FACULTY/COLLEGE		College of Business and Economics	
SCHOOL		School of Consumer Intelligence and Information Systems	
DEPARTMENT		Applied Information Systems	
CAMPUS(ES)		APB	
MODULE NAME		Development Software 2A	
MODULE CODE		DSW02A1	
SEMESTER		First	
ASSESSMENT OPPORTUNITY, MONTH AND YEAR		Assessment: Semester Test 2 May 2024	
ASSESSMENT DATE	15 May 2024 @ 09:00	**SUBMISSION DATE**	15 April 2024 @ 17h00
ASSESSOR(S)	Mr Ronny Mabokela		
MODERATOR	Prof. Mpho Primus		
DURATION	8 hours	**TOTAL MARKS**	100
NUMBER OF PAGES OF QUESTION PAPER (Including cover page)			11

INFORMATION/INSTRUCTIONS:

- ✓ Read instruction carefully to avoid making mistakes.
- ✓ This is NOT an open-book assessment.
- ✓ There are 4 questions and answer all questions.
- ✓ Submit the necessary files for codes where necessary.
- ✓ Read the questions carefully and answer only what is required.
- ✓ Make sure that your submission is in a zipped file.
- ✓ **Do not share the question paper in the WhatsApp group.**
- ✓ **Each student must access the question paper from Moodle.**

QUESTION 1 **[25 MARKS]**

Design a comprehensive PHP web service named **Ama2000.php** tasked with processing data comprising the names of babies born in the early 2000s. This service is responsible for reporting the best (lowest) popularity ranking that a name has ever held in the South African Home Affairs database.

Create a service which accepts a GET request parameter called name with the following cases:

- The output should consist of a single line of plain text containing the best ranking number, or -1 if no name parameter is passed or if the name is not found in the file.
- The input file to be read, **ama2000.txt**, follows a specific format. Each line contains a baby's first name followed by one or more popularity rankings.
- The number of rankings per line can vary and should be accommodated by your code.

The sample data below has 11 rankings per line, but for full credit your code should work regardless of how many rankings (1 or more) are on each line assigned to a name.

Bontle 31 25 24 26 30 46 93 163 209 289 382
Bonang 66 79 84 94 93 78 70 108 127 142 177
Bohlale 631 752 712 664 720 933 636 752 680 856 918
Martha 44 55 16 66 898 977 98 567 38 89 90 128 678 890
Martin 69 58 34 68 45 76 28 78 98 90 78 87 84 83 85 97
Mart 55 88 09 87 54 56 78 65 78 65 76 34 76 89
Lerato: 27 35 48 56 62 71 83 102 117 126 139
Thando: 94 87 76 62 51 42 39 47 55 64 73 85
Nomvula: 521 498 467 438 407 389 366 342 319 297 278 261
Kagiso: 77 63 51 46 39 32 25 19 14 12 9
Sibusiso: 154 139 127 115 102 89 77 64 51 39 27 14
Thabo: 61 49 37 25 16 12 8 5 3
Neo: 32 28 24 21 17 13 10 8 6 4 2
Noluthando: 342 326 301 278 256 234 212 189 173 157 141 126
Aphiwe: 82 75 68 62 57 51 46 41 38 35 31
Nthabiseng: 453 438 422 407 392 378 364 351 337 323 309 296

.....

For example, if your service were requested as **Ama2000.php?name=Bontle**, its output would be **24**, since that is Martha's best popularity ranking.

For instance:

- A request to **Ama2000.php?name=Bontle** should return 24, as that is the best popularity ranking for the name "Bontle".
- A request to **Ama2000.php?name=Bonolo** should return -1 since that name is not present in the file.
- A request to **Ama2000.php?name=66** should return -1 since numerical values are not valid names.
- A request to **Ama2000.php?name=B** should return -1 since "B" is not a complete name.
- Assume the name "John" has the ranking sequence: 88 16 34 16 590 67 265 987 657 87 545. A request to **Ama2000.php?name=John** should return 16 since the ranking 16 appears twice in the file.

Ensure that your code matches case-insensitively; for example, the request **Ama2000.php?name=BoHLalE** should match the name Bohlale in the data file.

QUESTION 2 **[25 MARKS]**

In recent years, cybersecurity breaches have become a significant concern, especially with the increasing reliance on digital platforms for sensitive data transactions. A study reveals that 35% of online users have experienced security breaches due to weak passwords. Despite this, 85% of respondents reported using two-factor authentication. Password managers and cybersecurity software are increasingly recognised as vital tools for managing passwords and preventing unauthorized data leaks, particularly for users with limited cybersecurity knowledge. In 2022, the Gauteng Department of Health's local government website was compromised by hackers. The breach resulted in the potential exposure of personal information, which may have been transferred outside the organization. The department is currently investigating the extent of the accessed data, expressing grave concerns about the data's confidentiality and integrity. As a developer and cybersecurity analyst for the Gauteng Department of Health, you are tasked with developing a secure registration system that validates user ID numbers. The system should adhere to stringent validation rules to enhance security and data integrity.

You need to build a secure registration system that can validate user inputs through a secure HTML form. The system should ensure the robustness of data protection measures. Write PHP code that can process the following HTML form:

PHP Requirements:

1. **Validation Rules:**
   - **Name:** Must be a non-empty string without any numbers.
   - **Password:** Must be at least 12 characters long, containing both characters, special and numbers.
   - **ID Number:** Must contain exactly 13 digits and can include dashes but not dots.
   - It validates the **name**, **password**, and **ID number** fields according to the specified rules.
2. **Security Measures:**
   - Implement CSS styling to visually differentiate successful and unsuccessful validation. If all validations pass, it displays "Successful" in green and bold, along with the user's name, masked password, and cleaned ID number.
   - If any validation fails, it displays "Access Denied - Invalid Data" in red and bold, along with the specific validation messages.
   - Perform server-side validation to ensure data integrity.
   - Display feedback using proper HTML structure with security considerations.
3. **Output Requirements:**
   - Display a Level 1 heading indicating "Successful" or "Access Denied - Invalid Data" in green or red, respectively.
   - Show user details (name and ID number) in a paragraph.
   - Mask the password with asterisks (*) while preserving its length.
   - Remove any dashes from the ID number for the output.

For example, the following are some examples of valid and invalid ID numbers:

Valid	Invalid
123456786544 245-154-873-888 394850980323 983 434 259 900 111222333444	2457.1543.5676 foo123456doo 123.4567810. 1234-5678:8986 102:304:506:658

For example, some outputs of your PHP script for form input:

Form Input	Output
name: Bontle password: Bon123! ID: 245-154-873-888	**Successful.** Bontle, ********, 245154873888
name: John password: mnnr@430 ID: 111/112'222	**Access Denied! Invalid data.** John, ********, 111112222
Name: Thabang password: Thab$$066 ID: foo123456doo	**Access Denied! Invalid data.** Thabang, ********, foo123456doo

Form Input	Output
name: 12345 password: Bon123! ID: 123-456-786	**Access Denied! Invalid data.** 12345, ********, 12345678
name: John password: mnmmr ID: 1111122222	**Access Denied! Invalid data.** John, ********, 111112222
Name: Thabang password: $$066 ID: foo123456doo	**Access Denied! Invalid data.** Thabang, ********, foo123456doo

Use regular expressions for pattern matching, validation, and replacement. HTML content is stored in the file protectaccess.html and some resulting content is stored in the file accessresults.txt both of which you must place into your page.

QUESTION 3 [25 MARKS]

Write the PHP code for a web page **electcamp_donations.php** that displays donors who have donated amounts in a specified range during the South African Presidential Candidate Campaign in 2024. Assume that there is a provided page with a form where the user can type the name of a donor and select a donation level. The form will submit to your electcamp_donations.php. Here is the relevant HTML from that provided page, and a screenshot of its appearance:

```
<h1>Donation Search</h1>
 <form action="electcamp_donations.php" method="get">
 <div>
 <input type="radio" name="level" value="bronze"/> bronze
 <input type="radio" name="level" value="silver"/> silver
 <input type="radio" name="level" value="gold"/> gold
 </div>
 <div><input type="text" name="donor" /> Donor name</div>
 <div><input type="submit" value="View donations" /></div>
 </form>
```

Each donor has a corresponding text file in the current directory that contains all donations for that donor.

- The file may not have the exact name of the donor, but it will contain the exact name.
- For example, if the user types "CR24" it should match "cr24forpresident.txt" or "JSM24" it should match from "jsm24forpresident.txt".
- If there is more than one file containing the donors' names, you should select the first one.
- Each line in the file contains the donator's name, the donation amount, "yes" if the donation was matched, and "no" otherwise.
- Each item is separated from the others by a semicolon (":"), as in the example below at right.

The bronze donation range amounts between 0m and 1m inclusive, silver between 2m exclusive and 5m inclusive and gold above 5m and 40m inclusive.

Your PHP page should accept the query parameters from the above form, open the text file for the correct donor, and search it for donations in the range matching the selected category.

- Display each donation as a bullet in an unordered list, showing the donor and the amount separated by a dash (-). At the bottom of the page output the total amount donated from this group.
- If the donation was matched, the amount donated should be added to the total twice.

CR24 for president: Donors

De Bears: R5m:no
Patrice Motsepe: R2m: yes
Nicky Oppenheimer: R10m: yes
Sifiso Dabengwa: R1m: yes
Maria Ramos: R10m: yes
Colin Coleman: R3m: yes
Mark Lambert: R5m: yes
Raymond Ackerman: R1m: yes
Julius Malema: R2m:no
Ruppert family: 30m: yes

JSM24 for president: Donors

Crony and friend: R1m: yes
Ace Magashule: R2m: yes
IEC: R37m: yes
Guptas: R10m: yes
Tom Moyani: R1m: yes
Batho Batho Trust: R5m: yes
DSW Students: R500: no
Zuma and Sons: R5m: no

You may assume that the user will submit parameters with the appropriate values (e.g. the level will always be gold, silver, or bronze) if they submit parameters. However, they may omit parameters.

- If they do your code should print an error message explaining what went wrong.
- You may assume that a file for the donor the user submits exists and is valid in the format described above.
- You can write just the code that would go inside the page body; you do not need it.

The following screenshots show the PHP page output after the user submits the form with various values:

1. *Bronze – donor: CR24 for president* • *Raymond Ackerman - 1m* • *Sifiso Dabengwa – 1m* ***Total: R2m***	2. *Silver – donor: CR24 for president* • *Patrice Motsepe - R2m* • *Colin Coleman - R3m* • *Mark Lambert - R5m* ***Total: R10m***
2. *Bronze – donor: JSM for president* • *Crony and friends - R1m* • *Tom Moyani - R1m* ***Total: R2m***	3. *No level selected* *Please, specify both the donor's name and category level*

QUESTION 4 **[20 MARKS]**

At JBS Books Commerce, we simplify the complexity, time, and effort required in the bookselling business through our specialized software. Our solution helps merchants effortlessly manage orders, inventory, and fulfilment, allowing them to sell more books to more customers via online marketplaces. This software manages asset collections and relationships with members, aiding libraries in tracking books, checkouts, subscriptions, and profiles.

As an enhancement to the existing system, you are tasked with developing a web page, **jbsbooksco.php**, which allows users to search for books available for sale under a specified maximum price. Users can input the category of books to search and the maximum price. Additionally, you need to account for special pricing for students, SANDF and pensioners.

Below is the HTML code provided to a page:

```
<h1>JBS Library Book Search</h1>
 <form action=" jbsbooksco.php " method="get">
 <div><input id="bookcat" type="text" name="category" /> Category</div>
 <div><input id="bookmax" type="text" name="max" /> Maximum Price</div>
 <div><input id="primecheck" type="checkbox" name="prime" /> Pensioner? </div>
<div><input id="studentcheck" type="checkbox" name="prime" /> Student? </div>
<div><input id="sandfcheck" type="checkbox" name="prime" /> sandf? </div>

 <div><input id="submitview" type="submit" value="View Books" /></div>
 </form>
```

✓ Each line of these files contains a book name followed by a colon (:) and the price of the book.

For example, below are **adventure.txt** and **novel.txt**:

Adventure.txt	Novel.txt
Into thin air: R455 The hunt for Z: R600	The hate you give: R300. The silent wife: R289

A summer adventure: R349 Lands of lost bodies: R800 The black stallion: R289	Something I never told you: R340. The beginning and end of us: R500 The reason is you: R589

1. **HTML Form:**
   - The form includes fields for category and maximum price, as well as checkboxes for "Pensioner?", "SANDF?" and "Student?" discounts.
   - The category input is required, and the maximum price input is a number with a minimum value of 0.
2. **Input Validation**:
   - Sanitize user inputs to prevent injection attacks.
   - Validate the category to ensure it only contains alphabetic characters.
   - Validate the maximum price to ensure it is a positive number.
   - Ensure that the maximum price is not excessively high to prevent performance issues.
3. **PHP Script:**
   - The script processes the form data upon submission (**GET** request).
   - It reads the category, maximum price, and discount options.
   - Converts the category to lowercase to match the file naming convention.
   - Checks if the corresponding category file exists. If not, it displays "Category not found – In Library".
   - If the file exists, it reads the file line by line, splitting each line into book title and price.
   - Applies a 50% discount for students, 30% for SANDF and a 40% discount for pensioners if applicable.
   - If both checkboxes are selected, apply the student discount only.
   - Filters books by the given maximum price and adds matching books to an array.
   - Displays the matching books as a bulleted list, or a message if no books match the criteria.
   - Display a count of the books found.

4. **Output:**
   - Displays matching books in an unordered list, with the price and title, showing the price and the title separated by a dash (-) or double colon (:).
   - Match is not case sensitive; the user might submit the category in any capitalization, but the input files are always in lower case.
   - Display a count of the books found.
   - Shows the number of books found or a message indicating no books were found within the price range.
   - The form should not produce any errors or warnings in case of incorrect input.

Sample search:

Novel, max price: 400	**Novel, max price: 350, Student?**	Category "romance"
• The hate you give: R300. • The silent wife: R289 • Something I never told you: R340. *3 found*	• The hate you give: R150. • The silent wife: R144.5 • Something I never told you: 170. • The beginning and end of us: R250 • The reason is you: R294.5 *2 found*	Category not found – In Library *0 found*
Adventure, max price: 350 • A summer adventure: R349 • The black stallion: R289 *2 found*	• **Adventure, max price: 600, Student?** • Into thin air: R227.5 • The hunt for Z: R300 • A summer adventure: R174.5 • The black stallion: R144.5 • Lands of lost bodies: R400 *4 found*	• **Adventure, max price: 100** *0 found*

NO LATE SUBMISSION
COPYING IS NOT ALLOWED
NO EMAIL SUBMISSION
WRONG FILES = ZERO MARK