### Group Members: Syed Muhammad Qammar Abbas Zaidi Syed Qalb-e-Abbas Kazmi Atif Waheed Project Report: ****Competition Scheduler for Cyber Security Hackathons****

#### ****1. Project Overview****

The **Competition Scheduler for Cyber Security Hackathons** is a comprehensive tournament management system designed to simulate and manage one-on-one knockout tournaments for 16 teams. The system features a graphical user interface (GUI), robust backend logic, and carefully selected data structures to ensure seamless functionality and a user-friendly experience. The project is structured to simulate a showdown-style hackathon, where teams compete in direct elimination rounds until a champion is declared.

#### ****2. Features and Functionalities****

##### ****Graphical User Interface (GUI)****

The GUI is designed with a minimalist and professional aesthetic using a combination of black, white, and blue color schemes. The main screens and their functionalities are:

**Main Menu (MyForm)**

* 1. Serves as the central hub for navigation.
  2. Buttons:
* **View Teams**: Displays a list of all participating teams and their ranks.
* **View Pools**: Displays teams divided into two pools based on their ranks.
* **Tournament**: Simulates and displays the tournament progression.
* **Match History**: Displays a record of all completed matches.

**Teams Overview (TeamsForm)**

* 1. Lists all participating teams with details such as team names, lead names, and ranks.
  2. Provides a search bar for filtering teams by name or rank.
  3. Allows users to view detailed information about a selected team.

**Team Details (TeamDetailsForm)**

* 1. Displays detailed information about a specific team, including:
* Team name.
* Team lead details (name, gender, contact, email).
* Member details (names, genders, contacts, emails).

**Pools Overview (PoolsForm)**

* 1. Divides teams into two pools:
* **Pool Alpha**: Teams ranked 1–8.
* **Pool Beta**: Teams ranked 9–16.
  1. Provides search functionality to locate teams in the pools.

**Tournament Progress (TournamentForm)**

* 1. Simulates the progression of the tournament across:
* **Knockout Round** (16 teams).
* **Quarter-Finals** (8 teams).
* **Semi-Finals** (4 teams).
* **Final** (2 teams).
  1. Displays results for each match and declares the champion.
  2. Supports manual or automatic mode for determining match winners.

**Match History (MatchHistoryForm)**

* 1. Displays a detailed log of all completed matches, including:
* Competing teams.
* Match winner.
* Date and time of the match.
  1. Includes a search bar for filtering matches by team names or winners.

#### ****3. Backend Logic****

##### ****Data Structures****

**Arrays**

* 1. Used to store team details and preserve original team data for pool resets.

**Queues**

* 1. Manage pools (Pool Alpha and Pool Beta).
  2. Handle remaining teams in each round.
  3. Store match history for retrieval and display.

**Binary Tree**

* 1. Represents the tournament structure.
  2. Each node corresponds to a match, linking winners to their next round.

##### ****Classes and Key Functionalities****

**Team Class**

* 1. Stores team details:
* Team name.
* Team lead details (name, gender, contact, email).
* Members' details.
* Rank.

**Match Class**

* 1. Represents a single match in the tournament.
  2. Stores:
* Participating teams.
* Match status (e.g., "Scheduled", "Completed").
* Winner.

**Scheduler Class**

* 1. Core logic for managing the tournament.
  2. Key Functions:
* Load team data from a file.
* Divide teams into pools based on ranks.
* Schedule and simulate matches for each round.
* Reset tournament progress.

#### ****4. System Flow****

**Data Loading**

* 1. Teams are loaded from an external file (teams.txt).
  2. File validation ensures exactly 16 teams are loaded with complete details.

**Pool Division**

* 1. Teams are divided into two pools based on their ranks:
* **Pool Alpha**: Teams ranked 1–8.
* **Pool Beta**: Teams ranked 9–16.

**Tournament Simulation**

* 1. Matches are scheduled in rounds:
* **Knockout Round**: Teams from Pool Alpha and Pool Beta compete.
* **Quarter-Finals**, **Semi-Finals**, and **Final**: Winners advance to subsequent rounds.
  1. Winners are determined randomly in automatic mode or manually in manual mode.

**Match History**

* 1. Completed matches are stored in a queue and displayed in the Match History screen.

**Tournament Reset**

* 1. Users can reset the tournament after completion, clearing all progress and reinitializing pools.

#### ****5. Implementation Details****

##### ****File Structure****

The project comprises 13 key files:

**Core Logic**

* Scheduler.h: Implements tournament logic.
* Match.h: Defines the Match class.
* Node.h: Implements nodes for the binary tree.

**GUI**

* MyForm.h: Main navigation screen.
* TeamsForm.h: Displays team list and search functionality.
* TeamDetailsForm.h: Displays detailed team information.
* PoolsForm.h: Displays pool division.
* TournamentForm.h: Simulates tournament progress.
* MatchHistoryForm.h: Displays completed match records.

**Data File**

* teams.txt: Contains team data (name, lead details, members, rank).

##### ****Key Functionalities****

* **Manual Mode**: Allows users to select match winners.
* **Search**: Supports searching by team names and ranks in various screens.
* **Error Handling**: Includes robust error handling for file loading and scheduling.
* **GUI Updates**: Dynamic updates ensure synchronization between backend logic and GUI.

#### ****6. Challenges and Resolutions****

**File Validation**

* + **Challenge**: Ensuring the file contains exactly 16 teams with complete details.
  + **Resolution**: Implemented strict validation logic during file loading.

**Dynamic Pool Management**

* + **Challenge**: Properly dividing and resetting pools.
  + **Resolution**: Used queues to dynamically manage pool teams.

**Tournament Progression**

* + **Challenge**: Ensuring winners propagate correctly through rounds.
  + **Resolution**: Implemented a binary tree structure for tournament representation.

**GUI-Backend Integration**

* + **Challenge**: Synchronizing GUI updates with backend logic.
  + **Resolution**: Designed modular components with clear data bindings.

#### ****8. Conclusion****

The **Competition Scheduler for Cyber Security Hackathons** successfully demonstrates the ability to manage and simulate a knockout-style tournament with 16 teams. It integrates robust backend logic with an intuitive GUI, offering a seamless user experience. The project adheres to the specified guidelines, making efficient use of data structures like arrays, queues, and binary trees. With its modular design and scalable architecture.