# Parallel and SIMD Programming – Assignment #3 (Parallel Othello AI) Due on 09/10/2008

Using the base code provided, you will implement a parallel minimax game tree search. The sample code implements a simple, single-threaded minimax game tree search with alpha-beta pruning. It also contains a simple game control logic. The top-level code should not need significant modification. Instead, create a new minimax search routine called `get_move_parallel` using the existing `get_move` as a model.

Place all new code in a new file or files. **Do not modify any code in `minimax.cpp`!** Do not modify or use a different heuristic than the one implemented in `evaluate_board`. This assignment is a parallel programming assignment *not* an AI assignment.

The only modification necessary to the top-level game code should be to use `get_move_parallel` instead of `get_move`.

As a text file separate from your source code, document your design decisions. Be sure to include your program decomposition (you may focus on either the task or data decomposition), dependency analysis, algorithm structures used, and supporting structures used. In the dependency analysis, be sure to include how tasks are grouped and any ordering and / or data dependencies.

| Criteria | Excellent | Good | Satisfactory | Unacceptable |
|---|---|---|---|---|
| Completion | Program correctly implements all required elements in a manner that is readily apparent when the program is executed. User interface is complete and responsive to input. Program documents user interface functionality. | Program implements all required elements, but some elements may not function correctly. User interface is complete and responsive to input. | Program implements most required elements. Some of the implemented elements may not function correctly. User interface is complete and responsive to input. | Many required elements are missing. User interface is incomplete or is not responsive to input. |
| Correctness | Program executes without errors. Program handles all special cases. Program contains error checking code. | Program executes without errors. Program handles most special cases. | Program executes without errors. Program handles some special cases. | Program does not execute due to errors. Little or no error checking code included. |
| Efficiency | Program uses solution that is easy to understand and maintain. Programmer has analysed many alternate solutions and has chosen the most efficient. Programmer has included the reasons for the solution chosen. | Program uses an efficient and easy to follow solution (i.e., no confusing tricks). Programmer has considered alternate solution and has chosen the most efficient. | Program uses a logical solution that is easy to follow, but it is not the most efficient. Programmer has considered alternate solutions. | Program uses a difficult and inefficient solution. Programmer has not considered alternate solutions. |
| Presentation & Organization | Program code is formatted in a consistent manner. Variables, functions, and data structures are named in a logical, consistent manner. Use of white space improves code readability. | Program code is formatted in mostly consistent with occasional inconsistencies. Variables, functions, and data structures are named in a logical, mostly consistent manner. Use of white space neither helps or hurts code reability. | Program code is formatted with multiple styles. Variables, functions, and data structures are named in a logical but inconsistent manner. Use of white space neither helps or hurts code reability. | Program code is formatted in an inconsistent manner. Variables, functions, and data structures are poorly named. Use of white space hurts code reability. |
| Documentation | Code clearly and effectively documented including descriptions of all global variables and all non-obvious local variables. The specific purpose of each data type is noted. The specific purpose of each function is noted, as are the input requirements and output results. | Code documented including descriptions of most global variables and most non-obvious local variables. The specific purpose of each data type is noted. The specific purpose of each function is noted, as are the input requirements and output results. | Code documented including descriptions of the most important global variables and the most important local variables. The specific purpose of each data type is noted. The specific purpose of each function is noted. | No useful documentation exists. |

This rubric is based loosely on the "Rubric for the Assessment of Computer Programming" used by Queens University (http://educ.queensu.ca/ compsci/assessment/Bauman.html).