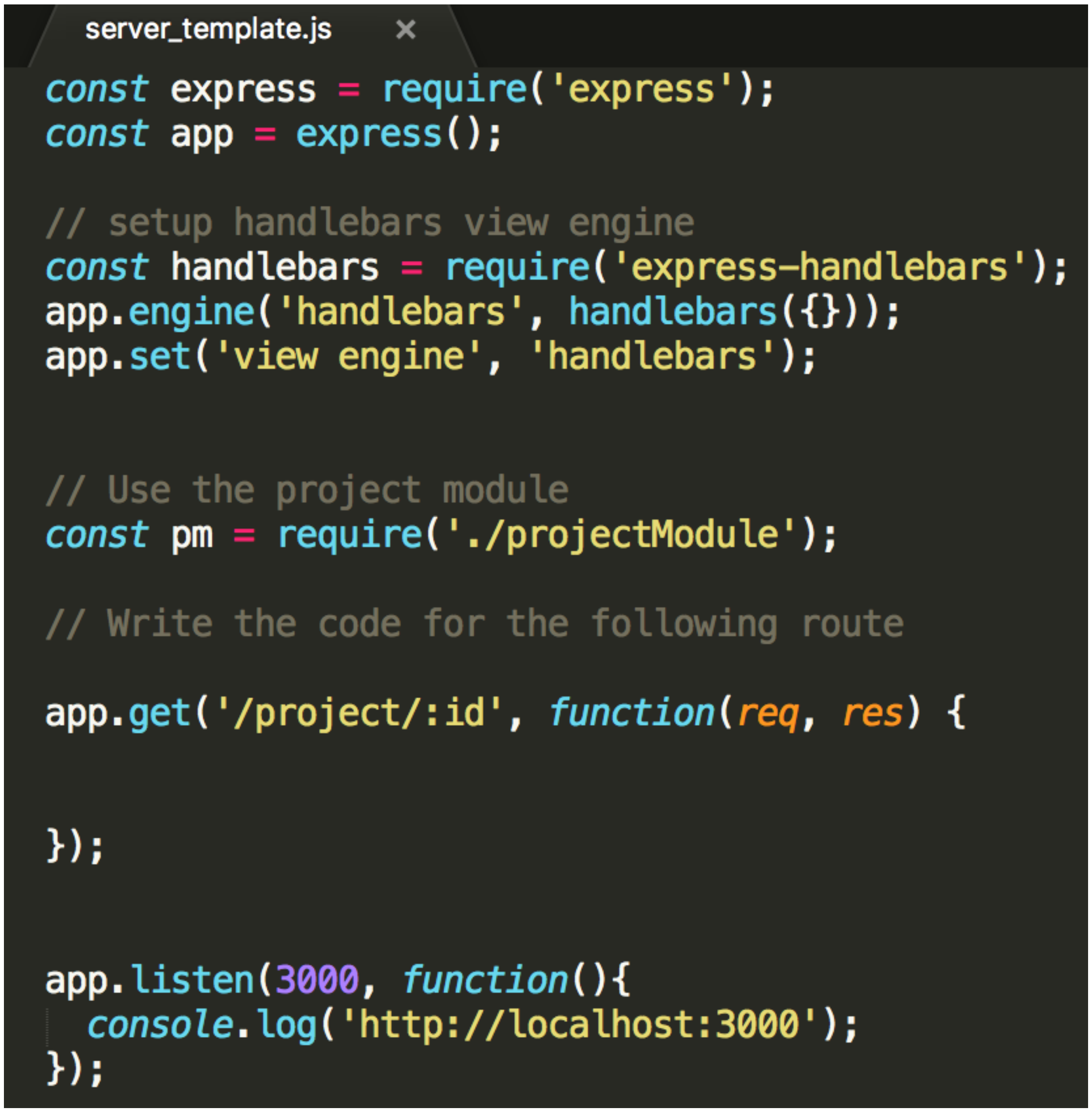
Project id - p2

Project Name - Project2

Employees in this project:

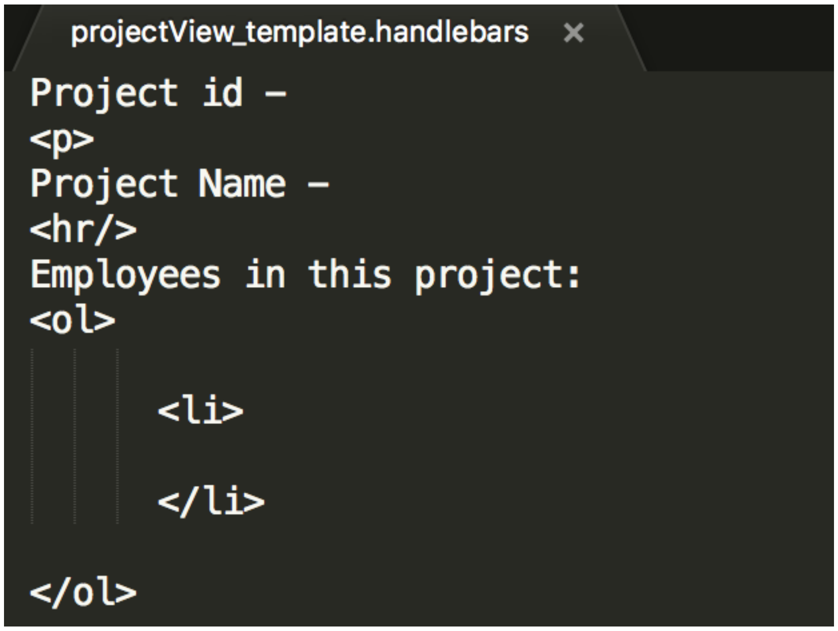
1. John Doe (e1)
2. Mary Jones (e3)
3. Bill Evans (e4)

The server code using the Express and handlebars framework is shown below. **Write only** the code for the /project/:id route as given in the template code below. The project should be looked up using the ***projectModule*** from the previous question. This project should be passed as the model for the ***projectView*** page.


server_template.js

```javascript
const express = require('express');
const app = express();

// setup handlebars view engine
const handlebars = require('express-handlebars');
app.engine('handlebars', handlebars({}));
app.set('view engine', 'handlebars');


// Use the project module
const pm = require('./projectModule');

// Write the code for the following route

app.get('/project/:id', function(req, res) {


});


app.listen(3000, function(){
  console.log('http://localhost:3000');
});
```

Now, write the ***projectView*** page for rendering the corresponding project. The template for the page is shown below.



```
projectView_template.handlebars
Project id -
<p>
Project Name -
<hr/>
Employees in this project:
<ol>

      <li>

      </li>

</ol>
```

Q3. (15 points)
Write a PHP script for determining which characters inside of a given array are vowels (a, e, i, o, u), consonants, and integers. You can assume that the array only holds letters and numbers and you do **not** need to handle objects. The letters in the array are strings of unit length. The numbers in the array will only be single digit numbers. The PHP function **is_int(arg)** returns true if the given argument is an integer. Note that the code should work for any size array containing a mixture of vowels, consonants, and single digits.

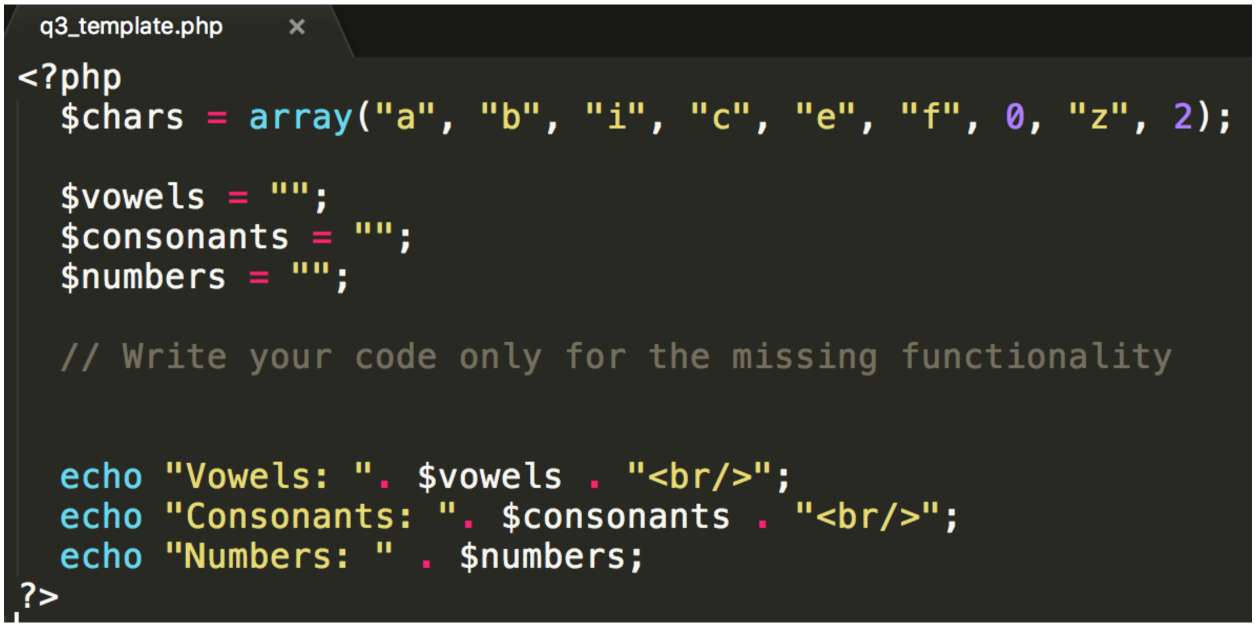
Example Input:

```
$chars = array("a", "b", "i", "c", "e", "f", 0, "z", 2);
```

The output for the above example input should be as follows.

```
Vowels: aie
Consonants: bcfz
Numbers: 02
```

The template code for php script is shown below. Write only the missing functionality in the solution.



```php
<?php
  $chars = array("a", "b", "i", "c", "e", "f", 0, "z", 2);

  $vowels = "";
  $consonants = "";
  $numbers = "";

  // Write your code only for the missing functionality


  echo "Vowels: ". $vowels . "<br/>";
  echo "Consonants: ". $consonants . "<br/>";
  echo "Numbers: " . $numbers;
?>
```

Q4. (15 points)
Write the code for a PHP class called **Rectangle** that can be used to create rectangle objects. Your class should include the following information:

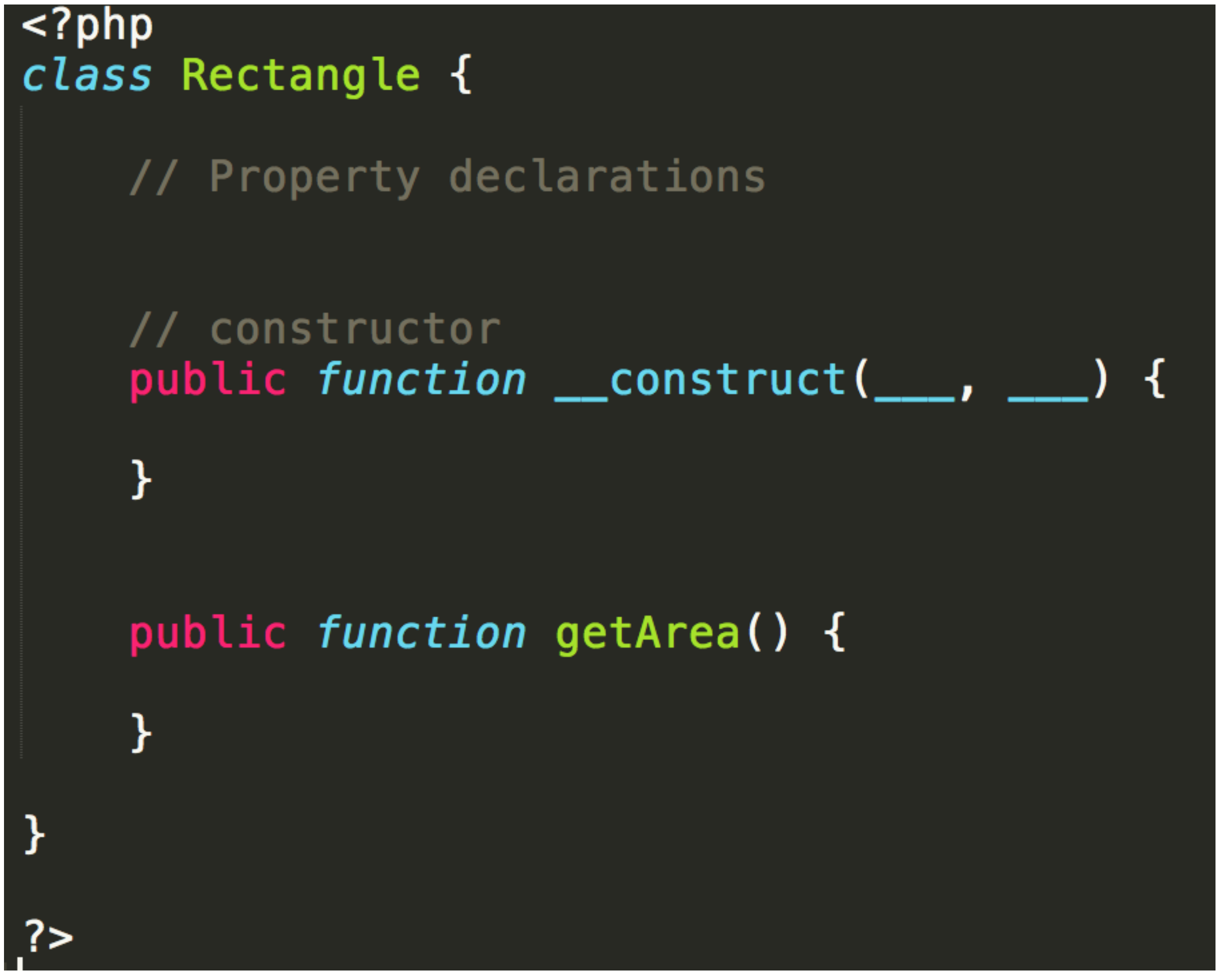
a. A private instance variable named **width** that specifies the width of the rectangle.
b. A private instance variable named **height** that specifies the height of the rectangle.
c. A constructor with two arguments that creates a rectangle with the specified width and height.
d. A method named **getArea()** that returns the area of the rectangle.
   Hint: Area = width * height

Now, write the code for a PHP class, **Square**, extending the Rectangle class, with **no** private instance variables and with a single constructor with the side of the square as the only constructor argument.

Now, create an instance of **Rectangle** with a width of 10 and a height of 20. Show how you would display the area of this rectangle.

Now, create an instance of **Square** with a side of 20. Show how you would display the area of this square.

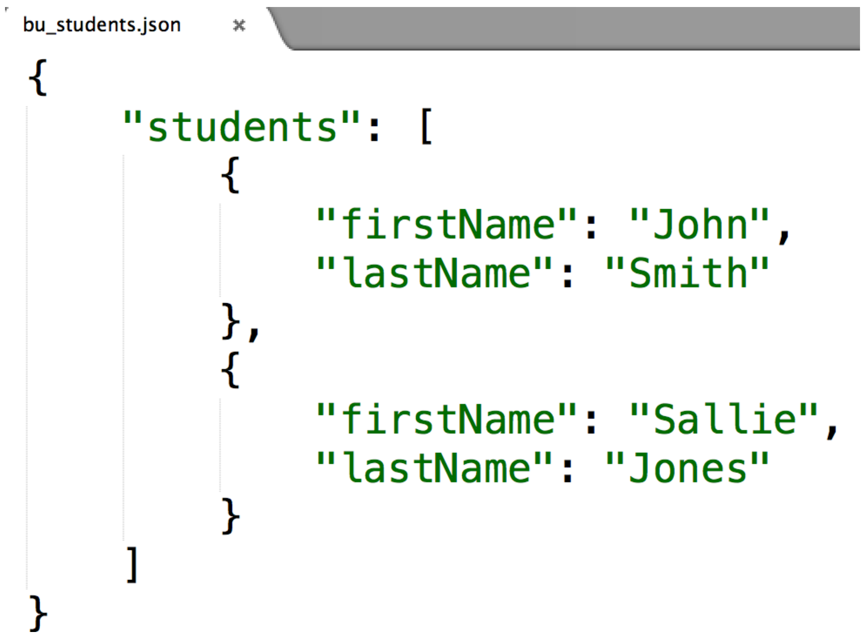
A template for the Rectangle class is shown below.



```php
<?php
class Rectangle {

    // Property declarations


    // constructor
    public function __construct(___, ___) {

    }


    public function getArea() {

    }

}

?>
```

Q5. (20 Points)

Given the following JSON structure in the file **bu_students.json**,



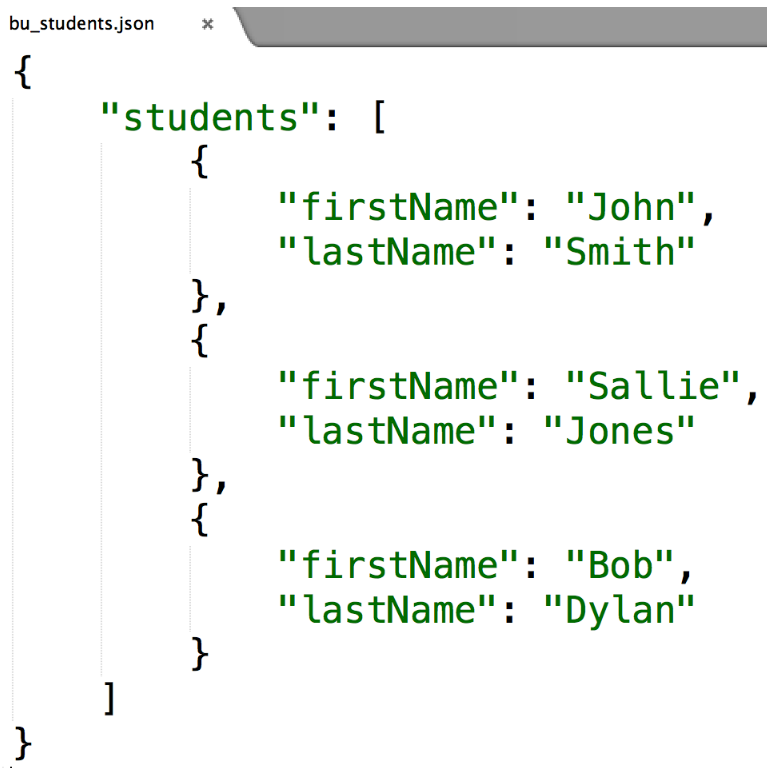
bu_students.json

```json
{
    "students": [
        {
            "firstName": "John",
            "lastName": "Smith"
        },
        {
            "firstName": "Sallie",
            "lastName": "Jones"
        }
    ]
}
```

and the following input form for submitting new data:

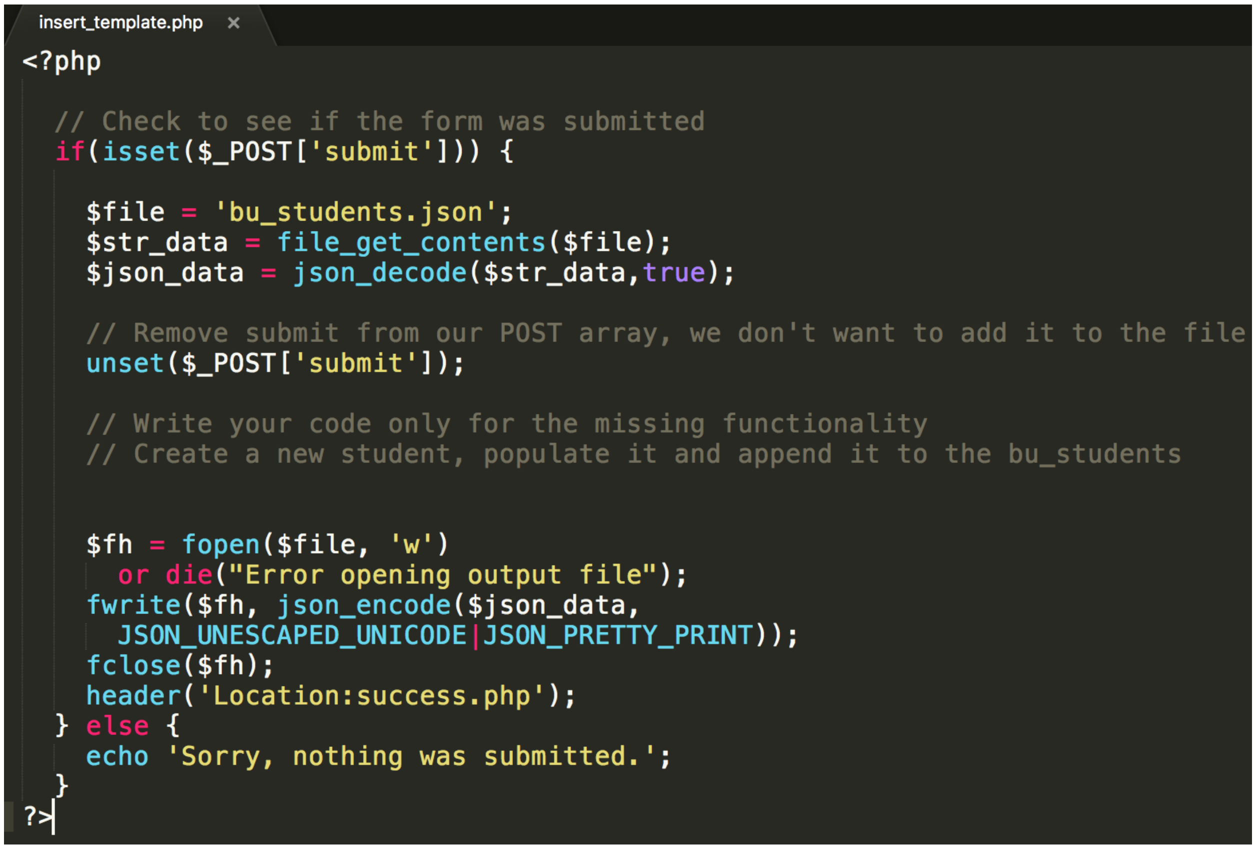
```html
<form action="insert.php" method="post">
    First Name:
    <input type="text" name="firstName">
    <br/>
    Last Name:
    <input type="text" name="lastName">
    <br/>
    <input type="submit" name="submit"
            value="Submit">
</form>
```

Write the code for the PHP script **insert.php**, that will insert the submitted data to the json file. For example, if the form was submitted using the first name, Bob, and last name, Dylan, the resulting bu_students.json file should as follows:



```
{
    "students": [
        {
            "firstName": "John",
            "lastName": "Smith"
        },
        {
            "firstName": "Sallie",
            "lastName": "Jones"
        },
        {
            "firstName": "Bob",
            "lastName": "Dylan"
        }
    ]
}
```

The template code for insert.php is shown below. Write only the missing functionality in the solution.

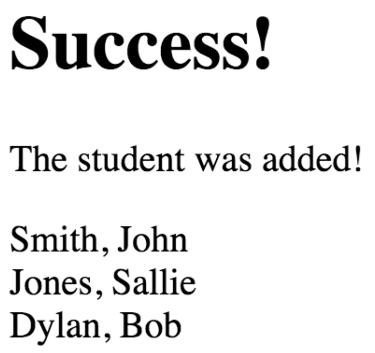


insert_template.php

```php
<?php

  // Check to see if the form was submitted
  if(isset($_POST['submit'])) {

    $file = 'bu_students.json';
    $str_data = file_get_contents($file);
    $json_data = json_decode($str_data,true);

    // Remove submit from our POST array, we don't want to add it to the file
    unset($_POST['submit']);

    // Write your code only for the missing functionality
    // Create a new student, populate it and append it to the bu_students


    $fh = fopen($file, 'w')
      or die("Error opening output file");
    fwrite($fh, json_encode($json_data,
      JSON_UNESCAPED_UNICODE|JSON_PRETTY_PRINT));
    fclose($fh);
    header('Location:success.php');
  } else {
    echo 'Sorry, nothing was submitted.';
  }
?>
```

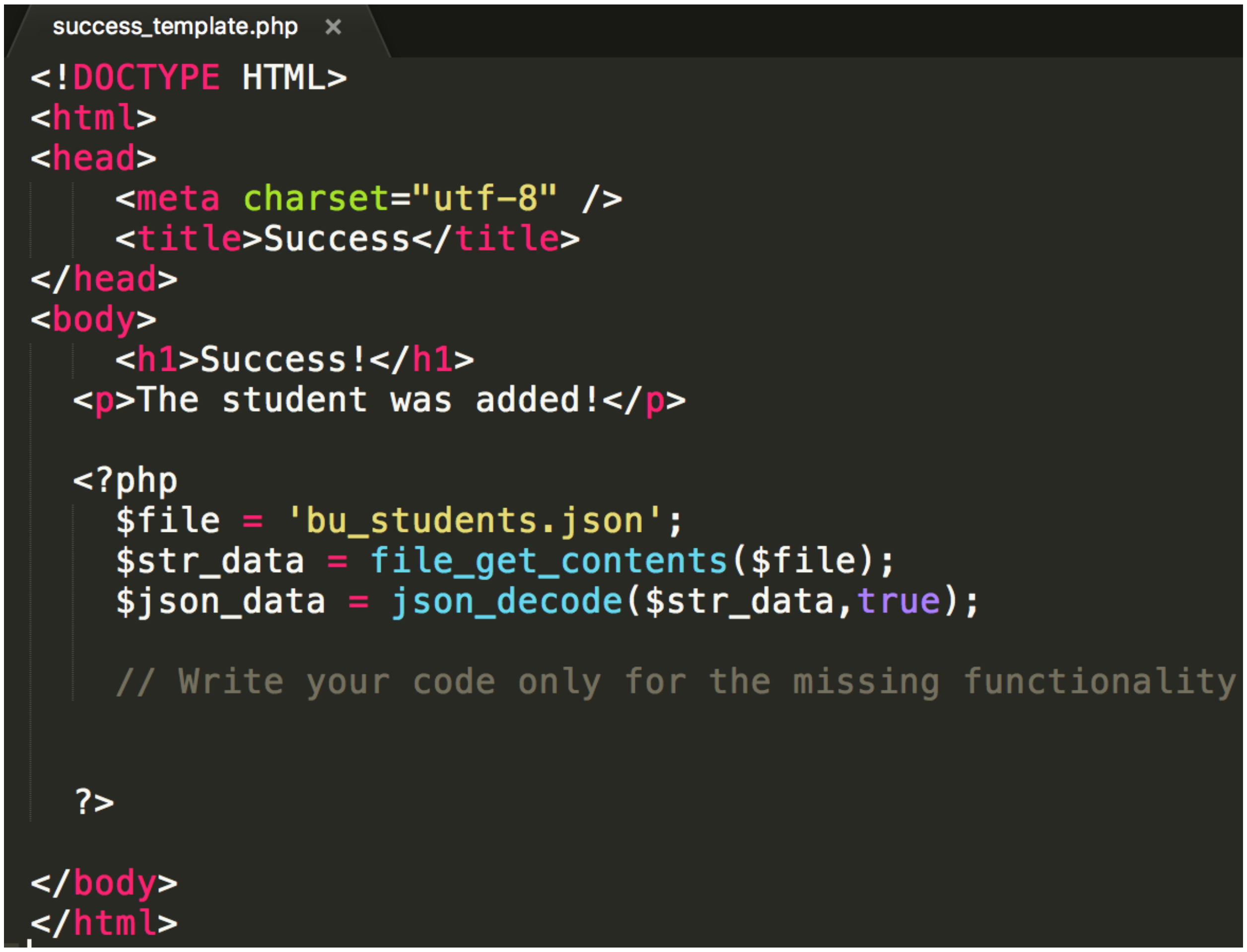
The insert.php script redirects the browser to success.php.

Now, write the code for PHP script **success.php** that will read the contents of the json file and shows the output as follows:



```
Success!

The student was added!

Smith, John
Jones, Sallie
Dylan, Bob
```

The template code for success.php is shown below. Write only the missing functionality in the solution.



success_template.php ×

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8" />
    <title>Success</title>
</head>
<body>
    <h1>Success!</h1>
  <p>The student was added!</p>

  <?php
    $file = 'bu_students.json';
    $str_data = file_get_contents($file);
    $json_data = json_decode($str_data,true);

    // Write your code only for the missing functionality


  ?>

</body>
</html>
```