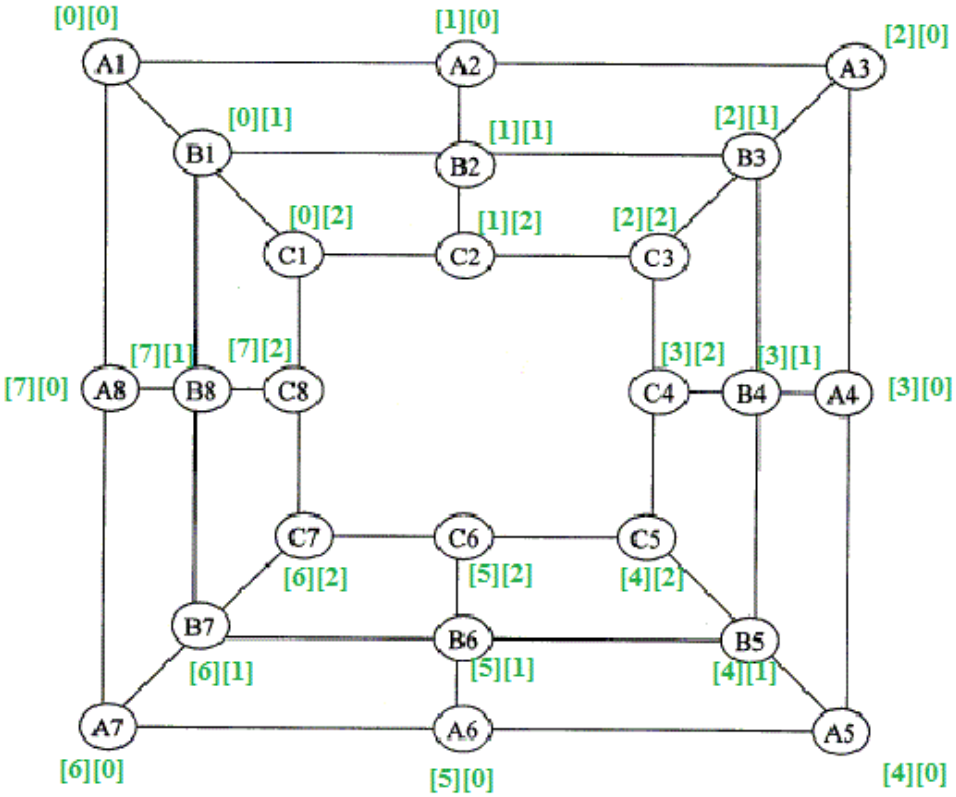
همانطور که گفته شد ، شما باید استراتژی های خود را به یکی از زبان های C++,JAVA,Python پیاده سازی کنید.

بازی از سه مربع (خارجی ، وسط ، داخل) تشکیل شده که آنها را با A، B،C نشان میدهیم و نقاط روی این مربع ها را در جهت ساعتگرد شماره گذاری میکنیم :



بنابراین صفحه بازی ، یک ماتریس ۳*۸ است.

با توجه به شرایط بازی ، یکی از کار های قرار دادن مهره خود در صفحه (put) ، حرکت دادن مهره خود در صفحه (move) و حذف مهره حریف (pop) انجام میشود . یعنی شما نیازی ندارید بررسی کنید که در هر سیکل باید کدام یک از اعمال put move pop را انجام دهید. صدا زدن توابع مربوطه به صورت خودکار توسط کلاینت انجام میشود . شما فقط نیاز دارید که استراتژی های خود را در فایل های مربوطه پیاده سازی کنید.

شما برای پیاده سازی استراتژی خود باید:

استراتژی قرار دادن مهره‌ها (put) را در فایل put_strategy بنویسید و سپس برای دستور دادن به سرور برای قرار دادن مهره جدید در یک مکان مشخص تابع put (Pos P) را از کلاس Game صدا زده و یک مکان P را به عنوان ورودی به آن بدهید تا سرور مهره جدید شما را در مختصات [P.x][P.y] قرار دهد. (مختصات متناظر با شکل بالا میباشد)

استراتژی حرکت دادن مهره‌ها (move) را در فایل move_strategy پیاده سازی کنید و سپس برای دستور دادن به سرور برای حرکت دادن مهره مورد نظر به مکان جدید ، تابع move(Checker c,Pos newpos) را از کلاس Game صدا زده و یک مهره و یک مکان به عنوان ورودی به آن بدهید تا سرور مهره موردنظر شما را به مکان newpos ببرد

استراتژی حذف مهره حریف (pop) را در فایل pop_strategy پیاده سازی کنید و سپس برای دستور دادن به سرور برای حذف مهره موردنظر از حریف، تابع pop (Checker c) را از کلاس Game صدا زده و یک مهره حریف را به عنوان ورودی به آن بدهید تا سرور، مهره موردنظر شما را از بین مهره های حریف حذف کند.

توجه کنید که حرکت مهره ها فقط بین خانه های همسایه است . برای مثال دستور حرکت از خانه[1][3] به خانه [2][3] صحیح و دستور حرکت از خانه [1][3] به خانه [2][2] نامعتبر است چون مستقیما به یکدیگر وصل نشده اند.

توابع کلاس Pos

تابع	توضیحات
Int getx()	برگرداندن مقدار x
Int gety()	برگرداندن مقدار y

توابع کلاس checker

تابع	توضیحات
Boolean isMyChecker()	تشخیص مالکیت مهره موردنظر (چنانچه مهره حریف باشد false و چنانچه مهره خودمان باشد true برمیگردد)
Pos get_pos()	برگرداندن مختصات آن خانه

توابع کلاس Cell

تابع	توضیحات
Checker get_checker()	دسترسی به مهره درون خانه (چنانچه هیچ مهره ای در آن خانه نباشد ، null برمیگرداند)
Pos get_pos()	برگرداندن مختصات آن خانه

توابع کلاس Board

تابع	توضیحات
Cell [][] get_cells()	آرایه دوبعدی ۸*۸ ، شامل تمامی نقاط صفحه را برمیگرداند
Cell[] get_myCells()	آرایه (وکتور) از سلول هایی که مهره های خودی هستند را برمیگرداند
Cell[] get_oppCells()	آرایه (وکتور) از سلول هایی که مهره های دشمن هستند را برمیگرداند
Cell[] get_emptyCells()	آرایه (وکتور) از سلول هایی که خالی هستند را برمیگرداند
Cell get_cell (Pos p) Cell get_cell (int x,int y)	سلولی را که در موقعیت [p.x][p.y] قرار دارد را برمیگرداند

توابع کلاس Game

تابع	توضیحات
Board get_board()	برگرداندن board (صفحه بازی)
void put (Pos p)	قرار دادن مهره جدید در مکان [p.x][p.y]
void pop (Checker c)	حذف مهره مورد نظر c از حریف
void move (Checker c,Pos p)	حرکت دادن مهره c به مکان جدید [p.x][p.y]
Int get_cycle()	برگرداندن شماره سیکل فعلی

توجه : در زبان C++ تمامی Cell و Pos و Checker بصورت پوینتر میباشند.

در هر قسمت از بازی ، چنانچه کد شما دستور نامعتبری بدهد ، سرور آن دستور را رد کرده و سپس یک حرکت دلخواه انجام می‌دهد. برای مثال:

- کد شما در هنگام قرار دادن مهره ها ، ادرس خانه‌ای را میدهد که قبلا توسط دشمن شما اشغال شده وشما مجاز به قرار دادن مهره در آن خانه نیستید. در اینصورت سرور ادرس خانه را نامعتبر دانسته و مهره شما را به دلخواه در خانه دیگری قرار میدهد
- کد شما در هنگام جابجایی مهره ها ، دستور میدهد که یکی از مهره ها، از خانه [2][1] به خانه [2][3] برود . این دستور نامعتبر است چون این دو خانه همسایه نیستند و هیچ مهره ای نمیتواند مستقیما بین این دو خانه حرکت کند. پس سرور مهره ای دلخواه را انتخاب کرده و آن را به یکی از خانه های همسایه انتقال میدهد
- کد شما در هنگام حذف مهره حریف ، ادرس خانه ای را میدهد که هیچ یک از مهره های حریف در آن خانه نیست. در اینصورت سرور دستور شما را رد کرده و مهره ای از حریف را به دلخواه انتخاب میکند و آن را حذف میکند.