Project “Amazons”

Game instruction

1. This is an extended version of board game "The Amazons" (https://en.wikipedia.org/wiki/Game_of_the_Amazons).
2. The game is played on chess-like board, but the size of the board is not fixed. Moreover, there can be treasure or some artifact on each field of the board. The game can be played by one, two or more players. Each player has a given number of amazons.
3. The game starts with the setting phase played in turns. In a single turn the current player places **one** of his amazons on an unoccupied field. Then, the next player does the same. This steps are repeated until all amazons are placed. Players do not get treasures of artefacts during that phase, however the treasure and artifacts are removed from the used fields.
4. Game is played in turns. In his turn the player moves one of his amazons one or more fields in a straight line horizontally, vertically or diagonally. Then, the moved amazon shoots an arrow in a straight line horizontally, vertically or diagonally at least at distance of one field. Neither amazon nor the arrow can cross or enter a field with an obstacle, that is, another amazon, landed arrow or spear.
5. When the player cannot make a valid move then his turn is skipped. Game ends when all players have no valid move.
6. After moving the amazon, the player collects the treasure and the artifact from the field where the amazon landed. The treasure can be worth from 1 to 5. The artifact must be used immediately.
7. The winner is the player who gets the most treasure.
8. The artifacts are as follows:
   a) a horse - the amazon immediately does the next move
   b) a broken arrow - the amazon does NOT shoot an arrow after moving
   c) a spear - instead of shooting an arrow, the amazon throws a spear; No obstacle blocks the throw, so the spear can literally land on any field with no amazon or arrow and the field becomes an obstacle, like in case of arrow. The spear can be moved only horizontally, vertically or diagonally.

Your task

Your aim is to deliver a program that will play the game, either autonomously or interactively, within the proposed procedure (see further in this text).

The text file format describing the state of the game

row 1 (board dimension): m n

rows 2 to m+1: n fields separated by single space, each field consists of 3 digits: first digit represents the value of treasure (0-5), second digit represents an artifact (0-none, 1-horse, 2-broken arrow, 3-spear), and third digit informs about occupation (0-tile is unoccupied, 1-8 ID of the player, 9-arrow or spear). A combination of numbers 000 represents an unoccupied field with no treasure and no artifact. In his turn the player modifies the field of departure and field of arrival of the amazon as well the field destination of arrow (spear). If the last digit is in the range 1-8, the first and the second digits should be 0.

row m+2 and consecutive: 3 fields separated by single spaces: first field is unique player name (string no longer than 15 characters, without spaces, without quotation marks), second field is

player's ID (number 1 to 8), third field shows the number of points collected so far from treasure. If there is no row with player's name (at the beginning of the game) player must add a corresponding row at the end of the file, containing the name and setting consecutive player's number as the ID and initial points set to 0. Otherwise, the player should alter his current score by the number of the collected points, and leave other fields unchanged.

Be aware that each line of the file may end with any number of spaces, try to prepare as much robust program as you can.

Your program should accept command line parameters:

- `phase=phase_mark` - `phase_mark` can take value `placement` or `movement`
- `amazons=N`, where `N` is a number of amazons that player has; this parameter is only specified if `phase=placement`;
- `inputboardfile` – name of the file that contains current game state
- `outputboardfile` – name of the file in which the new game state should be stored
- `name` – request to display player's `name` end exit the program.

`iputboardfile` and `outputboardfile` can be the same file (can have the same name)

Examples:

```
game.exe phase=placement amazons=3 inputboard.txt outpuboard.txt
game.exe phase=movement board.txt board.txt
game.exe name
```

If `phase=placement`, then the program is supposed to read the game state from the file, place one amazon on the board, save the game state to the output file and exit. In total, the program cannot place more than `N` amazons on the board (as defined by the `amazons` parameter). In such a case the program exits immediately returning appropriate error code (see later).

If `phase=movement`, then the program is supposed to read the game state, move one of his amazon and shoot an arrow, alter the score, save the game state and exit. If the move is not possible, the program should exit immediately returning appropriate error code (see later).

If parameter `name` is used, then other parameters are ignored. Program should display only the player's `name` hidden (written directly) in the code. This option enables the game master to identify players.

Interactive game

The program should also be ready to be compiled for an interactive mode (use preprocessor directives to choose the mode). In the interactive mode the program should allow to play interactive game between two players sitting in front of the computer. The game state does not have to be stored in the output file. The game state can be presented in a text mode (graphics mode is not compulsory).

Result returned by the program

Program should end in a controlled way, returning to the operating system one of the values (specified

as the argument of the `return` statement in the `main` function):

0 - program made the correct move
1 - program can not make any move (in the placement phase - all amazons have been placed; in the movement phase – all amazons are blocked)
2 - error of the game state in the input file; in such a case additional error message should be displayed identifying the cause and the position of the error in the input file
3 - internal program error, possibly with additional message

Additional issues: think how you can prevent cheating by your opponent.

Gameplay

Game is managed by game master, i.e. operating system script provided by Instructor. Script generates the board (players do not create initial board), then players' programs are run in a loop. For example, let `program1.exe, program2.exe and program3.exe` be the programs provided by students 1, 2 and 3, then the script works according to the following pseudo-code:

```
generate_board.exe
while(true)
      kod1 = program1.exe  phase=placement amazons=3 board.txt board.txt
      kod2 = program2.exe  phase=placement amazons=3 board.txt board.txt
      kod3 = program3.exe  phase=placement amazons=3 board.txt board.txt

      if (kod1 == 1 && kod2 == 1 && kod3 == 1) break;
}

while(true)
      kod1 = program1.exe  phase=movement board.txt board.txt
      kod2 = program2.exe  phase=movement board.txt board.txt
      kod3 = program3.exe  phase=movement board.txt board.txt

      if (kod1 == 1 && kod2 == 1 && kod3 == 1) break;
}
```

Programs compiled for an interactive mode should allow a human player (instead of the computer) to make a decision about next placement/move.

Teamwork

Project is made in groups consisting of 3-4 persons. Group members are selected by Instructor, you are not supposed to change the membership. Teamwork supporting tools will be utilized, including GitLab. It is group project, so consider choosing a project manager among team members to ease the workflow. However, each student is expected to be able to explain any part of his/her
code. Contribution of each student must be clear and reflected in the source files as well as clearly visible at GitLab.

Git repository

To ease the team work, the use of version control system is required. Git is one of the most popular ones. It is required that each group creates its own git repository (git project). It allows both web based and command line based access to projects. Use faculty gitlab server: https://gitlab-stud.elka.pw.edu.pl/

Assessment (max 30 points)

1. Assessment of your progress made during T1-T5 (10 points, detailed scores provided in the schedule below)
2. Final assessment and tournament during T6 (20 points)
   a) Project structuring (division of the source code into files, function breakdown) (4 points)
   b) Code quality (clarity, simplicity, comments, good naming convention, formatting, appropriate data types and data structures) (5 points)
   c) Testing (automated tests, documented manual tests) (4 points)
   d) General impression (5 points)
   e) Tournament result (2 points)

Extremely important remark: assessment is made individually (if the program is perfect, but your contribution is low, you will get very few points; if quality of the code is high, but your part is weak, you will get much less points than others; if your program is not fully functional, since one team member failed in his tasks, you still can get maximum).

To trace your activity and see your contribution we will use GIT statistics. Activities not registered at GIT or software not added to repo will be not taken into account.

Schedules (with points)

T0. Kick-off meeting, introduction to the challenge. Start working on flowcharts to sketch the general concept (start with an interactive mode and then extend it for autonomous mode).
Everybody needs to deliver individual flowchart (or flowcharts) one day before T1.

T1. Discussion of submitted flowcharts (max 2 points).
Establishing the teams, preliminary discussion in teams. Agree on communication channels in the team. Organize your work (GitLab required). Put chosen flowcharts to the repository.

T2. Present preliminary code for an interactive mode: main loop, interaction with user. Show it and run it. (max 1 point)

T3. Present the structure of the project in terms of division into files and functions. Present declarations of each function with comments describing arguments, results and purpose of the function. Create issues on GitLab and assign team members to issues. (2 points)

T4. Present data structures for board, printing board on the screen. The user should be able to set the amazons, make movement, but still it can be imperfect. (2 points)

T5. Handling the files, runtime arguments processing. Your program should read and write to file according to the specification. It should interpret properly command line arguments. (3 points)

T6. Tournament, final assessment.

Each presentation cannot be longer than 8 minutes. Everything what is to be presented and to be assessed must be pushed to GIT before the meeting (create folders with meaningful names). If you had team meeting, then make a note and attendance list and put it on GIT. Alternatively, usage of individual

group channel on Teams is recommended. Let every team member say few words about their contribution during last period. The main person to present the work you did can be chosen by the lecturer.
The aim of the presentation is to get the feedback and make sure that the project is going well. You are expected to be an active listener - remember, that you can find answers to your concerns or brilliant ideas in other students' projects and use it in your project.

TIPS & TRICKS

1. Always keep the source code ready to be compiled - use commenting for that
2. Assign the tasks to everybody in the team, especially from T3, when you can work in parallel.
3. Don’t start with loading/saving files. You may do a lot of things having some constants in your program and then you can replace them with data read from file.
4. It's better to have simple solution that will work than the most sophisticated algorithm but not working. Try to work incrementally, be agile!

Q & A (To be filled in as questions arise)

1. Q: Does the amazon shoot an arrow twice in case of "horse" artifact?
   A: Yes, shooting an arrow is a part of a move. So the amazon makes move, shoots an arrow, collects an artifact (a horse), moves and shoots an arrow once again.
2. Q: Is our program expected to do the optimal moves, so it must have a kind of artificial intelligent in it?
   A: First, you should do the program that will follow all assumptions especially regarding command line parameters and file format, but with the simplest possible intelligence. For instance you may make decisions randomly. Then you may master you decision algorithm. Try to put some reasonable rules, make some improvements. However, optimal decisions and very sophisticated algorithm is not expected. You still may do it for your satisfaction, glory and victory in the tournament, but notice, that amount of points for the winner is almost insignificant.