Project Carcassonne

Game description

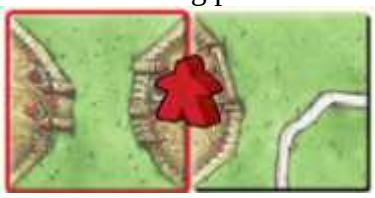
Player is given a set of squared tiles. Each tile is divided into 4 segments, i.e., north, east, south, and west. On each segment of a tile there can be one of the following symbols:
- city,
- road,
- plain.

Moreover, in the middle of a tile there can be a temple. Player may rotate the tiles. The aim is to arrange a board using the tiles to maximize the total amount of points. You don't have to use all tiles. Two tiles must have the same symbol on neighboring segments. Examples of valid tiles arrangement:

And invalid one:

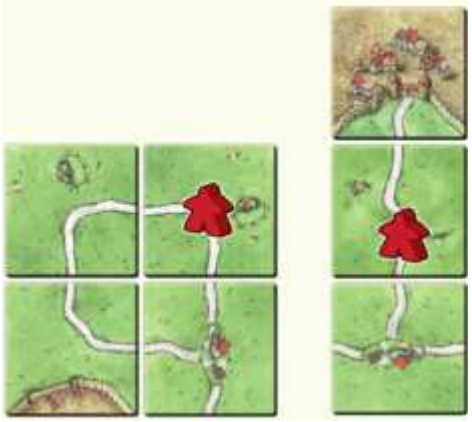
For each city segment you are given 1 point. However, if whole city is surrounded, then the city is completed, and each city segment is worth 2 points. There are also city segments with bonus point, marked with a small shield, which gives +1 points for a tile.
Here are 3 cities, no one is completed, but the city to the left gives bonus point (it has blue mark), so the cities are worth 4 points:

In the following picture:



there is one city in the middle which is completed, and one city which is incomplete. The scoring for the cities is 2+2 (completed city) + 1=5 points. Total scoring is 6 points due to the road.

For each road segment you are given 1 point. However, if a road is completed, then each road segment is worth 2 points. Road is completed if it starts and ends in: city, temple or crossroads. Here are two completed roads (and 3 road segments that are incomplete):



First pictures results in 11 points for roads, and second case counts to 8 points for roads.
For each temple you are given 1 point plus one point for each surrounding tile. So, maximum number of points for a temple is 1+8=9.

The main idea of the game is based on Carcassonne board game, however it is simplified.

Your task

Your aim is to deliver a program that will play the game on your behalf (auto mode). Your primary aim is to have program that will work, and then you can master the quality of your algorithm.
The set of tiles will be provided in a file. Program must put final board (and score) into the output file. Program must accept some command line parameters, including the names of files, for instance:

```
carassone.out tiles.txt board.txt
```

Complete set of command line parameters and format of input and output file are to be decided within the tutorial group.
In the auto mode your program can do its job in no more than 60 seconds for 30 tiles.

The program should also be ready to be compiled for an interactive mode (use preprocessor directives). In the interactive mode the program should ask a user for decisions about which tile and where to place it. After each placement (one turn), the program should write current map to the output file.

At the end of the semester we will arrange a tournament. Each program will be run with the same set of tiles and then we will validate the output file and calculate final score.

It is group project, however, each student is expected to be able to explain any part of your code. Contribution of each student must be clear and reflected in the source files. Each group is expected to deliver a report, which should be short and concise. It should include brief description of your algorithm and your code (structure of the code, main ideas) and your reflections - what you see as your success, what went wrong, what problems you have encountered.

Assessment

- quality of the code (use of proper data types, formatting, comments),
- structure of sources (division into files, functions, repetitions in code),
- fulfillment of the requirements,
- final result during tournament compared to other programs.

Expected schedule (7 meetings)

1. Introduction, preliminary discussion, kick-off. Establish your team. Use flowcharts to sketch the general concept. Design a structure of input/output files. Decide on command line parameters. Organize your work (git recommended).
2. Decision on the file formats and command line parameters (each team should prepare its suggestions). Preliminary code for interactive mode: main loop, interaction with user.
3. Structure of the project - files, main functions.
4. Data structures for board, printing board on the screen.
5. Handling the files.
6. Algorithm, final data structures.
7. Final tournament, assessment.

Other expectations for maximal points:

- sophisticated data types like structures, enum, ...
- preprocessor macros
- conditional compilation controlled by preprocessor macros
- dynamic memory allocation
- must work under windows or linux control without any additional libraries (if you really need any then contact me)

Don't try to include everything what is mentioned above at once. Remember - it's better to have simple solution that will work, then the most sophisticated algorithm but not working. Try to work incrementally, be agile! Try to always have code that can be compiled.

Tasks for bonus points:

- tiles generator
- points calculator

- validity checker

Anybody is welcome to do these additional tasks, but you are recommended to share your intentions and vision during the tutorials or via facebook.

You should send your code and report no later than 27.V (odd weeks) or 1.VI (even weeks).

Type of tiles in the game: