**IST105 - Introduction to Programming**

**Final Exam:** Create a "Digital Party Planner" Using Lists, Loops, Bitwise Operators, and Load Balancers

**Course:** IST105 - Introduction to Programming  
**Instructor:** Washington Valencia  
**Version:** **April - 2025 (Final Exam -RAv1)**

**Important Notes:**

* Code with syntax or runtime errors will receive a grade of zero.
* The submission must include screenshots of the fully functional web application running on the AWS EC2 instances, showing the public IP address and the load balancer.
* **Submissions are only accepted through Canvas.**

### **Objective:**

Build a Python-based "Digital Party Planner" application that utilizes lists, loops, and bitwise operators to calculate a **party code**, dynamically modify it using **conditional logic**, and deploy the application on two AWS EC2 instances with an **Application Load Balancer (ALB)** to distribute traffic. This exercise also involves version control using Git.

**Exam Tasks:**

**1. Create EC2 Instances and Install Required Software**

* **Launch two EC2 instances** using the **Amazon Linux 2 AMI**.
* **Configure security groups** to allow **HTTP** (port 80) and **SSH** (port 22) access.
* **Install Apache, Python, Git,** and all required dependencies on both instances.
* **Start and enable the Apache web server** on both instances.
* Confirm that Apache is working by accessing the public IP of the EC2 instances in your browser.
* **Note** the public IP addresses of both instances.

**2. Write the Python Application**

Clone the repository from [**https://github.com/eduval/DigitalPartyPlanner**](https://github.com/eduval/DigitalPartyPlanner) and work with the code to:

* Create a Python script called **party\_planner.py** to:

**Display a Predefined List of 15 Party Items**:

Display the following party items with their respective indices in the list:

**0:** Cake

**1:** Balloons

**2**: Music System

**3:** Lights

**4:** Catering Service

**5:** DJ

**6:** Photo Booth

**7:** Tables

**8:** Chairs

**9:** Drinks

**10:** Party Hats

**11:** Streamers

**12:** Invitation Cards

**13:** Party Games

**14:** Cleaning Service

**Allow the User to Select One or More Items by Their Indices**:

* Users can select one or more items by entering their indices from the list. For example, selecting indices 0, 1, and 2 corresponds to "Cake", "Balloons", and "Music System".

**Assign Values to Each Party Item**: Each party item is assigned a value, which is used in the bitwise AND operation:

0: **Cake (value = 20)**

1: **Balloons (value = 21)**

2: **Music System (value = 10**)

3: **Lights (value = 5)**

4: **Catering Service (value = 8)**

5: **DJ (value = 3)**

6: **Photo Booth (value = 15**)

7: **Tables (value = 7)**

8: **Chairs (value = 12)**

9: **Drinks (value = 6)**

10: **Party Hats (value = 9)**

11: **Streamers (value = 18)**

12: **Invitation Cards (value = 4)**

13: **Party Games (value = 2)**

14: **Cleaning Service (value = 11)**

**Use the Bitwise AND (&) Operation to Calculate a Base Party Code**:

* Perform the bitwise AND operation on the selected items' values to calculate the base party code.

**Example 1**:

* Selecting **"Cake"** (index 0) and **"Balloons"** (index 1):

Cake = 20 = 10100 (binary)

Balloons = 21 = 10101 (binary)

20 & 21 = 10100 & 10101 = 10100 (binary) = **20**

**Example 2**:

* Selecting **"Music System"** (index 2) and **"Lights"** (index 3):

Music System = 10 = 01010 (binary)

Lights = 5 = 00101 (binary)

10 & 5 = 01010 & 00101 = 00000 (binary) = **0**

**Modify the Base Party Code with If/Else Conditions**:

* If the **base\_code** is calculated, modify it based on the following conditions:
  + If base\_code == 0: Add 5 and append the string **"Epic Party Incoming!"**.
  + If base\_code > 5: Subtract 2 and append the string "**Let's keep it classy!".**
  + Else (if base\_code is between 1 and 5): Append the string **"Chill vibes only!".**

**Output the Selected Items, Final Party Code, and the Fun Message in HTML Format**:

* Render the output in HTML format to display the selected items, final party code, and fun message to the user.

**Example of expected output for server1:**

**Webserver 1:**

**Available Party Items:**

0: Cake

1: Balloons

2: Music System

3: Lights

4: Catering Service

5: DJ

6: Photo Booth

7: Tables

8: Chairs

9: Drinks

10: Party Hats

11: Streamers

12: Invitation Cards

13: Party Games

14: Cleaning Service

**Enter item indices separated by commas (e.g., 0, 2): 0, 1**

**Selected Items:** Cake, Balloons  
**Base Party Code:** 20 & 21 = 20  
**Adjusted Party Code:** 20 - 2 = 18  
**Final Party Code:** 18

**Message: Let's keep it classy!**

**3. Create the Web Interface (Optional)**

Write a **PHP** or **HTML form (party\_form.php)** to:

* Collect **user input** (party item selections) from a dropdown or checkboxes.
* Pass the selected item indices to the party\_planner.py script via either a **URL query string** or form submission.
* Display the result on the webpage with the **selected items**, **final party code**, and **fun message**.

**Note:** Creating the user input form is optional. The user can either use the form to submit values or manually input values via the URL query string.

**4. Version Control with Git**

* Clone the [provided GitHub repository](https://github.com/eduval/GiftSelectionExtravaganza).
* Create and switch between the following branches:
  + **main** (production-ready)
  + **development** (for testing new features)
  + **feature1** (for implementing the "party planning" feature)
* Push the code to the **public GitHub repository** with these branches.
* Provide the GitHub URL for your repository.

**5. Deploy the Application on EC2 with Load Balancer**

* Deploy your application on **both EC2 instances** (ensure both are running the Python script and Apache).
* Create an **Application Load Balancer (ALB)** to distribute traffic between the two EC2 instances.
* Verify that the ALB is routing traffic to both instances correctly.
* Confirm that the load balancer is healthy and distributing requests between the instances.

**Submission Requirements (Total: 50 points)**

|  |  |
| --- | --- |
| **Requirement** | **Points** |
| **GitHub Repository Link (Complete and Fully Functional)** | 35 |
| **Screenshot of EC2 Compute Instance 1 - Web Application** | 5 |
| **Screenshot of EC2 Compute Instance 2 - Web Application** | 5 |
| **Load Balancer Web Application and verification** | 5 |
| **Total** | 50 |

**Important Notes:**

* **Repository Completeness:** Ensure that all files from this lab exercise are included in your GitHub repository.
* **Functionality:** Ensure the application is fully operational on the EC2 instances, accessible through their public IP addresses, and properly routed via the Load Balancer domain.
* **Testing:** Thoroughly test all features before submitting. Testing itself does not count towards points but is essential to demonstrate the working functionality of the application.

[Add your Screenshot of EC2 Compute Instance 1 - Web Application here]

[Add your Screenshot of EC2 Compute Instance 2 - Web Application here]

[Load Balancer Web Application and verification]

[Add your public GitHub repo here]