# Structured Programming 2017-2018

# Projects Description Template

| | Connect Four (2 Players) |
|---|---|
| **Description** | This program is developed to allow two users to play Connect Four. Supposed that each player plays by choosing the coordinates (x and y) or cell number then the program should update the selected cell by (1) for Player One or (2) for Player Two. |
| **Team** | 4- 6 members |
| **Deliverables** | • Board size should be 8 * 8 cells.<br>• The program should only accept coordinates in the right positions, i.e. the program should NOT fill cell (1, 1) unless the cell (1, 0) is filled, so this should be handled by an error message and let the player enters the coordinates again.<br>• The player is considered a winner if four cells are connected together horizontally, vertically or diagonally.<br>• The program should display a message if<br>1. Player 1 wins.<br>2. Player 2 wins.<br>3. None of the players has won. |
| **Bonus** | • Graphical User Interface (GUI).<br>• Viewing a list of all available positions to play.<br>• Saving in a text file the history of the game between player one and player two, This history should include the following:<br>1. All played games number.<br>2. The winner of each played game.<br>3. Date and Time of each played game.<br>4. The total score along with the winner. |
| **Mentor** | |
| **Notes** | - |

| | **Company Profile** |
|---|---|
| **Description** | This project aims to develop a system for a company that provides projects and a list of news associated with each project.<br>Project Entities:<br>1. Project: ID, Title, Start Date, End Date, Brief.<br>2. News: ID, Title, Publishing Date, Brief, Project ID. |
| **Team** | 4 – 6 members |
| **Deliverables** | 1. Creating Project:<br>• Project ID is an auto incremented value and not entered by the user.<br>• The user enters Title, Start Date, End Date and Brief.<br>2. Listing all Projects in the system.<br>3. Listing projects filtered by the user, filtration options are:<br>• Start Date: list projects that starts from specific date entered by the user.<br>• End Date: list projects that ends by specific date entered by the user.<br>4. Creating News:<br>• News ID is an auto incremented value and not entered by the user.<br>• The user enters Title, Publishing Date, and Brief, then list all projects with their IDs to select the project ID associated to the news.<br>5. Listing all News and for each news item display project title.<br>6. Listing news filtered by the user, filtration options are:<br>• By Project: list all projects Id and title so that the user selects project Id to display news in this project.<br>• By Title: list all projects whose title contains a specific keyword entered by the user. |
| **Bonus** | 1. Graphical User Interface (GUI).<br>2. Saving data in a file or more so that when closing the program and reopening it the data still exist. |
| **Mentor** | |
| **Notes** | - |

| | **MineSweeper Project** |
|---|---|
| **Description** | Is a single-player puzzle board game. The objective of the game is to clear a rectangular board containing hidden "mines" or bombs without detonating any of them, with help from clues about the number of neighboring mines in each field.<br>The player is initially presented with a grid of undifferentiated squares. Some randomly selected squares, unknown to the player, are designated to contain mines.<br>The game is played by revealing squares of the grid by indicating each square. If a square containing a mine is revealed, the player loses the game. If no mine is revealed, a digit is instead displayed in the square, indicating how many adjacent squares contain mines; if no mines are adjacent, the square becomes blank, and all adjacent squares will be recursively revealed. The player uses this information to deduce the contents of other squares, and may either safely reveal each square or mark the square as containing a mine. |
| **Team** | 4-6 |
| **Deliverables** | << Minimum requirements of the projects >><br>1. The Game board is 10X10 board with 10 mines the fulfills the description.<br>2. To win the Game, the player either revel all the free squares or Mark all the mines.<br>3. the Game at the end revel A scoreboard of the best 10 records.<br>4. Each record contains playerName and his score.<br>7. The scoreboard must be saved in file. |
| **Bonus** | • Add levels of Difficulty<br>• A windows forms Gui<br>• Any Extra functionality. |
| **Mentor** | |
| **Notes** | |

| | **Conference Halls Reservation System** |
|---|---|
| **Description** | The aim is to manage reserving conference halls to do events easily. The System should contain details of each hall (hall name, place, List hall events) so users be able to reserve it in available times. Each User should access his account by name and password where he can check his old reservations and reserve new ones. Every event should be defined by (id, name, startDate, EndDate). |
| **Team** | 4-6 |
| **Deliverables** | << Minimum requirements of the projects >><br>The System should:<br>1. Add or delete Hall<br>2. Each Show Record consist of (showname, start Date, EndDate, available seats In each day) where each day has max 100 seat.<br>3. Show all Shows available in Certain timeline.<br>4. Show All the details of selected Hall including List of events.<br>5. Search the event in which hall<br>6. Users access their accounts by name and password.<br>7. Users can see their reserved upcoming Events<br>8. Users can make reservation on any available hall.<br>9. User can cancel the reservation he made.<br>10. Only reservation is made when the hall has no events in that time.<br>11. Each Event consists of (id, name, StartDate, EndDate).<br>12. Each User account consist of (name ,password , the reserved events' ID) |
| **Bonus** | • Graphical User Interface (GUI).<br>• Any Extra Functionalities/Modules that enhance the system. |
| **Mentor** | |
| **Notes** | |

| | **Snake and Ladder Game Project** |
|---|---|
| **Description** | Snakes and Ladders is board game played between two or more players on a gameboard having numbered, gridded squares. A number of "ladders" and "snakes" are on the board, each connecting two specific board squares. The object of the game is to navigate one's game piece, according to die rolls, from the start (bottom square) to the finish (top square), helped or hindered by ladders and snakes respectively. |
| **Team** | 4-6 |
| **Deliverables** | << Minimum requirements of the projects >><br>1. The Game adds the Ladders and snakes randomly on board with condition not to collide.<br>2. The Game starts with selecting how many players.<br>3. The Game Board is 10X10 board with 5 ladders and 5 snakes<br>4. all the players pieces Start at square 1.<br>5. Each player rolls the dice and move to the new square.<br>6. Through the hall game a score board is shown where each record has player name and place on board.<br>7. With Every new move the score board is sorted. |
| **Bonus** | • Add levels of Difficulty<br>• A windows forms GUI<br>• Any Extra functionality. |
| **Mentor** | |
| **Notes** | |

| | **Course Management System** |
|---|---|
| **Description** | The system aims to manage courses and students in a school. Data about course shall be (ID, Name, Instructor Name, #Enrolled Students). Data about student shall be (ID, Name). Data shared between the course and student shall be ( Course Id, Student Id, Grade) |
| **Team** | 4-6 Members |
| **Deliverables** | System Functionalities:<br>1-Enroll new student to a specific course.<br>2- Edit, delete, and display the student information.<br>3- Add, Edit, Display and deletes course.<br>4- Search for a course by ID then display all students enrolled in this course all with his/her grade in the course.<br>5- Search for a student by ID then displays the courses he is enrolled in if exists. |
| **Bonus** | Graphical User Interface.<br>Any extra functionality. |
| **Mentor** | |
| **Notes** | 1- Arrays and Structures must be used.<br>2- Data of courses and students should be stored in two separate files (if additional files required make them).<br>3- Data of students shall be loaded before searching for a student.<br>4- Stored Courses shall be loaded at the beginning of the program.<br>5- A list of predefined courses shall be shown to user when enrolling a new student. |

| | **Library Search System** |
|---|---|
| **Description** | “Library Search System” is an application that can be used at the front desk of any library to help visitors find their books of interest. Assume that a library has a number of different books, each stored as a text file such that the text file’s name refers to the book’s title. The user will enter a search word (e.g.: “technology”), then the application will display the number of books containing this word and their titles ordered in descending order according to the number of repetitions of this word within the book. The application should allow the user to search with composite search key (e.g.: “computer science”) so you also need to store the position of the word. |
| **Team** | 4-6 Members |
| **Deliverables** | System Functionalities:<br>1- All the text book files should be parsed at the beginning and indexed using an appropriate data structure.<br>2- The user can search by single key word.<br>3- The user can search by composite key word. |
| **Bonus** | Graphical User Interface.<br>Any extra functionality. |
| **Mentor** | |
| **Notes** | Arrays and Structures must be used. |

| | **Pacman** |
|---|---|
| **Description** | The player navigates Pac-Man through a maze containing various dots, known as Pac-Dots, and four multi-colored ghosts: Blinky, Pinky, Inky, and Clyde.<br>The goal of the game is to accumulate points by eating all the Pac-Dots in the maze, completing that 'stage' of the game and starting the next stage and maze of Pac-dots.<br>If any of the ghosts hit Pac-Man, he loses a life; when all lives have been lost, the game is over. |
| **Team** | 4-6 members |
| **Deliverables** | • A Pac-Man that can be moved with keyboard arrows.<br>• The Pac-Man can eat dots , and as a result the score increases.<br>• If their is no more dots the game ends. |
| **Bonus** | • Four multi-colored ghosts: Blinky, Pinky, Inky, and Clyde , If any of the ghosts hit Pac-Man, he loses a life.<br>• when all lives have been lost, the game is over. |
| **Mentor** | |
| **Notes** | --- |

| | **Linked List Data Structure** |
|---|---|
| **Description** | Implement linked list and its functions ( insert , search ,delete ) |
| **Team** | 4-6 members |
| **Deliverables** | • linked list data structure implementation<br>• Insert function<br>• Search Function<br>• Delete Funtion<br>• The using of the linked list and the calling of its functions. |
| **Bonus** | • Sort Function |
| **Mentor** | |
| **Notes** | --- |

|  | **Binary Search Tree Data Structure** |
|---|---|
| **Description** | Implement binary search tree and its functions ( insert , search ,delete ) |
| **Team** | 4-6 members |
| **Deliverables** | • Binary Search tree data structure implementation<br>• Insert function<br>• Search Function<br>• The using of Binary Search tree and the calling of its functions. |
| **Bonus** | • Delete function |
| **Mentor** |  |
| **Notes** | --- |

|  | **Budget Tracker** |
|---|---|
| **Description** | Write an application that keeps track of a household's budget. The user can add expenses, income, and recurring costs to find out how much they are saving or losing over a period of time. |
| **Team** | 4-6 |
| **Deliverables** | << Minimum requirements of the projects >><br>1. The User can add new expenses<br>2. The user can add new income<br>3. The system must have recurring costs (car fuel, house rent,etc…)<br>4. The user can add recurring cost.<br>5. List all recurring costs.<br>6. List all income sources.<br>7. The user can edit any values "income, recurring cost, etc.."<br>8. The user can delete any value.<br>9. Display to the user a quick view on the current status<br>10. Display a detailed status<br>11. allow the user to specify a date range and see the net flow of money in and out of the house budget for that time period.<br>12. All of these data must be saved in file. |
| **Bonus** | • Graphical User Interface (GUI).<br>• Any Extra Functionalities/Modules that enhance the system. |
| **Mentor** |  |
| **Notes** |  |

| | **Movie Store** |
|---|---|
| **Description** | Manage video rentals and controls when videos are checked out, due to return, overdue fees and for added complexity create a summary of those accounts which are overdue for contact. |
| **Team** | 4-6 |
| **Deliverables** | << Minimum requirements of the projects >><br>1. The User can add new customer<br>2. The user can add new movie, each movie has different fees and overdue fees<br>3. List all customers in the system<br>4. List all Movies in the system<br>5. List all rented movies in the system<br>6. Allow user to rent movie and specify the due date and the rental fees and overdue fees.<br>7. Allow user to return movie and calculate the customer fees.<br>8. Create a summary of those accounts which are overdue for contact.<br>9. Display the most rented movies<br>10. All of these data should be saved in files |
| **Bonus** | • Allow the user to rate specific movie and display the movies with the highest rates.<br>• Graphical User Interface (GUI).<br>• Any Extra Functionalities/Modules that enhance the system. |
| **Mentor** | |
| **Notes** | |

|  | **To Do list** |
|---|---|
| **Description** | Make an application which allows the user to enter his tasks and specify their priorities |
| **Team** | 4-6 |
| **Deliverables** | << Minimum requirements of the projects >><br>1. Allow the user to add new task (Title, details, due date and its priority)<br>2. Allow the user to mark the task as done.<br>3. Allow the user to edit specific task.<br>4. Allow the user to delete specific task<br>5. Display all the tasks.<br>6. Display all the Done tasks.<br>7. Display all the overdue tasks.<br>8. Display the tasks that need to be done today<br>9. Search for specific task by its title or number or timing<br>10. All of these data should be saved in files |
| **Bonus** | • Graphical User Interface (GUI).<br>• Any Extra Functionalities/Modules that enhance the system. |
| **Mentor** |  |
| **Notes** |  |

|  | Online Shopping Store |
|---|---|
| **Description** | The Project aims to serve both shoppers and shop owners to facilitate the shopping service. |
| **Team** | 4-6 Members |
| **Deliverables** | Shopper Functionalities:<br>1- Registration (Name – ID – Password – Age – Gender - Address).<br>2- Login (ID - Password).<br>3- Edit his/her data.<br>4- Search for a specific product.<br>5- View Products information.<br>6- Add items in a shopping cart.<br>7- View Receipt.<br>Shop Owner Functionalities:<br>1- Registration (Name – ID – Password – Age – Gender - Address).<br>2- Add shop data (Shop Name – Address - Category)<br>3- Login (ID - Password).<br>4- Edit his/her data.<br>5- Add new product (Product Name – Price – Category – Quantity).<br>6- Remove product.<br>7- View all shop products. |
| **Bonus** | Graphical User Interface.<br>Any extra functionality. |
| **Mentor** |  |
| **Notes** | Arrays and Structures must be used.<br>All data should be stored in files. |

| | ATM System |
|---|---|
| **Description** | The Project aims to build a system for ATM of a specific bank. ATM can handle withdrawal, deposition and credit inquiry transactions. Beside that ATM can serve bank employees by generating reports about daily transactions. |
| **Team** | 4-6 Members |
| **Deliverables** | ATM User Functionalities:<br>1- Log in (User Name - Password).<br>2- Deposit cash in his/her account.<br>3- Withdraw cash from his/her account.<br>4- Inquiry about his/her balance.<br>5- View report about his/her last 5 transactions.<br>Bank Employee Functionalities:<br>1- Generate report about daily transaction.<br>2- Search for all transactions for a specific user account. |
| **Bonus** | Graphical User Interface.<br>Any extra functionality. |
| **Mentor** | |
| **Notes** | Arrays and Structures must be used.<br>All data should be stored in files. |