Práctica 2: Algoritmos Minimax y alpha-beta. Heurísticas para juegos.

Apéndice: instrucciones especiales para C++

Se definen algunos tipos y se proporciona una clase `Board` para facilitar la implementación.
Los tipos son:

- Tipo Square: `typedef pair<char,char>Square`. Es un par de dos caracteres que representa una casilla del tablero. Para ser válido el primer carácter debe ser '1',...,'8' (la fila), y el segundo, 'A',...,'H' (la columna). Nótese que los números están represntados como caracteres, no como enteros.
- Tipo Sqlist: `typedef list<Square>Sqlist`. Una lista de casillas.
- Tipo Token: definido por enumeracion como `enum Token {x,o,e }`. La x representa negro, la o blanco y la e vacío (empty).
- Tipo Player: definido por enumeración como `enum Player {Bob,Alice}`.
- Tipo Move: `typedef pair<Player,Square>Move`. Representa un movimiento, donde la primera componente es el jugador que va a realizarlo y la segunda, la casilla donde pondrá la ficha de su color.

Las funciones auxiliares son:

- `string Name(Player pl)`: Toma un jugador (como tipo Player) y devuelve una string con su nombre. (Pensada para mostrar por pantalla).
- `Player Opponent(Player pl)`: Toma un jugador como tipo Player y devuelve su oponente, también como tipo Player.
- `Token Tk(Player pl)`: Toma un jugador como tipo Player y devuelve el valor (como tipo Token) de la ficha con las que juega.
- `char Symbol(Token tk)`: Toma una ficha (como tipo Token) y devuelve un carácter. (Pensada para imprimir por pabntalla).
- `bool ValidSquare(Square sq)`: Toma una casilla y devuelve true si es válida y false si no. Véase arriba las restricciones a los caracteres del par para el tipo Square.
- `bool ValidPlayer(Player pl)`: Toma un jugador y devuelve true si es uno de los dos jugadores válidos y false si no.

La clase se llama `Board`, y su interface público es:

- `Board()`: Crea un tablero en el estado inicial (es decir, con dos fichas 'o' en ('4','C') y ('5','E') y dos fichas 'x' en ('5','C') y (4,'E')).
- `void Show(ostream &outst)`: Muestra por pantalla (pasando cout ó cerr) o en cualquier otra stream de salida el estado actual del tablero.
- `Token Content(Square sq)`: Devuelve el token (x,o,e) que ocupa la casilla que se le pasa.
- `bool IsLegal(Move mv)`: Devuelve `true` si el movimiento pasado puede ejecutarse en el estado actual del tablero y `false` si no.
- `void MakeMove(Move mv)`: Efectúa el movimiento que se le pasa, si es legal (no es necesario comprobar de antemano, esta funcion lo hace), lo cual altera el estado interno del tablero (del objeto que llama a esta función).
- `Board TryMove(Move mv,bool &legal)`: Si el movimiento que se le pasa es legal (lo comprueba internamente), copia el tablero actual en uno nuevo, ejecuta el movimiento sobre éste, y devuelve dicha copia, con lo que el objeto que llama a esta función no se altera. La variable `legal`, pasada por referencia, vuelve con el valor obvio según el movimiento pueda ejecutarse o no.

- `bool GameOver(Player &winner,unsigned &winnerpoints,unsigned &looserpoints)`: Devuelve `true` si el juego ha terminado, es decir, si ninguno de los contendientes puede realizar más jugadas (por bloqueo, o porque el tablero está lleno) y `false` en caso contrario. Además, si el juego ha terminado, las variables pasadas por referencia vuelven llenas con respectivamente el ganador (como tipo Player), el número de fichas del ganador y el número del perdedor (como tipo `unsigned`).
- `Sqlist ValidMoves(Player pl)`: devuelve una lista de casillas en las que al jugador que se le pasa se le permite poner una ficha (por supuesto, de su color)

La función heurística está definida en los archivos `heuristic.h/cpp`. Se entrega la heurística no informativa (devuelve siempre 0) y el ejercicio es hacer otra u otras mejores, usando las funciones que se considere necesario de la clase `Board` (esencialmente, `Content` para ver qué fichas hay en el tablero y dónde están).

Además, el programa principal juega con un algoritmo minimax, definido en `Minimax.h/cpp`. La implementación está detalladamente comentada. El ejercicio es copiar estos dos archivos en otros, `Alfabeta.h/cpp` y modificar todo lo necesario para escribir un alfa/beta, probándolo luego en el programa principal en sustitución del minimax.