

بسمه تعالی



مستند مسابقه

یازدهمین نبرد هوش مصنوعی شریف

مرحله ی غیر حضوری

بهمن و اسفند ۱۳۹۷

آخرین نبرد

در زمان‌های بسیار قدیم، آشوب نیز همانند آب، خاک و هوا عنصری از عناصر طبیعت بود. در آن زمان موجوداتی ساکن زمین بودند که با استفاده از قدرت این آشوب می‌توانستند کارهای خارق‌العاده‌ای انجام دهند؛ برخی از این موجودات خیر بودند و از سایر موجودات زنده محافظت می‌کردند، برخی دیگر نیز طرف شر را گرفته و هدفی جز تخریب و نابودی سایر موجودات نداشتند.

امروزه در افسانه‌ها از این آشوب، با کلماتی مانند جادو و سحر نام برده می‌شود. همچنین گروه خیر در افسانه‌ها به نام‌های قهرمانان و اسطوره‌ها ثبت شدند و گروه شر نام‌هایی چون دیوها و شیاطین را گرفتند. مادامی که جادو در زمین فراوان بود، مبارزه بین این دو گروه خیر و شر همواره ادامه داشت و هیچ یک از دو گروه نمی‌توانست گروه دیگر را به طور کامل شکست دهد و کنترل زمین را به عهده بگیرد.



اما با گذشت زمان، شرایط زمین رو به تعادل می‌رفت و جادو که ماهیت آن همان آشوب بود، به مرور در زمین کمرنگ شد و با سایر عناصر طبیعت ترکیب شد. با تضعیف جادو در زمین، موجودات جادویی نیز کم کم قدرت خود را از دست دادند و یکی یکی در مکان‌های مختلف به خواب فرو رفتند. زندگی روی زمین بدون جادو و موجودات جادویی ادامه داشت و رفته‌رفته انسان‌ها پیشرفت کرده و با رشد تکنولوژی و علم، زمین را در اختیار خود درآوردند. اما این رشد بدون توجه به طبیعت باعث شد که انسان‌ها صدمات بسیاری به طبیعت وارد آورند و تعادل موجود در آن را به هم بزنند. فشارهای انسان‌ها به طبیعت ادامه داشت تا

زمانی که این تعادل به هم خورد و آشوب یا جادو که در تمام این مدت به بخشی از طبیعت تبدیل شده بود، همانند آتش‌فشانی در نقطه‌ای از زمین فوران کرد.

با ظهور دوباره‌ی جادو در زمین، موجودات افسانه‌ای که به خواب رفته بودند، بار دیگر برخاستند تا این نیروی عظیم را تصاحب کنند. در تمام این مدت، جادو که به بخشی از طبیعت تبدیل شده بود، به شکلی دست‌نخورده باقی مانده بود؛ و به همین علت حجم نیروی فورانی به حدی بود که هم خیر و هم شر می‌توانستند با استفاده از آن نیروی مقابل را به طور کامل نابود کنند و به این جنگ طولانی پایان ببخشند. قهرمانان هر دو طرف به سرعت به مکان فوران جادو رفتند تا در آنجا مبارزه‌ای نهایی انجام دهند، بر سر نیرویی که می‌توانست این جنگ طولانی را به اتمام برساند.



وضعیت بازی:

فاصله: پیش از هر چیز لازم است مفهوم فاصله در این بازی بیان شود. هرگونه فاصله در این بازی، جمع اختلاف شماره‌ی سطرها و اختلاف شماره‌ی ستون‌های دو خانه (برای قهرمانان خانه‌ی فعلی آن‌ها) می‌باشد. به این نوع محاسبه‌ی فاصله، فاصله‌ی **منهتنی** گفته می‌شود.

تیم‌ها: در هر مسابقه، تیم‌های شرکت کننده در قالب دو تیم با یکدیگر به رقابت می‌پردازند. هر تیم شامل ۴ قهرمان می‌باشد. ویژگی‌های تیم به شرح زیر است:

- **امتیاز عمل (AP):** هر تیم مشخصه‌ای تحت عنوان "امتیاز عمل" دارد که برای استفاده از قابلیت‌ها و حرکت قهرمانان خود، از این مشخصه هزینه می‌کند. در انتهای هر نوبت، مقدار این مشخصه به حداکثر خود باز می‌گردد.
 - **دید (Vision):** تمامی خانه‌های معمولی نقشه که در دید قهرمانان تیم قرار دارد، برای کل تیم قابل مشاهده است.
- نوبت (Turn):** بازی به صورت نوبتی انجام می‌شود. هر نوبت متشکل از ۷ فاز می‌باشد. ۶ فاز اول فاز حرکت (Move Turn) و فاز آخر هر نوبت فاز عمل (Action Turn) می‌باشد. در ادامه توضیحات کامل‌تری نسبت به فازها ارائه خواهد شد.
- دو تیم روی یک نقشه به مبارزه می‌پردازند. نقشه‌ی بازی به شکل یک آرایه‌ی دو بعدی از خانه‌هاست. اولین درایه را سطر و دومین درایه را ستون می‌نامیم. خانه‌ی مبدا (۰,۰) نیز در بالا و چپ نقشه قرار دارد.
- در هر لحظه اجزای زیر در نقشه‌ی بازی وجود دارد:

خانه‌های نقشه:

خانه‌های معمولی (Normal Cells): قهرمانان دو تیم می‌توانند به این خانه‌ها وارد شوند. با این محدودیت که در هر لحظه، حداکثر یک قهرمان از هر تیم در یک خانه‌ی مشخص می‌تواند وجود داشته باشد. در یک خانه دو قهرمان از تیم‌های متفاوت به طور هم‌زمان می‌توانند حضور داشته باشند. مرزهای خانه‌ی معمولی جزء آن محسوب نمی‌شود.

خانه‌های مسدود (Wall Cells): هیچ قهرمانی به این خانه‌ها نمی‌تواند وارد شود. همچنین این خانه‌ها، دید قهرمانان و مسیر قابلیت‌های خطی (که در قسمت قابلیت‌ها توضیح داده خواهد شد) را مسدود می‌سازند. مرزهای خانه‌ی مسدود جزء آن محسوب می‌شود.

منطقه‌های خاص:

منطقه‌ی هدف (Objective Zone): مجموعه‌ای از خانه‌های نقشه که حضور قهرمان در هر نوبت در آن، برای تیم امتیاز به همراه دارد.

منطقه‌ی زنده شدن (Respawn Zone): به ازای هر تیم یکی از این محدوده‌ها در نقشه وجود دارد. قهرمان‌ها پس از مرگ و گذشتن مدت مرگ، به تصادف در یکی از خانه‌های این محدوده زنده می‌شوند. این منطقه، منطقه‌ای امن است و نمی‌توان از داخل آن به خارج و برعکس آسیبی زد.

قهرمانان (Heroes): قهرمانان در ۴ کلاس نگهبان (Guardian)، درمانگر (Healer)، ویرانگر (Blaster) و قراول (Sentry) می‌باشند. هر کلاس از قهرمان‌ها، قابلیت‌های ویژه‌ی خود را دارد.

ویژگی‌های هر قهرمان به شرح زیر است:

دید (Vision):

یک قهرمان به یک خانه دید دارد اگر در میان خط واصل مرکز خانه‌ی کنونی و مرکز خانه‌ی مقصد، هیچ خانه‌ی مسدودی نباشد. در تصویر مقابل، خانه‌هایی که با دایره‌ی قرمز مشخص شده‌اند، خارج از محدوده‌ی دید قهرمان آبی رنگند. خانه‌های مشکی نیز مسدودند.

سلامتی اولیه (Max HP):

مقدار اولیه‌ی سلامتی قهرمان پس از زنده شدن.

سلامتی فعلی (Current HP):

میزان سلامت فعلی قهرمان. با رسیدن به صفر یا کمتر قهرمان می‌میرد.

مدت زمان زنده شدن (Respawn Time):

پس از مرگ قهرمان، با گذشتن این تعداد نوبت، قهرمان در خانه‌ای از منطقه‌ی زنده شدن تیم خودی به تصادف زنده می‌شود.

مدت زمان زنده شدن باقی‌مانده (Remaining Respawn Time):

در صورت مرده بودن، قهرمان پس از این تعداد نوبت زنده خواهد شد.

هزینه‌ی حرکت (Move Cost):

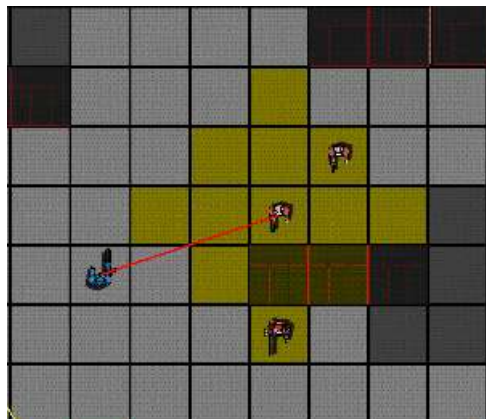
به ازای هر خانه حرکت قهرمان، به این میزان از امتیاز عمل بازیکن کم می‌شود.

قابلیت (Ability):

هر قهرمان ۳ قابلیت دارد. مشخصات هر قابلیت به صورت زیر است:

- **نوع قابلیت (Ability Type):** نوع قابلیت به یکی از سه شکل زیر است و به همین ترتیب نیز اعمال می‌شوند:
 - **دفاعی (Defensive):** قابلیت‌هایی که بر قهرمانان خودی اثر می‌گذارد. مانند افزایش سلامتی.
 - **گریز (Dodge):** جابجایی سریع و آبی خود قهرمان در نقشه. با مشخص کردن یک خانه، قهرمان در همین نوبت به آن خانه جابجا می‌شود. از این قابلیت‌ها برای گریز از حمله‌ی دشمن می‌توان استفاده کرد.
 - **تهاجمی (Offensive):** مهارت‌هایی که بر قهرمانان حریف اثر می‌گذارد. مانند کاهش سلامتی.
- **هزینه‌ی انجام (Ability Cost):** با انجام قابلیت، این میزان از امتیاز عمل بازیکن کم می‌شود.
- **مدت زمان غیرفعال (Cooldown):** پس از استفاده، قابلیت پس از این تعداد نوبت قابل استفاده می‌شود.

- **قدرت (Power):** مقدار اثرگذاری قابلیت. برای قابلیت‌های تهاجمی میزان آسیب وارده به حریف و برای قابلیت‌های تدافعی میزان افزایش سلامتی قهرمان خودی می‌باشد.



- **برد اثرگذاری (Area Of Effect):** تا این فاصله از خانه‌ی هدف را تحت تاثیر قرار می‌دهد. برای مثال قابلیتی با برد اثرگذاری ۲، بر تمامی خانه‌هایی که مجموع فاصله‌ی سطر آن از سطر خانه‌ی هدف و فاصله‌ی ستون آن از ستون خانه‌ی هدف کوچکتر مساوی ۲ باشد، اعمال می‌شود. به تصویر این مثال توجه کنید. خانه‌های زرد رنگ تحت تاثیر قرار می‌گیرند:

- **حداکثر فاصله اعمال (Range):** حداکثر فاصله از قهرمان که می‌توان به عنوان خانه‌ی هدف قابلیت انتخاب کرد.

- **پرتابی/خطی (Is Lobbing):** قابلیت‌های پرتابی مستقیماً در خانه‌ی هدف اعمال می‌شوند و در واقع چیزی در میان مسیر نمی‌تواند جلوی آن را بگیرد. برای قابلیت‌های خطی، خط واصل مرکز خانه‌ی مبدا و مرکز خانه‌ی هدف را در نظر بگیرید. این خط تا جایی ادامه پیدا می‌کند که به یک قهرمان یا خانه‌ی مسدود برسد. در صورت رسیدن به قهرمان، قابلیت در همان خانه‌ی قهرمان اعمال می‌شود. در صورت رسیدن به خانه‌ی مسدود، قابلیت در خانه‌ی معمولی پیش از آن اعمال می‌شود. در غیر این دو صورت، قابلیت در خانه‌ی هدف تعیین شده‌ی خود اعمال می‌شود.

قابلیت اعمال شده (Cast Ability): هر تیم به قابلیت‌های اعمال شده در نوبت قبلی بازی دسترسی دارد. اما این دسترسی همیشه به طور کامل وجود ندارد. در صورت تحقق حداقل یکی از شرط‌های زیر دسترسی به نام قابلیت (Ability Name) نیز وجود خواهد داشت:

- در صورت حضور قهرمان اعمال کننده در دید ما، شناسه‌ی این قهرمان (Caster ID) معتبر است.
- در صورت حضور قهرمان اعمال کننده در ابتدای اعمال قابلیت در دید ما، خانه‌ی آغاز (Start Cell) معتبر است.
- در صورت مشاهده‌ی خانه‌ی هدف در دید ما، خانه‌ی هدف (End Cell) معتبر است.
- به شناسه‌ی تمامی قهرمانانی که این قابلیت بر آن‌ها اثر داشته و در دید ما حضور داشته‌اند، دسترسی داریم (Target Hero Ids).

برای درک بهتر به دو مثال زیر توجه کنید:

- فرض کنید در نوبت پیشین، قهرمان ویرانگر ما یک بمب به پشت خانه‌ی مسدودی انداخته باشد که آن سمت در محدوده‌ی دید ما نیست. حال حتی اگر در آن محدوده قهرمانی از حریف باشد، چون در دید ما نبوده است، با `getTargetHerolds` شناسه‌ی هیچ قهرمانی مشاهده نمی‌شود.
- در همین مثال قبلی، اگر قهرمان ویرانگر ما نیز در محدوده‌ی دید حریف نباشد، حریف چون خانه‌ی هدف بمب را دیده است (قهرمانش در آن خانه حضور داشته)، در `getCastAbilities` این قابلیت بمب را مشاهده می‌کند. اما چون قهرمان ویرانگر ما در محدوده‌ی دید او نیست، `Start Cell` و `Caster ID` برای او نامعتبر است.

روند بازی:

پیش‌پردازش (PreProcess): پیش از شروع بازی، طی یک نوبت به هر تیم اجازه‌ی انجام پیش‌پردازش‌هایی داده می‌شود.

فاز انتخاب (Pick Turn): در ابتدای بازی، دو طرف به انتخاب قهرمان‌های خود می‌پردازند. این انتخاب در ۴ مرحله به صورت هم‌زمان صورت می‌گیرد و در پایان هر مرحله، قهرمانان انتخاب شده تا اینجا به دو طرف معرفی می‌شوند. دقت شود که انتخاب قهرمانان یکسان مجاز است.

پس از پایان فاز انتخاب، بازی آغاز می‌شود. هر نوبت بازی شامل ۶ فاز حرکت و ۱ فاز عمل است. پس از اتمام هر نوبت، امتیاز عمل هر تیم به مقدار اولیه باز می‌گردد.

فازهای حرکت (Move Turn): در هر کدام از این فازها، هر قهرمان را می‌توان حداکثر به یکی از خانه‌های مجاور ضلعی جابه‌جا کرد. پس از تصمیم‌گیری دو طرف و در انتهای هر فاز، حرکات به شکل هم‌زمان صورت می‌گیرند.

فاز عمل (Action Turn): در این فاز، حداکثر یک قابلیت از هر قهرمان را می‌توان استفاده کرد. در انتهای فاز، به ترتیب نوع قابلیت‌ها، یعنی ابتدا دفاعی‌ها، سپس گریزها و در انتها تهاجمی‌ها اعمال می‌شوند.

در انتهای هر نوبت، به ازای تعداد قهرمانان زنده‌ی موجود در منطقه‌ی هدف، و ۱۰ برابر تعداد قهرمانان کشته شده از حریف در این نوبت، به امتیاز تیم مربوطه اضافه می‌شود.

پایان بازی:

اولین تیمی که به ۲۰۰ امتیاز برسد پیروز خواهد شد. در غیر این صورت بازی به مدت ۱۰۰ نوبت انجام می‌پذیرد. در انتها تیمی که امتیاز بیشتری کسب کرده باشد، پیروز خواهد شد. امتیاز هر تیم شامل مجموع امتیاز کشتن قهرمانان حریف و مجموع امتیاز حضور در محدوده‌ی هدف می‌باشد. در صورت تساوی امتیازات، تیمی که در مجموع ۱۰۰ نوبت امتیاز عمل کمتری استفاده کرده، پیروز خواهد شد. در صورت تساوی در مراحل انتخابی و حضوری، تیمی که کد خود را زودتر ارسال کرده، برنده اعلام می‌شود.

ویژگی‌های بازی:

نام ویژگی	مقدار ویژگی
امتیاز عمل اولیه (AP)	۱۰۰
حداکثر امتیاز پیروزی (Max Score)	۲۰۰
حداکثر نوبت بازی (Max Turn)	۱۰۰
تعداد فازهای انتخاب (Pick Turn)	۴
تعداد فازهای حرکت در هر نوبت (Move Turn)	۶
تعداد فازهای عمل در هر نوبت (Action Turn)	۱
امتیاز کشتن یک قهرمان از حریف (Kill Score)	۱۰
امتیاز منطقه‌ی هدف (Objective Zone Score)	۱

در ادامه به معرفی کلاس‌های مختلف قهرمان‌ها می‌پردازیم. دو تصویر هر کلاس، ظاهر قهرمانان آن کلاس در دو تیم مختلف را نشان می‌دهد و این تفاوت صرفاً ظاهری می‌باشد.

ویرانگر – Blaster



Red Demon



Slayer

توضیحات: ویرانگر نسبت به دیگر قهرمان ها از سرعت خوبی برخوردار است. قابلیت خاص این قهرمان، پرتاب بمب با قدرت آسیب و محدوده اثرگذاری بالاست که می توان چند قهرمان کنار هم را در یک نوبت هدف قرار داد.

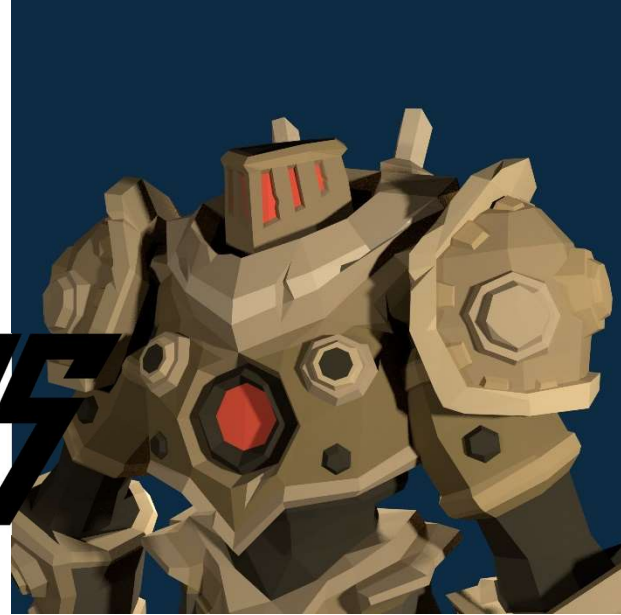
ویژگی های کلی:

مقدار ویژگی	نام ویژگی
۲۵۰	سلامتی اولیه
۵	مدت زمان زنده شدن
۴	هزینه حرکت

ویژگی های قابلیت ها:

نام ویژگی	حمله	گریز	بمب
نوع قابلیت	تهاجمی	گریز	تهاجمی
هزینه انجام	۱۵	۲۵	۲۵
مدت زمان غیرفعال	۱	۶	۴
قدرت	۳۰	-	۴۰
برد اثرگذاری	۱	-	۲
حداکثر فاصله اعمال	۴	۴	۵
پرتابی/خطی	خطی	پرتابی	پرتابی

قراول – Sentry

**Ancient Warrior****Mechanical Golem**

توضیحات: قراول قهرمانی دوربرد اما نسبتاً کند است. قابلیت خاص این قهرمان، شلیک یک اشعه با برد نامحدود و قدرت آسیب بالاست. این اشعه به اولین قهرمان یا دیوار در مسیر خود اصابت کرده و اثر خود را اعمال می‌کند.

ویژگی های کلی:

نام ویژگی	مقدار ویژگی
سلامتی اولیه	۱۲۰
مدت زمان زنده شدن	۵
هزینه‌ی حرکت	۶

ویژگی های قابلیت‌ها:

نام ویژگی	حمله	گریز	اشعه
نوع قابلیت	تهاجمی	گریز	تهاجمی
هزینه‌ی انجام	۱۵	۲۵	۲۵
مدت زمان غیرفعال	۱	۷	۵
قدرت	۳۰	-	۵۰
برد اثرگذاری	۰	-	۰
حداکثر فاصله‌ی اعمال	۷	۳	نامحدود
پرتابی/خطی	خطی	پرتابی	خطی

درمانگر – Healer



Ancient Queen



Mystic

توضیحات: درمانگر قهرمانی با سرعت و سلامتی متوسط است. قابلیت خاص این قهرمان، درمانگری است که می‌تواند سلامتی یک قهرمان خودی را حداکثر ۴۰ واحد تا مقدار سلامتی اولیه افزایش دهد.

ویژگی های کلی:

مقدار ویژگی	نام ویژگی
۲۰۰	سلامتی اولیه
۵	مدت زمان زنده شدن
۴	هزینه‌ی حرکت

ویژگی های قابلیت‌ها:

درمانگری	گریز	حمله	نام ویژگی
تدافعی	گریز	تهاجمی	نوع قابلیت
۲۵	۲۵	۱۵	هزینه‌ی انجام
۳	۶	۱	مدت زمان غیرفعال
۴۰	-	۲۰	قدرت
۰	-	۰	برد اثرگذاری
۴	۴	۴	حداکثر فاصله‌ی اعمال
پرتابی	پرتابی	پرتابی	پرتابی/خطی

نگهبان – Guardian

**Big Ork****Elemental Golem**

توضیحات: نگهبان قهرمانی با سلامتی اولیه زیاد اما سرعتی بسیار کند است. قابلیت خاص این قهرمان، تقویت یک قهرمان خودی است که باعث می شود در این نوبت به مقدار نامحدود آسیب های وارده به قهرمان هدف بی اثر شود.

ویژگی های کلی:

نام ویژگی	مقدار ویژگی
سلامتی اولیه	۴۰۰
مدت زمان زنده شدن	۵
هزینه ی حرکت	۸

ویژگی های قابلیت ها:

نام ویژگی	حمله	گریز	تقویت
نوع قابلیت	تهاجمی	گریز	تدافعی
هزینه ی انجام	۱۵	۲۵	۲۵
مدت زمان غیرفعال	۱	۸	۷
قدرت	۲۵	-	نامحدود
برد اثرگذاری	۱	-	۰
حداکثر فاصله ی اعمال	۱	۲	۴
پرتابی/خطی	پرتابی	پرتابی	پرتابی

جزئیات فنی اتفاقات بازی

- در صورت نامعتبر بودن یک دستور، این دستور نادیده گرفته می‌شود و اولین دستور معتبر بعدی انجام می‌شود. ترتیب دستورات به همان ترتیبی است که شما در کلاینت ارسال می‌کنید.
- در صورتی که یک حرکت یا قابلیت به هر دلیل انجام نشود، امتیاز عمل آن از تیم کسر نمی‌شود.
- دستور حرکت با مقصد خانه‌ی مسدود (Wall Cell) یا خانه‌ی پر شده با قهرمان خودی، نامعتبر است.
- دستور قابلیت گریز با مقصد خانه‌ی مسدود (Wall Cell) یا خانه‌ی پر شده با قهرمان خودی، نامعتبر است.
- در صورت فاصله‌ی بیشتر خانه‌ی هدف قابلیت گریز از حداکثر فاصله‌ی اعمال آن، آخرین خانه‌ی موجود در خط واصل خانه‌ی قهرمان و خانه‌ی هدف اولیه، به عنوان خانه‌ی هدف در نظر گرفته می‌شود. همانند شرط قبلی در صورت مسدود یا پر بودن خانه‌ی هدف جدید، این دستور نامعتبر تلقی می‌شود.
- پس از اجرای اولین دستور معتبر، تمامی دستورات با قهرمان یکسان اعمال کننده با این دستور که پس از آن آمده، نامعتبر تلقی می‌شود.
- در رابطه با قابلیت‌های غیر از گریز، مقصد می‌تواند هر نوع خانه‌ای باشد و معتبر است.
- در صورت کمبود امتیاز عمل برای انجام یک دستور، آن دستور نامعتبر تلقی می‌شود.
- آسیب رسانی به قهرمانان خودی وجود ندارد و قابلیت‌های خطی تهاجمی از قهرمانان خودی عبور می‌کند.

نکته‌ی مهم: پس از گذشت یک هفته از آغاز مرحله‌ی غیرحضور، احتمال دارد مقادیر ویژگی‌های قهرمانان تغییرات جزئی داشته باشند.

همچنین برای مرحله‌ی حضوری نیز تغییراتی بر بازی اعمال خواهد شد.

شروع کد نویسی (بایدها و نبایدها)

۱. شما باید کد هوش مصنوعی خود را در توابع `preProcessTurn` و `pickTurn` و `moveTurn` و `actionTurn` در فایل `Al.java` (برای زبان های دیگر فایلی به همین نام) قرار دهید.

۲. شما می توانید کد کلاینت داده شده را تغییر دهید، به آن فایل اضافه کنید یا از آن فایل حذف کنید، به شرط آن که تغییرات داده شده در کامپایل و اجرای کلاینت و ارتباط آن با سرور اختلالی ایجاد نکند. در مورد هر کلاینت نکاتی ذکر شده که به آن ها نیز باید توجه شود. هم چنین باید تغییرات احتمالی فایل های دیگر کلاینت (فایل هایی غیر از فایلی که در آن کد می زنید) را در نظر بگیرید.

۳. شما می توانید برای به روز بودن کلاینت ها یا سرور خود به آخرین نسخه منتشر شده در `repository` مسابقات مراجعه کنید. این کار کاملاً اختیاری است و اگر به کار با `git` آشنایی ندارید توصیه نمی شود.

<https://github.com/SharifAIChallenge/>

(repo کلاینت ها با پسوند `AIC19-Client` آغاز می شود)

نحوه ی اجرا و توسعه (کلاینت جاوا)

پیشنهاد ما برای توسعه ی کلاینت جاوا استفاده از `IntelliJ` است. پس از باز کردن `IntelliJ` گزینه ی `Import Project` را انتخاب کنید و پوشه ای که کلاینت در آن قرار دارد را انتخاب کنید. در صفحه ی `Libraries` مطمئن شوید `gson-2-3-1.jar` انتخاب شده است.

نحوه ی اجرا و توسعه (کلاینت سی پلاس پلاس)

برای توسعه ی کلاینت سی پلاس پلاس پیشنهاد ما استفاده از `Clion` است. پس از نصب `Clion` در صورتی که از ویندوز استفاده می کنید از توضیحات لینک زیر برای تنظیم `Clion` استفاده کنید:

<https://www.jetbrains.com/help/clion/quick-tutorial-on-configuring-clion-on-windows.html>

پس از نصب و راه اندازی `Clion` کافیسیت پوشه ای که پروژه در آن قرار دارد را در `Clion` باز کنید.

در صورتی که نمی خواهید از `Clion` استفاده کنید نیاز به نصب `Cmake` دارید. پس از نصب `Cmake` یک ترمینال باز کنید و به پوشه ای که پروژه در آن قرار دارد بروید و دستور `Cmake` را اجرا کنید. پس از این می توانید با استفاده از دستور `make` پروژه را `build` کنید. بعد از اجرای دستور `make` فایل باینری `client` در پوشه ی جاری ساخته خواهد شد.

نحوه‌ی اجرا و توسعه (کلاينت پايتون)

برای اجرای کلاينت پايتون نیاز به پايتون ۳ دارید. پس از نصب پايتون ۳ کافيست فايل `Controller.py` را اجرا نماييد. شما می‌توانيد به اين کلاينت فايل‌های مورد نظرتان را اضافه کنيد يا فايل‌های آن را تغيير دهيد به شرط آن که کد شما با سرور سازگار باشد. به سادگی بتوانيد نسخه‌های احتمالي بعدي کلاينت را جايگزين نماييد. برای توسعه و کدنويسي به اين زبان، PyCharm را توصيه می‌کنيم.

ارسال فايل ها

برای ارسال فايل‌های خود، پوشه‌ی کلاينت را فشرده کنيد و فايل فشرده (با پسوند zip) را ارسال نماييد.

رابط برنامه نویسی:

نکات مهم:

۱. رفرنس‌های موجود در نقشه، در هر نوبت از ابتدا ساخته می‌شوند و رفرنس‌های نوبت قبل معتبر نیستند. (برای حفظ دسترسی به آبجکت‌ها، توصیه می‌شود از id آن‌ها، که همواره ثابت است، استفاده کنید.)

۲. در تمامی قسمت‌هایی که علامت ستاره (*) وجود دارد، تمامی حالات تابع شامل شناسه‌ی قهرمان به جای آبجکت قهرمان، مختصات سطر و ستون به جای آبجکت خانه و نام قابلیت به جای آبجکت پیاده شده است.

۳. در کلاینت پایتون، متدهای **get** و **set** پیاده نشده‌اند و به متغیرها دسترسی مستقیم وجود دارد.

۴. در کلاینت پایتون، متد های ذکر شده در مورد شماره ۲، به صورت دیگری مقداردهی می‌شوند. برای مثال فرض کنید در یک جا **Cell** و در جای دیگر **row** و **column** به عنوان ورودی داده شود. نحوه نگارش به صورت زیر خواهد بود:

method_name(Cell = ..., ...)

method_name(row = ..., column = ..., ...)

در ادامه، اجزایی که برای برنامه‌نویسی به آنان نیاز دارید به همراه توضیحی مختصر آمده‌اند. سینتکس این توابع متعلق به کلاینت جاوا می‌باشد. برای کلاینت پایتون همین اسامی به صورت snake_case می‌باشند.

Class (or interface) Ability

int getRemCooldown(); مقدار باقی‌مانده از مدت زمان غیر فعال. این مقدار برای دشمن نامعتبر است.

boolean isReady(); در صورتی که قابلیت آماده‌ی انجام در این نوبت باشد، true برمی‌گرداند.

همه‌ی متدهای کلاس **AbilityConstants** در این کلاس نیز موجود است.

Class (or interface) AbilityConstants

AbilityName getName(); نام قابلیت

AbilityType getType(); نوع قابلیت

int getRange(); حداکثر فاصله‌ی اعمال

int getAPCost(); هزینه‌ی انجام قابلیت

int getCooldown(); مدت زمان غیر فعال بودن قابلیت پس از استفاده

int getAreaOfEffect(); برد اثرگذاری قابلیت

<code>int getPower();</code>	قدرت قابلیت
<code>boolean isLobbing();</code>	پرتابی/خطی بودن قابلیت. برای پرتابی مقدار <code>true</code> و برای خطی مقدار <code>false</code> .

enum AbilityName

<code>SENTRY_ATTACK</code>	حمله ی قراول
<code>SENTRY_DODGE</code>	گریز قراول
<code>SENTRY_RAY</code>	اشعه ی قراول
<code>BLASTER_ATTACK</code>	حمله ی ویرانگر
<code>BLASTER_DODGE</code>	گریز ویرانگر
<code>BLASTER_BOMB</code>	بمب ویرانگر
<code>HEALER_ATTACK</code>	حمله ی درمانگر
<code>HEALER_DODGE</code>	گریز درمانگر
<code>HEALER_HEAL</code>	درمانگری درمانگر
<code>GUARDIAN_ATTACK</code>	حمله ی محافظ
<code>GUARDIAN_DODGE</code>	گریز محافظ
<code>GUARDIAN_FORTIFY</code>	تقویت محافظ

enum AbilityType

<code>DEFENSIVE</code>	دفاعی
<code>DODGE</code>	گریز
<code>OFFENSIVE</code>	تهاجمی

Class (or interface) CastAbility

<code>int getCasterId();</code>	شناسه ی قهرمان اعمال کننده ی قابلیت. در صورت عدم مشاهده ی قهرمان اعمال کننده، نامعتبر است.
<code>int[] getTargetHeroIds();</code>	شناسه های قهرمان های تحت تاثیر قابلیت که در دید هستند. برای گریز خود قهرمان جزء تحت تاثیرها به حساب نمی آید.
<code>Cell getStartCell();</code>	خانه ی مبدا اعمال قابلیت. در صورت عدم حضور این خانه در دید، نامعتبر است.
<code>Cell getEndCell();</code>	خانه ی مقصد اعمال قابلیت. در صورت عدم حضور این خانه در دید، نامعتبر است.
<code>AbilityName getAbilityName();</code>	نام قابلیت. در صورت نامعتبر بودن همه اطلاعات بالا، نامعتبر است.

Class (or interface) Cell

<code>int getRow();</code>	شماره‌ی سطر خانه
<code>int getColumn();</code>	شماره‌ی ستون خانه
<code>boolean isWall();</code>	در صورت مسدود بودن خانه <code>true</code> و در صورت معمولی بودن <code>false</code> برمی‌گرداند.
<code>boolean isInMyRespawnZone();</code>	اگر در منطقه‌ی زنده شدن خودی باشد، <code>true</code> برمی‌گرداند.
<code>boolean isInOppRespawnZone();</code>	اگر در منطقه‌ی زنده شدن دشمن باشد، <code>true</code> برمی‌گرداند.
<code>boolean isInObjectiveZone();</code>	اگر در منطقه‌ی هدف باشد، <code>true</code> برمی‌گرداند.
<code>boolean isInVision();</code>	اگر در محدوده‌ی دید فعلی باشد، <code>true</code> برمی‌گرداند.

enum Direction

UP	بالا
DOWN	پایین
LEFT	چپ
RIGHT	راست

Class (or interface) Hero

<code>Ability getAbility(AbilityName);</code>	قابلیت با این نام را در صورت وجود برمی‌گرداند.
<code>int getId();</code>	شناسه‌ی قهرمان را برمی‌گرداند.
<code>Ability[] getAbilities();</code>	قابلیت‌های قهرمان را برمی‌گرداند.
<code>Ability[] getDefensiveAbilities();</code>	قابلیت‌های دفاعی قهرمان را برمی‌گرداند.
<code>Ability[] getDodgeAbilities();</code>	قابلیت‌های گریز قهرمان را برمی‌گرداند. اندازه‌ی این آرایه همواره یک است.
<code>Ability[] getOffensiveAbilities();</code>	قابلیت‌های تهاجمی قهرمان را برمی‌گرداند.
<code>int getCurrentHP();</code>	سلامتی کنونی قهرمان. برای قهرمانان دشمن خارج از دید، نامعتبر است.
<code>Cell getCurrentCell();</code>	خانه‌ی کنونی قهرمان. برای قهرمانان دشمن خارج از دید، نامعتبر است.
<code>Cell[] getRecentPath();</code>	مسیر حرکت قهرمان در فاز پیشین. برای قهرمانان دشمن، حرکت‌هایی که در محدوده‌ی دید بوده برمی‌گردد.
<code>int getRemRespawnTime();</code>	مقدار باقی‌مانده از مدت زمان زنده شدن. برای قهرمانان دشمن نیز همواره معتبر است.
همه‌ی متدهای کلاس <code>HeroConstants</code> در این کلاس نیز موجود است.	

Class (or interface) HeroConstants

HeroName getName();	نوع قهرمان را برمی گرداند.
AbilityName[] getAbilityNames();	نام قابلیت های قهرمان را برمی گرداند.
int getMaxHP();	سلامتی اولیه قهرمان
int getMoveAPCost();	هزینه حرکت قهرمان
int getRespawnTime();	مدت زمان زنده شدن

enum HeroName

SENTRY	قراول
BLASTER	ویرانگر
HEALER	درمانگر
GUARDIAN	محافظ

enum Phase

PICK	فاز انتخاب
MOVE	فاز حرکت
ACTION	فاز عمل

Class (or interface) Map

Cell[][] getCells();	آرایه ی دوبعدی خانه های نقشه را برمی گرداند.
Cell getCell(int, int);	خانه ی مورد نظر در مختصات داده شده را برمی گرداند.
boolean isInMap(int, int);	در صورت خارج محدوده بودن مختصات، false برمی گرداند.
int getRowNum();	تعداد سطرها ی نقشه
int getColumnNum();	تعداد ستون های نقشه
Cell[] getMyRespawnZone();	خانه های محدودی زنده شدن خودی را برمی گرداند.
Cell[] getOppRespawnZone();	خانه های محدودی زنده شدن دشمن را برمی گرداند.
Cell[] getObjectiveZone();	خانه های محدودی هدف را برمی گرداند.

Class (or interface) World

Hero getHero(int);	قهرمان با شناسه ی داده شده را برمی گرداند.
Hero getMyHero(Cell);	قهرمان خودی در خانه ی داده شده (مختصات) را برمی گرداند.
Hero getMyHero(int, int);	
Hero getOppHero(Cell);	قهرمان دشمن را در خانه ی داده شده (مختصات) در صورتی
Hero getOppHero(int, int);	که در دید باشد، برمی گرداند.

<code>void castAbility(Hero, Ability, Cell);</code>	اعمال قابلیت قهرمان به خانه‌ی موردنظر.*
<code>void moveHero(Hero, Direction);</code>	حرکت قهرمان در جهت مورد نظر به اندازه‌ی یک خانه.
<code>void moveHero(int, Direction);</code>	
<code>void pickHero(HeroName);</code>	انتخاب یکی از این نوع قهرمان در فاز انتخاب.
<code>Direction[]</code> <code>getPathMoveDirections(Cell, Cell);</code>	دنباله‌ی جهت‌ها برای رسیدن از خانه‌ی اول به خانه‌ی دوم.*
<code>int manhattanDistance(Cell, Cell);</code>	فاصله منتهی دو خانه. پیاده‌سازی با مختصات نیز موجود است.*
<code>Cell getImpactCell(Ability, Cell, Cell);</code>	در صورت استفاده از قابلیت خطی، با استفاده از این تابع می‌توان خانه‌ی محل اثر را بر اساس دید فعلی بدست آورد.*
<code>Hero[] getAbilityTargets(Ability, Cell, Cell);</code>	در صورت اعمال قابلیت از خانه‌ی اول به دوم، قهرمانانی که مورد اثر قرار می‌گیرند و در دید هستند را برمی‌گرداند.*
<code>boolean isInVision(Cell, Cell);</code>	در صورت قابلیت دیدن خانه‌ی دوم از خانه‌ی اول، true برمی‌گرداند.*
<code>Hero[] getMyHeroes();</code>	قهرمانان خودی را برمی‌گرداند.
<code>Hero[] getOppHeroes();</code>	همه‌ی قهرمانان انتخاب شده‌ی دشمن تا اینجا را برمی‌گرداند. در صورت خارج از محدوده دید بودن، مختصات خانه 1- خواهد بود.
<code>Hero[] getMyDeadHeroes();</code>	قهرمانان مرده‌ی خودی را برمی‌گرداند.
<code>Hero[] getOppDeadHeroes();</code>	قهرمانان مرده‌ی دشمن را برمی‌گرداند.
<code>Map getMap();</code>	نقشه‌ی بازی را برمی‌گرداند.
<code>CastAbility[]</code> <code>getMyCastAbilities();</code>	قابلیت‌های اعمال شده‌ی خودی در نوبت قبلی.
<code>CastAbility[]</code> <code>getOppCastAbilities();</code>	قابلیت‌های اعمال و مشاهده شده‌ی دشمن در نوبت قبلی.
<code>HeroConstants[]</code> <code>getHeroConstants();</code>	ویژگی‌های انواع مختلف قهرمانان را برمی‌گرداند.
<code>AbilityConstants[]</code> <code>getAbilityConstants();</code>	ویژگی‌های انواع مختلف قابلیت‌ها را برمی‌گرداند.
<code>int getAP();</code>	مقدار امتیاز عمل فعلی را برمی‌گرداند.
<code>int getMyScore();</code>	امتیاز فعلی خودی را برمی‌گرداند.
<code>int getOppScore();</code>	امتیاز فعلی دشمن را برمی‌گرداند.
<code>int getCurrentTurn();</code>	شماره نوبت فعلی را برمی‌گرداند.
<code>Phase getCurrentPhase();</code>	فاز فعلی را برمی‌گرداند.
<code>int getMaxScore();</code>	مقدار سقف امتیاز بازی برای پیروزی را برمی‌گرداند.

<code>int getMaxAP();</code>	مقدار اولیه‌ی امتیاز عمل را برمی‌گرداند.
<code>int getMaxTurns();</code>	تعداد کل نوبت‌های بازی را برمی‌گرداند.
<code>int getKillScore();</code>	امتیاز حاصل از کشتن یک قهرمان دشمن را برمی‌گرداند.
<code>int getObjectiveZoneScore();</code>	امتیاز حاصل از حضور یک قهرمان خودی در منطقه‌ی هدف را برمی‌گرداند.
<code>int getMovePhaseNum();</code>	شماره‌ی فاز حرکت فعلی در نوبت جاری را برمی‌گرداند.

Class (or interface) AI

<code>void preProcess(World);</code>	این تابع فقط در ابتدای شروع بازی و پیش از فاز انتخاب اول صدا زده می‌شود. محدودیت زمانی ۵ ثانیه
<code>void pickTurn(World);</code>	تابع فازهای انتخاب. به ازای هر کدام از این فازها، یکبار صدا زده می‌شود. محدودیت زمانی ۰,۲ ثانیه.
<code>void moveTurn(World);</code>	تابع فازهای حرکت. به ازای هر کدام از این فازها، یکبار صدا زده می‌شود. محدودیت زمانی اولین فاز هر نوبت، ۰,۵ ثانیه و بقیه ۰,۲ ثانیه
<code>void actionTurn(World);</code>	تابع فازهای عمل. به ازای هر کدام از این فازها، یکبار صدا زده می‌شود. محدودیت زمانی ۰,۲ ثانیه.
