**نام و نام خانوادگی : منا رداد**

**توضیح خلاصه نحوه عملکرد کد ها**

**Main:**

ابتدا object مهره های سفید و سیاه را ایجاد می کنیم سپس mode های بازی را انتخاب میکنیم حال اگر mode انتخاب شده مجاز نباشد پیغام خطا را چاپ می کنیم و در نهایت بازی را آغاز می کنیم.

**Player:**

در این کلاس ابتدا رنگ و نام مهره ها را تعیین میکنیم و یک متد move داریم که بدنه آن خالی است و صرفا با کلمه کلیدی pass از خطا جلوگیری کرده و در واقع هیچ کاری انجام نمی دهیم.

**Chess game:**

در این کلاس ابتدا در تابع –init— تعیین میکنیم که مهره های سفید را با w و مهره های مشکی را با b نمایش دهیم و برد اولیه بازی را برابر initial state قرار می دهیم.

سپس در تابع –str—تمامی خانه های صفحه بازی را بررسی می کنیم اگر در خانه ای مهره ای قرار داشت کاراکتر آن مهره را قرار میدهیم و اگر آن خالی بود آن را با "." نمایش می دهیم. این کار را هم برای خانه های سفید صفحه شطرنج و هم برای خانه های مشکی آن انجام می دهیم.اگر متغیر count ما برابر 8 شد به دلیل اینکه صفحه بازی 8 در 8 می باشد پس به خط بعدی می رویم.

در متد play ما حرکات انجام شده دو بازیکن را در move می ریزیم و به نوعی آن را وارد صفحه بازی می کنیم. حال اگر بازی پایان یافته باشد مقدار board.outcom را بر میگردانیم.

**Ai:**

در ابتدا ارزش هر یک از انواع مهره ها را تعیین می کنیم. سپس ارزش خانه های صفحه بازی را با توجه به نوع مهره ها و محل قرار گیری آن ها تعیین می کنیم.لازم به ذکر است که خانه های سیاه ،عکس خانه های سفید می باشند. در متد evaluate\_position نیز برای هرنوع مهره (هم سیاه هم سفید) پوزیشن آن را مشخص می کنیم توجه کنید که برای مهره شاه باید شرط end\_game را هم بررسی کنیم.

حال در متد are\_we\_in\_end\_game بررسی می کنیم که اگر مهره موجود در خانه های جدول وزیر بود تعداد وزیر ها را افزایش میدهیم و اگر مهره موجود در خانه های شطرنج فیل و یا اسب بود تعداد فیل ها و اسب ها را نیز افزایش می دهیم. حال اگر هیچ وزیری وجود نداشت و یا هیچ مهره دیگری در بازی نبود یا اگر تعداد وزیر ها برابر 2 بود و تعداد یک فیل یا اسب در بازی بود و یا اصلا این مهره ها وجود نداشتند مقدار true را برمیگردانیم.

تابع maximizer : ابتدا شرط توقف بازی( مات شدن) را چک میکند اما اگر بازی هنوز تمام نشده باشد به ازای هر حرکت مجاز که میتواند انجام دهد تابع minimizer را صدا میزند و بعد بین مقادیری که تابع minimizer بر میگرداند max میگیرد.

تابع minimizer : ابتدا شرط توقف بازی( مات شدن) را چک میکند اما اگر بازی هنوز تمام نشده باشد به ازای هر حرکت مجاز که میتواند انجام دهد تابع maximizer را صدا میزند و بعد بین مقادیری که تابع maximizer بر میگرداند min میگیرد.

برای هر حرکت این حرکات را انجام می دهد تا به برگ برسد به برگ که رسید باز برمیگردد به اول حلقه برای فرزند بعدی(حرکت بعدی) اینکار را انجام می دهد و همه فرزندان که چک شد نتیجه را به پدر برمیگرداند و باز همین روند برای فرزندان پدر یعنی همه حرکات بازی به صورت اول عمق انجام می شود.

**MiniMaxPlayer:**

در متد evaluate که در واقع هیوریستیک الگوریتم مینی مکس ما میباشد ، ما شرط پایان بازی رو برابر متد are\_we\_in\_end\_game قرار می دهیم سپس هریک از خانه های صفحه شطرنج را بررسی می کنیم که اگر دارای مهره بود اعمال لازم را انجام دهد در غیر این صورت اگر مهره ای در خانه نبود برود خانه بعدی.سپس برای هر مهره ای که در خانه های شطرنج وجود دارد، ارزش هر مهره رو با توجه به نوع آن و پوزیشن قرار گرفته شده آن مهره، در value میریزیم. حال اگر مهره ما سفید بود که با توجه به حرکت انجام شده مقدارvalue را به total اضافه می کنیم و امتیاز را افزایش می دهیم در غیر این صورت اگر مهره مشکی بود به همان میزان از مقدار value را از total کم می کنیم.

در متد move ابتدا حرکات مجاز رو در داخل لیستی قرار میدهد، سپس برای هر حرکت در لیست حرکات مجاز،ابتدا حرکات را در board ، push می کند سپس چک میکند اگر مات میشد آن را از board ، pop میکند. سپس چک میکند اگر حرکتی که بازیکن سفید انجام میدهد حرکت بهتری هست آن را جایگزین حرکت قبلی کرده و آن را درون best\_move می ریزد. و همین فرض را برای بازیکن مشکی نیز در نظر می گیرد

.متد can\_claim\_draw برای قوانین threefold , fivefold میباشد که حالت تساوی را چک میکند.حال اگر حالت تساوی اتفاق نیوفتاد توابع minimizer را برای بازیکن سفید و maximizer را برای بازیکن مشکی فراخوانی میکند و در نهایت به تعداد حرکات می افزاید.