codesprint

NATIONAL CODING COMPETITION

Qualifiers Task

Post-Secondary Category 2026

POWERED BY mdia MALTA DIGITAL INNOVATION AUTHORITY

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION AND SPORTS
DIRECTORATE FOR STEM AND VET PROGRAMMES

× icecampus

Qualifiers Round Schedule

- Task Analysis (15 min)
- Task Development (60min)
- Interval (15 min)
- Continuation of Task Development (60 min)
- Task Submission (15 min)
- **Top 10 finalists will be announced**

The Wild Dominion

The wilderness is no longer wild. It is controlled and planned, and it is highly competitive.

In Wild Dominion, two opponents enter a simulated ecosystem where survival depends on smart decisions. In a boardgame-like grid, Eagles, Foxes and Rabbits compete for control. Every placement matters. Every move can shift the balance of power.

Will you think better than your opponent and build the strongest population, or will your animals disappear over time?

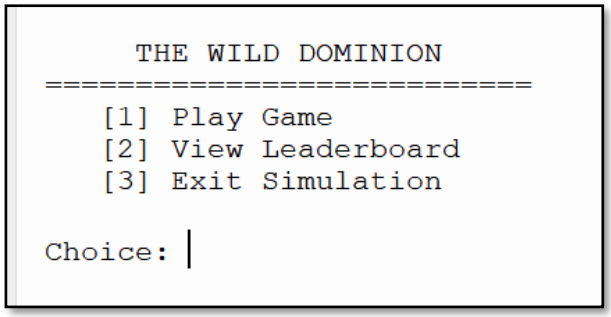
Your task is to build a Java or Python program that runs a turn-based battle between a user and the computer. The program may use either a text-based interface, OR a graphical user interface. The system must manage animal placements, movements, interactions, and environmental events across two separate grids.

Think carefully about how you structure your solution so every decision shapes what happens next.

Functionality #1: Program initialisation

When the program starts, it must

- Display the title: '**The Wild Dominion**'.
- Display a main menu allowing players to choose whether to start a new game, view the scoreboard, or exit.



Screenshot 1: Program initialisation visual sample

💡 **A text-based sample visual is shown in Screenshot 1.**

Functionality #2: Gameplay Initialisation

If the user chooses to play a game

- Ask the user to enter a username.
- Prompt the user to define the game grid dimensions (N)
- Create a N×N grid for the user, and another N×N grid for the computer.
- Specify the number of Eagles (E), Foxes (F), and Rabbits (R) available to each player, using the following formulas. Answers should be rounded to nearest unit.
  - E = $N^2/36$
  - F = $N^2/12$
  - R = $N^2/8$

💡 **For example:**
For a 6x6 grid (N=6, N^2=36): x1 Eagle, x3 Foxes, x5 Rabbits.
For an 9x9 grid (N=9, N^2=81): x2 Eagles, x7 Foxes, x10 Rabbits.

- For the first game turn, each player (user and computer) must place the number of established Eagles (E), Foxes (F), and Rabbits (R) on their respective grid.
- The user must select a valid position on the grid for each animal placement. *A position represents a single cell in the grid.*
- Input method may vary depending on interface. Users may choose animals placement by specifying the row and column, or by a mouse selection.
- The computer places animals randomly within the grid.

🔍 **Validation rules:**

- The username must contain letters only.
- The username must be at least 2 characters long, excluding spaces.
- The username should be unique and not included already in the `scoreboard.txt` file (*Check functionality 8*).
- Animals must be placed inside the grid. Invalid grid positions must not be accepted.
- Animals of the same player cannot be placed in the same position.

⚠ **Important:** Do not display grids until all animals for both players have been placed.

💡 **A text-based sample visual is shown in Screenshot 2.**

```
TURN 1

🎮 John's animals placement:
Eagle placed at (x,y): 3,2
Fox #1 placed at (x,y): 2,1
Fox #2 placed at (x,y): 3,7
❌ INVALID POSITION

Fox #2 placed at (x,y): 3,5
Fox #3 placed at (x,y): 5,4
Rabbit #1 placed at (x,y): 1,1
Rabbit #2 placed at (x,y): 6,1
Rabbit #3 placed at (x,y): 2,1
🚫 Cell occupied by a Fox
🔁 Please choose another position

Rabbit #3 placed at (x,y): 5,2
Rabbit #4 placed at (x,y): 4,3
Rabbit #5 placed at (x,y): 1,4
Rabbit #6 placed at (x,y): 6,6

💻 Computer placement complete
```

Screenshot 2: 1st turn sample visual

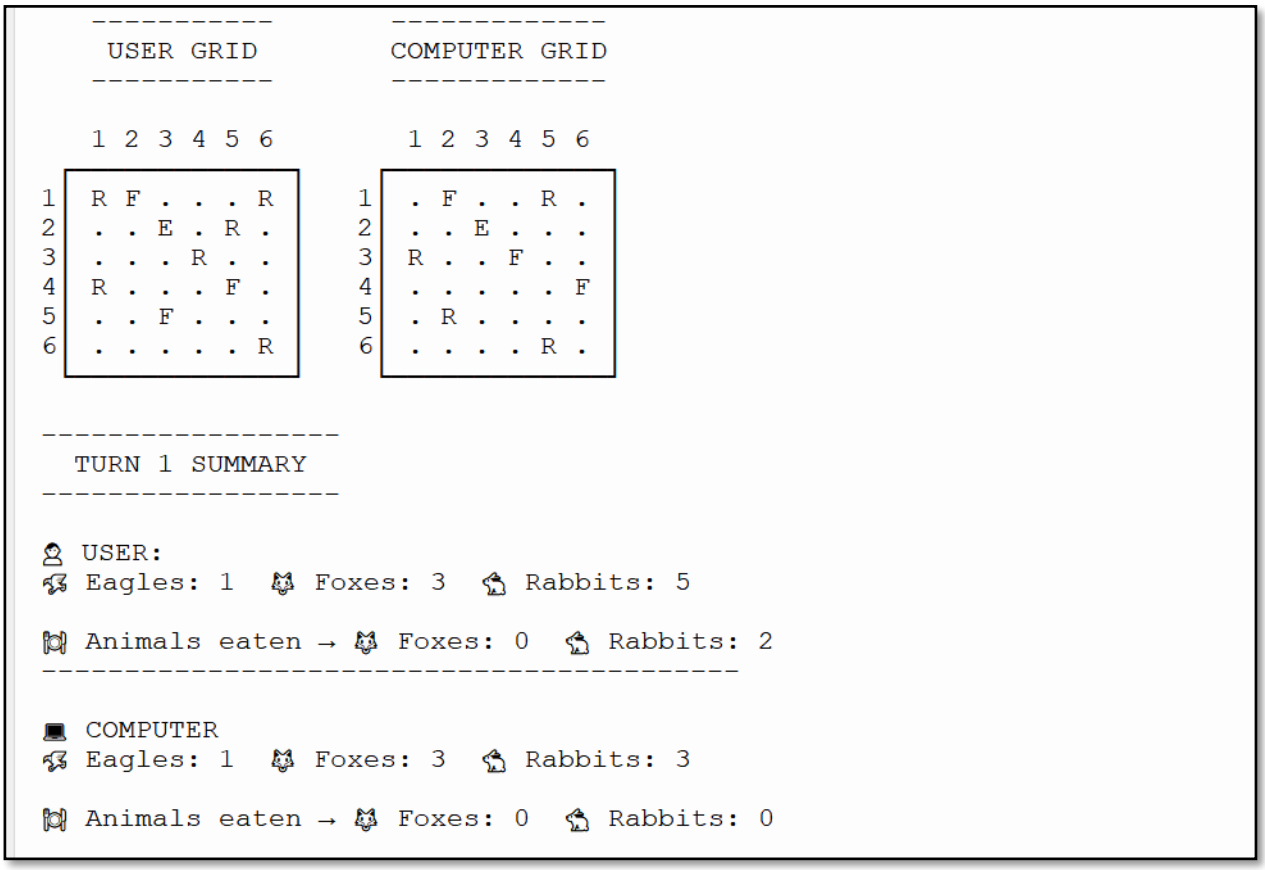
Functionality 3: Gameplay Sequence

The simulation runs for a maximum of 3 turns, including the first turn upon gameplay initialisation (Turn 1: Placement & Interactions; Turn 2: Movement & Interactions; and Turn 3: Movement, Interactions & Environment Event).

- The second and third turn consist of:
  - User move, as per functionality 4 below.
  - Computer move, as per functionality 4 below.
  - Possible predator interactions, as per functionality 5 below.
  - A possible environmental event, as per functionality 6 below.
  - Display the updated user and computer grids.
  - Display the status of the turn, including:
    - any predator interaction/s
    - any environment event
    - The number of animals remaining for each player

– Gameplay ends after 3 turns have been completed, or when at least one player has no Foxes and no Rabbits remaining.

💡 **A text-based sample visual is shown in Screenshot 3.**



Screenshot 3: 1st Turn Results Summary

Functionality #4: Animal Movement

– Starting from the second turn, both players (user and computer) may, or may not, move one or more than one animal.
– Each animal can move only **ONE** cell per turn in any direction.

🔍 **Movement Validation Rules:**

– Animals must stay within the grid. Invalid grid positions must not be accepted.
– Animals of the same player cannot move onto a cell occupied by another animal.
– Animals can only be moved once per turn.

⚠️ **Important:**

– The computer can choose to move some, all, or none of its animals randomly.
– Computer's chosen cell destination must be random; and not as a reaction to or depending on the user's moves.

💡 **A text-based sample visual is shown in Screenshot 4.**

```
┌──────────┐
│  TURN 2  │
└──────────┘

Do you want to move an animal? [Y/N]: y
Select animal to move (row,column): 3,1
⚠ There is no animal placed in this position

Select animal to move (row,column): 2,1
Select new position (row,column): 7,1
✗ POSITION NOT IN GRID

Select new position (row,column): 3,2
⦸ Cell already occupied by an Eagle

Select new position (row,column): 4,1
⚠ Cannot move more than one cell

Select new position (row,column): 2,2
✔ Fox moved from (2,1) → (2,2)
----------------------------------------

Move another animal [Y/N]: n
```

Screenshot 4: Visual sample of animal movement

Functionality 5: Predator Interactions

- When animals from both players occupy the same cell position (*same row and column*), interactions occur.
- If the animals are of the same species, both animals survive.
- If animals are not of the same species:
  - Eagle eats Fox and Rabbit;
  - Fox eats Rabbit;
  - Rabbit does not eat.
- Any animal that is eaten must be removed from the grid.

💡 These rules are applied after every turn.

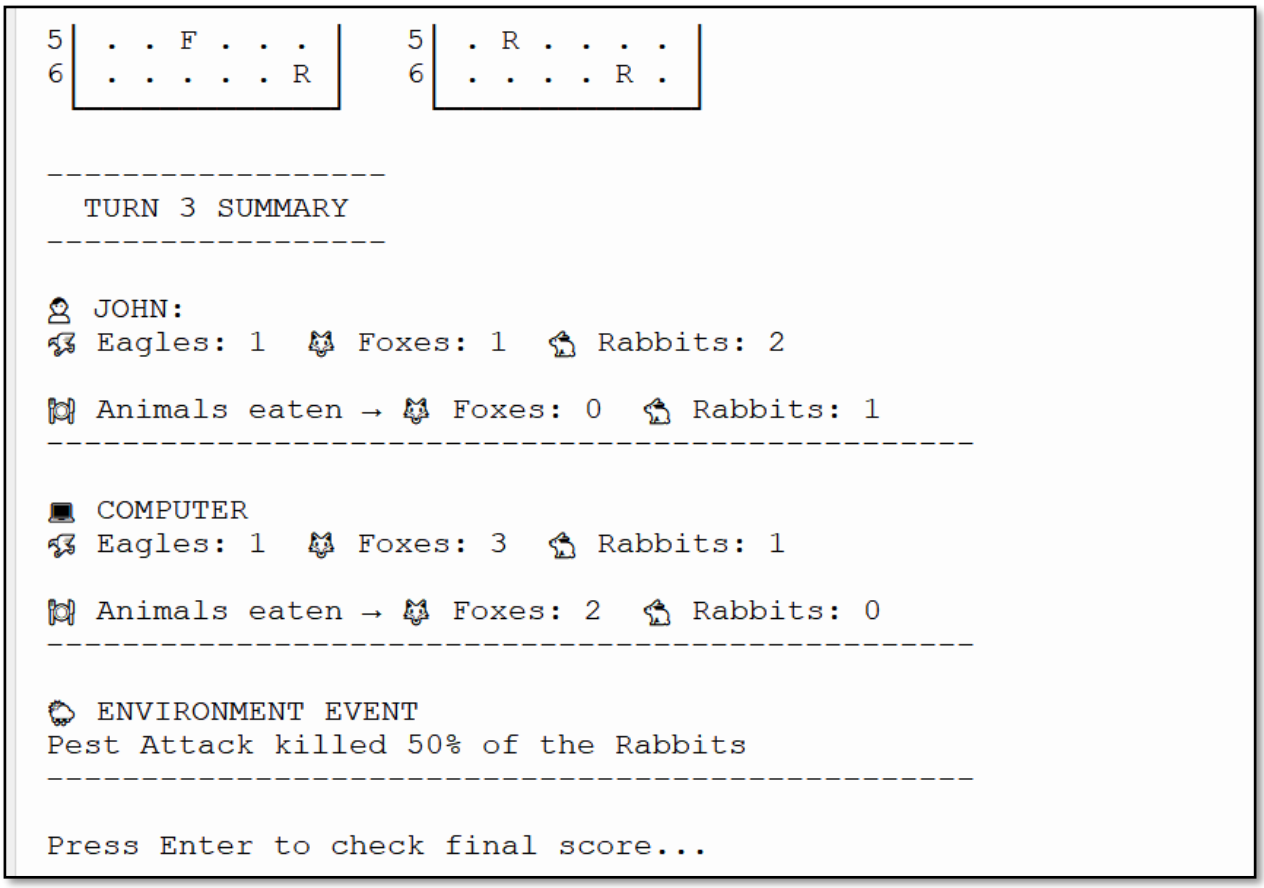
Functionality #6: Environmental Events

On the third turn, one random event may occur:

- Pest Attack (25% probability): Removes 50% of one random species for both players. *(Rounded to the nearest unit; e.g., if a player has x3 Foxes, x2 are removed)*

- Foreign Predator (10% probability): Removes one (x1) random Eagle of one random player.
- Favourable Weather (20% probability): Adds two (x2) animals of one random species to one random player.
- Otherwise, no event occurs.

💡 **A text-based sample visual is shown in Screenshot 5.**



Screenshot 5: Visual sample of turn 3 summary results

Functionality #7: Scoring System

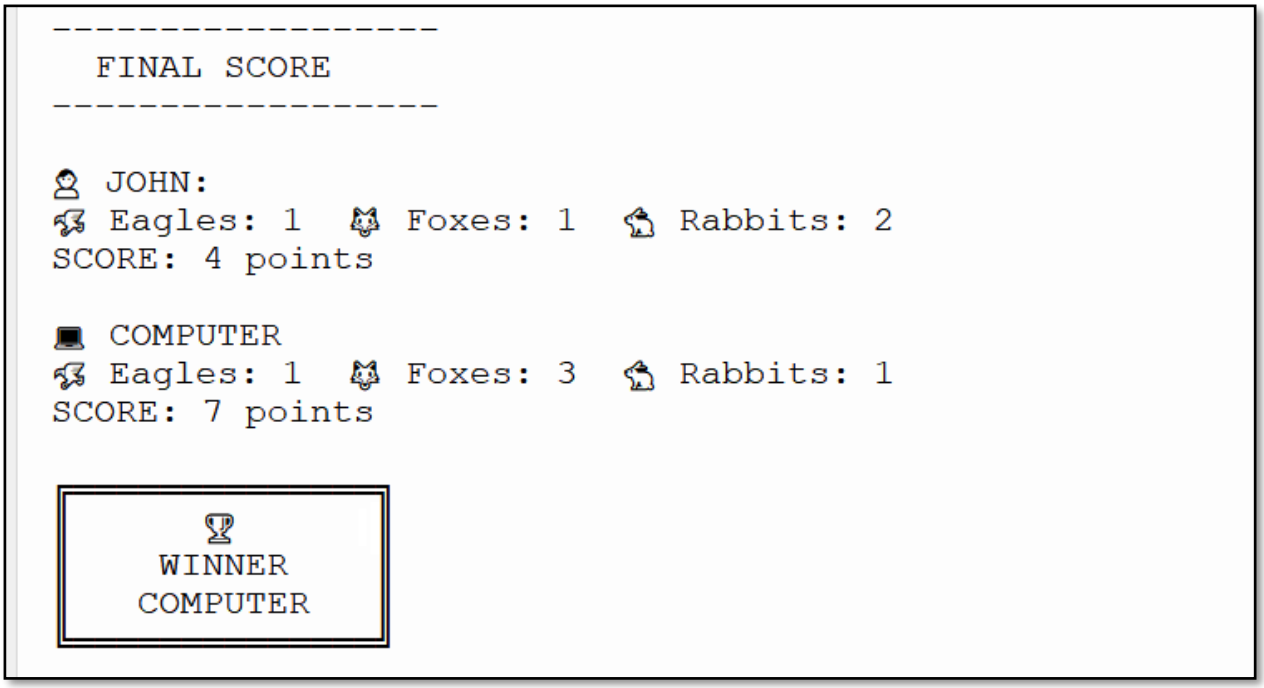
At the end of Turn 3, or upon early game termination, the final scores must be calculated and saved as follows:

- Add 2 points for each remaining fox; and
- Add 1 point for each remaining rabbit;
- Eagles equate to 0 points.

🏁 **Game Result:**

- The player with the highest points wins, OR
- If both players have the same number of points, the game ends in a tie.

💡 **A text-based sample screenshot is shown in Screenshot 6.**



Screenshot 6: Visual sample of the game ending

Functionality #8: Scoreboard System

At the end of the game, the user's score must be saved in a file called `scoreboard.txt`.

- If the file does not exist, it must be created automatically.
- If the file already exists, the new score must be added to the end of the file.
- Each scoreboard entry must include:
  - Username
  - Grid Size
  - Final Score
  - Result (Win / Loss / Tie)
- After saving the result, the program must return to the main menu.
- If the user chooses to see the scoreboard from the main menu, the scoreboard should:
  - Read all records from the file.
  - If it includes saved records:
    - Sort the records by highest score first.
    - If scores are tied, players with a Win result must appear before Tie results, followed by Loss results.
    - If records are still tied, usernames must be sorted alphabetically.
    - Display a structured, readable scoreboard on-screen.
  - If it does not include any saved records, an appropriate message should be displayed.

⚠ **Important:** The scoreboard must save previous results.

💡 **A text-based sample screenshot is shown in Screenshot 7.**

```
============================================================
                    WILD DOMINION SCOREBOARD
============================================================
 Rank   | Username          | Score       | Outcome   |Grid Size
--------+-------------------+-------------+-----------------
   1    | Alice             | 15 points   | WIN       |8x8
   2    | Bob               | 12 points   | WIN       |7x7
   3    | Zack              | 12 points   | WIN       |10x10
   4    | Charlie           | 12 points   | TIE       |9x9
   5    | Eve               | 12 points   | TIE       |6x6
   6    | Dave              | 12 points   | LOSS      |8x8
   7    | Grace             | 12 points   | LOSS      |8x8
   8    | Frank             | 10 points   | WIN       |8x8
============================================================
```

Screenshot 7: Leaderboard display sample

Functionality #9: Validation Rules

Besides the validation rules mentioned above, the program must:

- Prevent any invalid input.
- Re-prompt the user when the input is invalid.
- Avoid runtime errors.

Functionality #10: Modular Programming & Good Programming Practices

- Structure your code modularly, using any preferred programming paradigm of your choice, being it function base and/or object-oriented.
- Use any data structure/s of your choice where appropriate.
- Include comments and apply Java or Python naming conventions.

FOR PYTHON:

Name your program **main_app.py**
include all files into a folder called **postSec_wildDominion**,
*and submit as a Zip file named **postSec_name_surname**.*

FOR JAVA:

Name the class containing the main method **'RunApp'**,
include all files into a folder called **postSec_wildDominion**,
*and submit as a Zip file named **postSec_name_surname**.*

Assessment Rubric

Core Functionality (40 points)	
Program Initialisation	Displays title and shows main menu.
Gameplay Initialisation	Accepts username, asks for grid size, and creates grids for both players.
Game Sequence	Stops when turn limit is reached OR no Foxes/Rabbits remain.
Animal Population	Correctly initialises animal population as per grid size for both players (following formulas).
Animal Setup Completion (Player)	Player places all animals using valid empty cells.
Animal Setup Completion (Computer)	Computer places all animals at random using valid empty cells.
Movement Execution (User)	Applies animal movement correctly when valid (many, one or none).
Movement Execution (Computer)	Executes valid animal movement at random (many, one or none).
Predator Interactions	Correctly resolves all interactions based on species hierarchy.
Environmental Event Probability.	Correct event probability mechanism applied on turn 3 only.
Environmental Event Application	Applies correct effect according to event type.
Turn Grid Update	Correctly updates grid state after every turn.
Turn Summary Update	Correctly updates movement, interaction, and events.
Scoring Calculation	Computes final scores correctly based on remaining animals.
Result Determination	Correctly identifies winner or tie.
Save Results	Correctly saves result at the end of file. Including file creation if it does not exist.
Sort Records	Sort records as requested before displaying the scoreboard.
Display Scoreboard	Displays all scoreboard fields OR Appropriate message if scoreboard is empty.
Input Handling	Manages all the required user input.
Other Functionalities	Other possible 'hidden' functionalities work correctly.

Validation Checks (14 Points)	
Username	Enforces letters only, minimum length, and unique (not included already in scoreboard.txt)
Grid Boundary	Rejects positions outside grid.
Animal Overlap	Prevents multiple animals of same player in same cell.
Animal Placement Completion on 1st turn.	All required animals are placed before proceeding.
Movement Distance	Enforces movement of exactly one cell.
Movement Frequency	Enforces each animal moves at most once per turn.
Avoidance of Run Time Errors	Program does not crash unexpectedly during runtime.
Technical Implementation (6 Points)	
Data Structures Usage	Appropriate and effective use of lists and/or dictionaries and/or tuples and/or sets, etc.
Modular Programming	Using function-based, and/or OO programming and/or any other programming paradigm.
Code Efficiency & Readability	Code is efficient, avoids unnecessary complexity, and is easy to understand.
User Experience & Good Programming Practices (10 Points)	
Interface Clarity	Output is clear and user-friendly (text or GUI).
User Prompts	Instructions and prompts are clear and understandable.
Java / Python Conventions	Adheres to Java / Python naming conventions (e.g., camelCase or snake_case), uses meaningful variable names, and maintains consistent indentation.
Code Comments	Includes clear and concise comments to explain complex logic, functions, and important sections of the code.
File Naming	The program file is correctly named according to the specified format.
0 – Not Satisfactory | 1- Partially Satisfactory | 2- Entirely Satisfactory	
Maximum Score: 70 + 2 for every extra feature*	

* Extra features should extend the task functionality or demonstrate technical creativity WITHOUT altering or interfering with the core rules or required functionality of the task.

codesprint

ORGANISED BY

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION AND SPORT
DIRECTORATE FOR STEM AND VET PROGRAMMES

×

icecampus

POWERED BY TOP BRANDS

MAIN TITLE SPONSORS

mdia MALTA DIGITAL INNOVATION AUTHORITY

me direct

COMMUNITY & EXPERIENCE SPONSORS

pwc

melita

Bolt

KNIGHTS COLLEGE

JOli

XJENZA MALTA

SUPPORTING SPONSORS

mita

edQuanta RESEARCH FOR A BETTER TOMORROW

computime BUSINESS SYSTEMS ENGINEERING

9H

MEDIA PARTNERS

Lovin Malta

TIMES OF MALTA

meet INC.