Project Brief 2019

Introduction

This brief provides details for the Software Development I Project. It consists of several parts. The first part describes in general what you are required to do. The second part presents a detailed specification of how data will be presented to your program and what you will be required to generate as output. The third and final part discusses the deliverables that must be submitted before the project hand-in deadline.

Important information:

- The deadline for submission of the final project deliverables is **7h50 29 April 2019**. Late submissions will be penalised as described in the Schools policy on late submissions.
- The project contributes 30% to the final mark of the course.
- Full details about final deliverables and assessment criteria are included in this project brief.
- The project is an individual effort and you are warned that cheating and plagiarism will not be tolerated - this applies to both source code and written reports. Any suspected cases of cheating or plagiarism may be referred to the legal office of the University for further investigation.
- Although you may make extensive use of the Internet in the background and research component of the project, be sure to only make use of respected online references in the final report.
- At the end of the project, you will be required to provide a comprehensive timesheet which includes a complete breakdown of both the original estimated time as well as the actual time spent. It is therefore critical to establish a disciplined approach right from the start of the project (hint: make use of an engineering notebook as well as a tool such as a spreadsheet for logging time. There are also free apps available for time logging, such as Toggl: `https://toggl.com/`).

Part 1: Background

Reversi, also known as *Othello* is a two-player board game played traditionally on an 8 x 8 board. Players take alternating turns, placing black or white pieces in empty blocks on the grid. The goal of the game is to occupy the most number of spots on the board. You are required to develop a computer program in C++ that allows **two different algorithms** to compete against each other in a **game of Reversi**.

Each reversi piece has a black side and a white side. The game always starts with an initial board setup as seen in figure 1. On your turn, you place one piece on the board with your colour facing up. You **must** place the piece so that an opponent's piece, or a row of opponent's pieces, is flanked by your pieces. All of the opponent's pieces between your pieces are then turned over to become your colour.

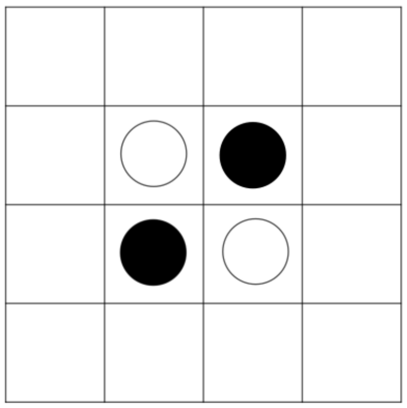

Figure 1: Starting configuration of Reversi.

You can capture vertical, horizontal, and diagonal rows of pieces and on a single turn you can capture more than one piece. If a player cannot place a piece that flanks at least one of their opponents pieces, they forfeit their turn. The objective of the game is to own more pieces than your opponent when the game is over. The game is over when neither player has a move which usually means the board is full. A player wins, by making his colour dominant on the board.

It is easier to learn a game by playing it yourself. Here is a link to an online version with a fixed board size of 8 x 8: `https://m.twoplayergames.org/play/reversi.html`.

In this project you are required to:

1. **Implement Reversi for a board size of n x n where $4 \leq n \leq 16$ and n is always even.**
2. **Implement two different algorithms that will play Reversi against each other e.g. an algorithm that randomly chooses a valid move may be one of your algorithms.**

This project will expose you to many key aspects of engineering and software development. The following are two of the most important aspects:

- The (software) engineering lifecycle: problem identification, requirements analysis, design, implementation, testing and verification, documentation and maintenance.
- Project planning and time management.

Take time to research and find out more about the above.

1.1 Project and Time Management

Project and time management are essential skills that must be consistently developed and refined, especially in software development. As the saying goes time costs money - failure to accurately predict in advance the time spent on a project can be very costly. There are many examples of large software project disasters where deadlines were not met.

It is therefore essential that you develop good habits as early as possible, for example, always make use of an engineers notebook and maintain a comprehensive spreadsheet that contains a breakdown of the time you spend on various activities within a project. Get into the habit of estimating the amount of time you think a task will take and then retrospectively comparing the prediction against the actual time spent. Critically analyse these times, even if the actual time was exactly what you predicted it to be. With practice and experience you will develop an important skill.

As a starting point, you may wish to use the following broad software development activities and the proposed time breakdown:

Activity	Time Breakdown
Background (problem understanding, requirements)	15%
Analysis & Design	20%
Implementation (coding)	20%
Testing	20%
Documentation	25%

Another important piece of information to consider is that each second year student taking Software Development I is expected to spend 12.5 hours every week on the course (including lectures, laboratories and tutorials).

Given the above information, as well as the deadline for the submission of project deliverables, you should be able to formulate a suitable time plan for your project. You may use the breakdown introduced in the previous section, or propose your own. For each major activity, list the time that will be spent on the activity as a percentage of the total time, as well as the actual number of hours. You must also provide the total number of hours you plan to spend on the project.

Part 2: Data and Specifications

You are required to develop a computer program in C++ that allows two different algorithms to compete against each other in a game of Reversi.
Your code must meet the following specifications:

- Read in a list of numbers specifying board sizes from a file.
- Every game must begin in the starting configuration as seen in figure 1.
- Algorithm 1 will take alternating turns with algorithm 2 at placing their markers in the grid.
- A marker cannot overwrite an already filled position in the grid.
- No global variables may be used - this will result in failing the project with FCOM.
- The list of moves made by each algorithm must be stored to an output file.

You MUST ensure that you follow ALL of the instructions, and meet all the specifications and requirements, provided in this document. Failure to do so will result in certain assessment outcomes being rated *Poor* or *Unacceptable.*

2.1 Input format

The input file named `input.txt` will contain a list of board sizes, seperated by commas, that should be used to test the two algorithms against each other. The input file will have the following format

```
4, 4
```

Note that the input file may have *any number of board sizes.* You can assume that this file will contain board sizes in the range of 4 to 16, always being even. Tip: test your implementation of the game and algorithms with an input of 4 first as a 4 x 4 board is easier to debug than a 16 x 16 board.

2.2 Output format

Your program must produce one output file. The file must be named `results.txt`. This file must contain the following:

- size of the board for the current game,
- each position filled, which algorithm filled it (in the order of placement) and the locations where pieces flipped in that turn,
- the number of blocks occupied by each algorithm and
- the winning algorithm.

The history for each board size listed in the input file must be appended to this file with one empty line inbetween different games.

Note that coordinates on the board are labelled using the value of the row (`r`) and column (`c`) where the top left position is `r0c0` and bottom right is `r3c3` for a board of size 4 by 4. An example of `results.txt` is given below:

Listing 1: An example of the format for `results.txt`.

```
size = 4
r0c1 alg1, r1c1
r0c0 alg2, r1c1
r1c0 alg1, r1c1
r2c0 alg2, r1c0 r2c1
r3c1 alg1, r2c2
r2c3 alg2, r2c2
r3c2 alg1, r2c2
r0c3 alg2, r1c2
r1c3 alg1, r1c2 r2c3
r0c2 alg2, r0c1 r1c1
r3c0 alg1, r2c1
r3c1 alg2, r2c1
alg1 = 7
alg2 = 9
win = alg2

size = 4
r2c3 alg1, r2c2
r3c1 alg2, r2c1
r3c0 alg1, r2c1
r1c3 alg2, r1c2
r0c3 alg1, r1c2
r3c3 alg2, r2c3 r2c2
r0c1 alg1, r1c1
r1c0 alg2, r1c1 r1c2
r3c2 alg1, r2c1
r2c0 alg2, r2c1
r0c0 alg1, r1c0 r2c0
r0c2 alg1, r1c2 r2c2 r1c1
alg1 = 12
alg2 = 4
win = alg1
```

The first games play is shown in figure 2. Note that the positions that appear after the comma on each line have a bold block around them. This shows which positions flipped colour in that turn.

Note you must use the exact format above - it is case sensitive, spacing sensitive and the names `alg1` and `alg2` must be used. In your report you can elaborate on what each algorithm is. There should be no console output produced in your final submission.

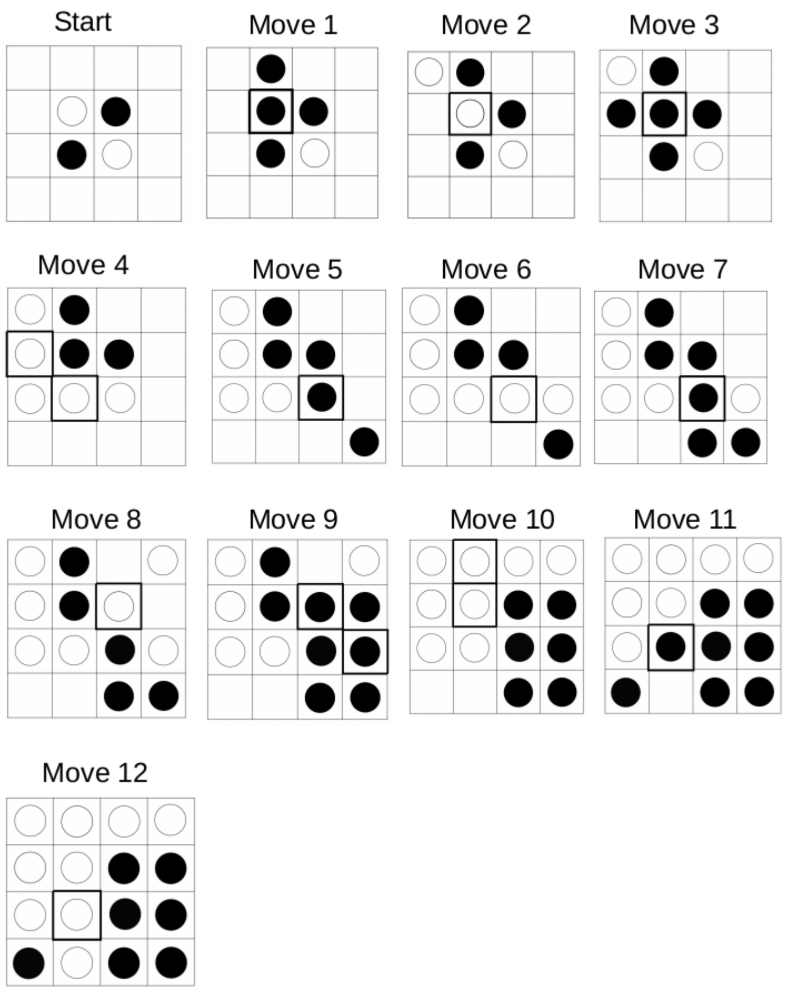

Figure 2: Step by step game play of the first Reversi game in `output.txt`. The bold outlined blocks signify a piece that has flipped colour because of a placement of a piece in this move.

Part 3: Deliverables and Submission

You are required to submit the following deliverables before the submission deadline:

- Electronic submission of project source code (`.cpp` and `.h` files)
- Electronic submission of PDF version of project report

Links will be provided on Sakai for submission of your source code and PDF report online. **Note that there is a separate link for your PDF and a separate link for your code. You will be penalised if you do not submit correctly.** If you have made use of any external software libraries or have used multiple source code files, be sure to include these in your online submission.

NOTE: failure to submit either of the above deliverables before the deadline will result in the overall project result being subject to a penalty as described in the Schools policy on late submissions (refer to the Red Book). Any exceptions must be discussed by the student with the Head of the School of Electrical & Information Engineering.

3.1 C++ source code

Your source code will be assessed by looking for evidence supporting the successful engineering execution of the project. This means that coding style, use of comments, clarity of code, ability

to compile, run and generate correct output and adherence to specifications and requirements are important factors.
The first line of your C++ source code must be a single line comment with the format (replace with your own student number):

```
// <StudentNumber>
```

Test data will be available on the course website. Note, however, that the input data file your program makes use of must be named `input.txt`.

Your code will be marked on a machine running g++ version 8.2. Ensure your code includes any necessary headers correctly. You can check this by compiling your code via the command line (on Windows, Ubuntu or Mac OS). Open the command prompt window from within your source code directory (or navigate there) and type `g++` followed by a list of all source code files you have created. For example you may have:

```
g++ main.cpp class.h -o run
```

where you replace `main.cpp class.h` with the names of your source code files. You should not see any errors. You can then run your code from the command prompt by typing

```
./run
```

When writing source code, make sure to practise good programming techniques as discussed during the course. In particular, pay attention to variable naming, comment your code, be consistent with indentation, avoid global variables at all costs, avoid placing too much code in `main()`, include thorough error checking and validation, and make use of well thought out modules and/or classes to solve the problem effectively.

Remember that careful planning and time management is critical to a successful project - at all stages keep ongoing notes about your activities and time spent for your time management record.

3.2 Documentation requirements

An important deliverable of the course project is a written report which provides tangible evidence that you have engineered a successful solution to the problem you were given. The body of this report along with your source code needs to provide evidence towards each of the following outcomes:

- Background and problem understanding
- Quality of engineering output
- Critical analysis and evaluation

Note, however, that this does not mean that your report should have the above items as section headings! The body of your report, excluding title page and appendices, must be no longer than five pages. Note that appendices must be self-standing documents.

The report must be submitted as a single electronic Portable Document Format (PDF) document. Microsoft Word has a built in export to PDF function that can be used when saving your document.

The completed documentation must be concise and of a high standard, in accordance with the accepted guidelines for report writing and presentation delivering that apply within the School of Electrical and Information Engineering (see the blue book).

Your report must meet the following specific requirements:

- It must be submitted electronically as a single PDF document via the Wits Sakai system. You are not required to submit a printed copy. Submit your report to the project report submission link.
- The report must consist of the following sections:
  - A title page, showing the name of the University and School, the course name (Software Development I), your name and student number, the branch of Engineering you are registered in, the Title: for example A simple strategy for Reversi, and the date.
  - A short abstract (50 words approximately)
  - Section 1: Introduction (see blue book)
  - Section 2: Design
    This section details the planning of your solution. Ideally this section would have been completed before a line of code was written. Detail any assumptions made prior to implementation and motivate them. Provide an overview of the major modules involved in your solution and provide a detailed flow chart which illustrates the operation of your program (2 pages or less). The flow chart must illustrate file handling and all algorithms. The focus of this section should be descriptive rather than technical. Note that the objective of this section is for someone to be able to follow what the program does and why it does it (the how is taken care of by the source code).
  - Section 3: Results
    Each student is required to test their program using data which can be downloaded from the website. You must run your program multiple times using different data sets. Using the output files created, as well as other tools (such as a spreadsheet), you should generate various tables and graphs that can be presented in your report. These results can be used to motivate which algorithm performs better (1 page or less).
  - Section 4: Discussion
    Critically analyse and discuss your approach, solution and findings. For example, what conclusions can be made about the two implemented algorithms? Were your initial assumptions valid? What improvements can be made to your approach or solution?
  - Section 5: Conclusions
    As per the Blue Book.
  - Section 6: References
  - Appendix A: Project Time Management
    Predicted versus actual with a brief critical discussion (maximum 2 pages).
- The body of your report (from the Introduction to the end of the References) may not exceed 5 pages additional pages will not be considered.
- You may include your flowchart as an appendix.

Final notes

- There is no provision for user input to your program, it must act on and produce files as specified in Part 2 and 3 without any human interaction!
- If you include multiple files in your submission please ensure you include a README file (find out what this is).
- Students aiming for good results will want to satisfy all the requirements mentioned in the previous sections, as well as tackling additional issues.
- Do not include any input or output files in your submission!

NB: Your output files must be named as specified in Part 3. Pay attention to issues related to case-sensitivity. Your report must be submitted as a single PDF document. Your source code must be submitted as separate files. Important: Check and double-check that your project submission is successful on the online system.