**Advance Development and Emerging Technologies**

**Machine Problem:**

| | |
|---|---|
| Name of the student | |
| Course/section | |
| Date of Submission | |

| | |
|---|---|
| **Weight** | **60% of Lab Midterm Grade** |
| **% Lead Time** | **15 July (0700H) – 17 April 2023 (800H)** |
| **Means of submission** | **Online (GitHub and Canvas Account)** |

**For those who will get 100 in this Machine Problem will be exempted in FINAL EXAMINATIONS!!!**

**GROUP YOURSELVES INTO TWO. AVOID COPYING FROM DIFFERENT GROUP.**

| Check Item | Points (A) | Weight (B) | A * B |
|---|---|---|---|
| [1] **Algorithm**<br>Correctness / appropriateness of logic control structures used to solve the problem (sequence, conditional execution, iteration, functions). Specified input / output screen layout are followed. | 5 | 7 | **35** |
| [2] **Creation of Variables and/or Constants**<br>Sufficient number, conformity to rules for giving names to memory spaces. | 5 | 4 | **20** |
| [3] **Correct Initialization and/or Reinitialization of Variables**<br>Each declared memory spaces are correctly set to its starting value every time the program is run or re-runned. | 5 | 2 | **10** |
| [4] **Correct Program Syntax**<br>Source code containing the written algorithm can be compiled successfully. | 5 | 3 | **15** |
| [5] **Code Readability and Maintainability**<br>Code layout makes it easy to understand and debug (well-spaced, proper use of code indentions, line skipping, use of comments when needed) 5 | 5 | 4 | **20** |
| **TOTAL** | 25 | 20 | **100** |

Deductions and/or Penalties

- Date and time code submission: **July 15 2023, Monday, no later than 8:00 AM**

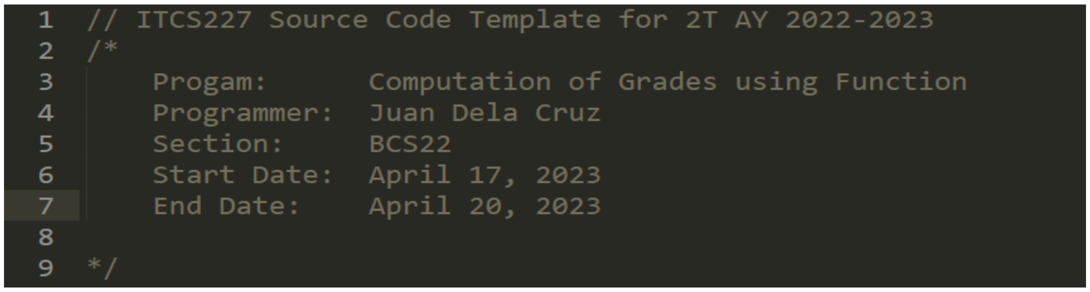
- Mode of source code submission: **DLSUD – SB and GitHub**
- **July 15 2023, 8:01 PM** as date and time of submission is already considered as 1 day late

| Days Late | Maximum Grade |
|---|---|
| 0 | 100/100 |
| 1 – 7 | 80/100 |
| 8 – 14 | 60 / 100 |
| At least 15 | 50 / 100 |

- Penalties due to nonconformity to special instructions on
  - Form of submission: source code (not a link to a work created using an online IDE)
  - source code file name format
  - required data from the student
  - specified running program interface specifications
  - others

**Special Instructions**

1. Include this name template in your source code:



```
// ITCS227 Source Code Template for 2T AY 2022-2023
/*
    Progam:       Computation of Grades using Function
    Programmer:   Juan Dela Cruz
    Section:      BCS22
    Start Date:   April 17, 2023
    End Date:     April 20, 2023

*/
```

2. Save the source code following the file naming format of:
   section_userName(firstletterOfYourName)(lastName)_machineProblem2.
   Example:
   bcs22_JDelaCruz_machineProblem2
3. Design and implement the algorithm USING ONLY the programming concepts covered in sessions discussion.
4. Make sure that the identifier used refer to each memory space conform to the rules specified in session (rules of naming conventions)
5. Follow the program user interface and format shown in the sample input/sample output

**Problem Statement:** E-commerce API MVP requirements

- User registration
- User authentication
- Create Product (Admin only)
- Retrieve all products
- Retrieve all active products
- Retrieve single product
- Update Product information (Admin only)
- Archive Product (Admin only)
- Activate Product (Admin only)
- Non-admin User checkout (Create Order)
- Retrieve User Details

- Stretch Goal

- Set user as admin (Admin only)
- Retrieve authenticated user's orders
- Retrieve all orders (Admin only)
- Add to Cart
    - Added Products
    - Change product quantities
    - Remove products from cart
    - Subtotal for each item
    - Total price for all items

**Formula:**
Grade = (Class participation * 30%) + (Summative Assessment * 30%) + (Final Examination * 40%)

| Grade Score | Letter Grade |
|---|---|
| 90 – 100 | A |
| 80 – 89 | B |
| 70 – 79 | C |
| 60 – 69 | D |
| Below 60 | F |

**Requirements:**

| Sections | Deliverables |
|---|---|
| 1 | ○ Data Model Design<br>○ User registration<br>○ User authentication |
| 2 | ○ Create Product (Admin only)<br>○ Retrieve all products<br>○ Retrieve all active products |
| 3 | ○ Retrieve single product<br>○ Update Product information (Admin only)<br>○ Archive Product (Admin only) |
| 4 | ○ Non-admin User checkout (Create Order)<br>○ Retrieve User Details<br>○ Initial deployment of API to Render |
| 5 | ○ Postman Collections |

Data Model Requirements – Option 1

- User
  - Email (String)
  - Password (String)
  - isAdmin (Boolean - defaults to false)
  - ordereredProduct (Array of Objects)
    - products (Array of Objects)
    - productId(ObjectId)
    - productName (String)
    - quantity (Number)
  - totalAmount (Number)
  - purchasedOn (Date - defaults to current timestamp)
- Product
  - Name (String)
  - Description (String)
  - Price (Number)
  - isActive (Boolean - defaults to true)

    - createdOn (Date - defaults to current timestamp)
    - userOrders (Array of Objects)
        - userId(ObjectId)
        - orderId (String) - this can be optional

Data Model Requirements – Option 2

- User
    - Email (String)
    - Password (String)
    - isAdmin (Boolean - defaults to false)
- Product
    - Name (String)
    - Description (String)
    - Price (Number)
    - isActive (Boolean - defaults to true)
    - createdOn (Date - defaults to current timestamp)
- Order
    - userId (ObjectId)
    - Must be associated with user who owns the order
    - products (Array of Objects)
    - productId (ObjectId)
    - quantity (Number)
    - totalAmount (Number)
    - purchasedOn (Date - defaults to current timestamp)

Product Workflow

- *Create product functionality (admin only)*
- *Retrieve all active products*

- Create product workflow
    - An authenticated admin user sends a POST request containing a JWT in its header to the /products endpoint.
    - API validates JWT, returns false if validation fails.
    - If validation successful, API creates product using the contents of the request body.

- Retrieve All Active Products Workflow
    - A GET request is sent to the /products endpoint.
    - API retrieves all active products and returns them in its response.

    - *Retrieve single product*
    - *Update Product Info (admin only)*
    - *Archive Product (admin only)*

- Retrieve Single Product Workflow

    - A GET request is sent to the /products/:productId endpoint.
    - API retrieves product that matches productId URL parameter and returns it in its response.
- Update Product
    - An authenticated admin user sends a PUT request containing a JWT in its header to the /products/:productId endpoint.
    - API validates JWT, returns false if validation fails.
    - If validation successful, API finds product with ID matching the productId URL parameter and overwrites its info with those from the request body.

- Archive Product
    - An authenticated admin user sends a PUT request containing a JWT in its header to the /products/:productId/archive endpoint.
    - API validates JWT, returns false if validation fails.
    - If validation successful, API finds product with ID matching the productId URL parameter and sets its isActive property to false.

        - *Create Order (NON-admin only)*
        - *Retrieve User Details*
        - *Initial deployment of API to Render*
        - *Stretch Goals*
        - *Retrieve All Orders (admin only)*
        - *Retrieve Authenticated User's Orders (NON-admin only)*
        - *Set User as Admin functionality*
- Create Order workflow
    - An authenticated NON-admin user sends a POST request containing a JWT in its header to the /users/checkout endpoint.
    - API validates user identity via JWT, returns false if validation fails.
    - If validation successful, API creates order using the contents of the request body.
- Retrieve User Details workflow
    - An authenticated user sends a GET request containing a JWT in its header to the /:userId/userDetails endpoint.
    - Reassign the password of the returned document to an empty string.
    - API validates JWT, returns false if validation fails.
    - If validation is successful, API retrieves all user’s details and returns them in its response.
- Retrieve All Orders Workflow
    - A GET request containing a JWT in its header is sent to the /users/orders endpoint.
    - API validates user is an admin via JWT, returns false if validation fails.
    - If validation is successful, API retrieves all orders and returns them in its response.
- Retrieve Authenticated User’s Orders Workflow
    - A GET request containing a JWT in its header is sent to the /users/myOrders endpoint.
    - API validates user is NOT an admin via JWT, returns false if validation fails.
    - If validation is successful, API retrieves orders belonging to authenticated user and returns them in its response.
- Set User as Admin workflow

- An authenticated admin user sends a PUT request containing a JWT in its header to the /:userId/setAsAdmin endpoint.
- API validates JWT, returns false if validation fails.
- If validation successful, API finds user with ID matching the userId URL parameter and sets its isAdmin property to true.