National Institute of Business Management
School of Computing and Engineering
Course work | Assesement Announcement Sheet

Course Name	**BSc (Hons) Computing-23.2**		
Module Name	PDSA		
Batch	23.2		
Learning Outcomes Covered (Mention according to the Module Descriptor)	1. Understand and select appropriate algorithms for solving a range of problems and reason about their complexity and efficiency. 2. Design and implement algorithms and data structures for novel problems. 3. Specify and implement methods to estimate solutions to intractable problems		
Assesement | CW No	1		
Assesement Mode	**~~Individual~~ | Group**	**Group (if it is group mode only)**	
		Group Size	**Grouping Criteria**
		5	Students can select the group members
Assesement Type	**Practical Test | Report | Software | Presentation | ~~VIVA | MCQ~~**		
	If other specify		
Hand in Date | Time	12/03/2024		
Hand out Date | Time	06/04/2024		
Submission Details (Format and Location)	NIBM Moodle		
Plagiarism Criteria			

Assesement | CW Description

1. Create a game menu option called Eight queens' puzzle:

- Eight chess queens must be placed on an 8x8 chessboard in such a way that no two queens threaten each other
- Using a program, find the maximum number of solutions for this and save them as answers in the system.
- Allow game players to provide answers using a user interface.
- When a game player correctly identifies an answer, save that person's name along with the correct response in the database.
- Select appropriate Application Data Structures & algorithm to determine correct answer.
- If another game player provides the same right response, indicate that the solution has already been recognized. Ask them to try again until the maximum number of solutions has been achieved.
- When all the solutions have been identified by game players, the system should clear the flag that indicate solution has already been recognized. so that future game player can give same answers.
- When a game player correctly identifies an answer, save that person's name along with the correct response in the database.
- Include the Code used for unit Testing.
- Identify the Complexity of the Algorithm & Give a brief Description about the algorithm used to identify the maximum number of solutions.
- Create a threaded application to identify the maximum number of solutions & Compare the time taken for the Sequential program & the Threaded program.

2. Create a game menu option called Tic-Tac-Toe

- Implement a Human vs. Computer Tic Tac Toe Game
- Determining the optimal Tic-Tac-Toe move for a computer player using appropriate Application Data Structures & algorithm
- Allow game players to provide answers using a user interface
- Provide user interface when players win, lose, or draw a game
- When a game player correctly identifies an answer, save that person's name along with the correct response in the database.
- Include the Code used for unit Testing

3. Create a game menu option called Identify Shortest Path

- Each game round, the distance between cities is assigned at random between 5 and 50 kilometers. (Refer Figure 1)
- In each game round ask game player to find the shortest path and distance for other cities from the system's randomly selected city. Allow game players to provide answers using a user interface.
- Determine whether the user response is correct or incorrect by utilizing the Bellman-Ford and Dijkstra algorithms to find the right answer. Additionally, record the duration of each algorithm in a separate database table.
- Save a game player's name, answer, and the distance between cities in the database when they correctly identify an answer.
- Include the Code used for unit Testing

City A	---								
City B	random Distance	---							
City C	random Distance	random Distance	---						
City D	random Distance	random Distance	random Distance	---					
City E	random Distance	random Distance	random Distance	random Distance	---				
City F	random Distance	random Distance	random Distance	random Distance	random Distance	---			
City G	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	---		
City H	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	---	
City I	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	---
City J	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance	random Distance
	City A	City B	City C	City D	City E	City F	City G	City H	City I

Figure 1

4. Create a game menu option called Remember the Value Index

- Each game round, generate 5000 random numbers between 1 to 1000000
- Implement the logic to Sort the numbers using bubble sort, insertion sort, merge sort, radix sort, shell sort, quick sort & tim sort & record the time taken for each sort method in the database
- Request the player remember the value positions so that they can be entered into the index when the game asks. Display the 1st 20 numbers from the sorted list one at a time to the user , every two seconds. Next, choose two values at random from 1st 20 numbers, ask the user to enter the index. If user correctly identified the Answer save that person's name along with the correct response in the database

5. Create a game menu option called Predict the Value Index

- Each game round, generate 5000 random numbers between 1 to 1000000
- Implement the logic to search a number using Binary Search, Jump Search, Exponential Search, Fibonacci Search
- Search values from 1 to 1000000. In each scenario record the time taken for each search method & the index values returned from each search algorithm in the database.
- Select a Random number from 1 to 1000000. Ask user to Predict the index of that Value in the List by giving 4 choices. If user correctly identified the Answer save that person's name along with the correct response in the database

DELIVERABLES: The students should submit.

1. Software with Source Code
2. Database with Data Dump
3. Report including: (1500-word individual written report. Written in academic style with appropriate referencing)
   - 3.1. Eight queens' puzzle
     - 3.1.1. Explain the Program Logic used to solve the problem
     - 3.1.2. UI screenshot allowing game players to provide answers
     - 3.1.3. UI screenshots when game players to provide correct answer & incorrect answers
     - 3.1.4. Explain the Validations and Exception Handling in this application.
     - 3.1.5. Code Segment screenshots: unit Testing
     - 3.1.6. Indicate the Data Structures used and Explain its purpose
     - 3.1.7. Screenshot of the Normalized DB Table Structure used for this Game Option
     - 3.1.8. Compare the time taken & logic used for the Sequential program & the Threaded program.
   - 3.2. Tic-Tac-Toe
     - 3.2.1. Explain the Program Logic used to solve the problem
     - 3.2.2. UI screenshot allowing game players to provide answers
     - 3.2.3. UI screenshots when game players to provide correct answer & incorrect answers
     - 3.2.4. Explain the Validations and Exception Handling in this application.
     - 3.2.5. Code Segment screenshots: unit Testing
     - 3.2.6. Indicate the Data Structures used and purpose
     - 3.2.7. Screenshot of the Normalized DB Table Structure used for this Game Option
   - 3.3. Identify Shortest Path
     - 3.3.1. Explain the Program Logic used to solve the problem
     - 3.3.2. UI screenshot allowing game players to provide answers
     - 3.3.3. UI screenshots when game players to provide correct answer & incorrect answers
     - 3.3.4. Explain the Validations and Exception Handling in this application.
     - 3.3.5. Code Segment screenshots: unit Testing
     - 3.3.6. Indicate the Data Structures used with its purpose
     - 3.3.7. Screenshot of the Normalized DB Table Structure used for this Game Option
     - 3.3.8. Compare the Shortest path identifying algorithms (logic / complexity, which scenarios are best to use it, etc.)
   - 3.4. Remember the Value Index
     - 3.4.1. UI screenshot allowing game players to provide answers
     - 3.4.2. UI screenshots when game players to provide correct answer & incorrect answers
     - 3.4.3. Chart Containing the time Taken for each Sorting Technique
     - 3.4.4. Compare the sorting techniques (e.g., explain the logic used in the sorting Techniques, determine their complexity, which scenarios are best to use it, etc.).
       bubble sort, insertion sort, merge sort, radix sort, shell sort, quick sort & tim sort
   - 3.5. Predict the Value Index
     - 3.5.1. UI screenshot allowing game players to provide answers
     - 3.5.2. UI screenshots when game players to provide correct answer & incorrect answers
     - 3.5.3. Chart Containing the time Taken for each search Technique
     - 3.5.4. Compare the search techniques (e.g., explain the logic used in the search Techniques, determine their complexity, which scenarios are best to use it, etc.).
       Binary Search, Jump Search, Exponential Search, Fibonacci Search
   - 3.6. Git Hub Link
4. Small Video Clip indicating the features of the games