توصیف پروژه پایانی

برنامه‌سازی پیشـرفته

نیم‌سال دوم ۱۴۰۲-۱۴۰۱

مدرس: حسن بشیری

دستیاران آموزشی: رضا محمودیان، فرهاد یونسی، یونس سهرابی

خردادماه ۱۴۰۲

۱ عنوان پروژه

میراث آدم خطی

۲ هدف پروژه

- کار لیست‌ها، آریه‌ها، مجموعه‌ها، کلاس‌ها و فایل‌ها
- افزایش مهارت حل مساله
- نمره (-: (۴ نمره)

۳ خلاصه‌ای از توصیف پروژه

بازی میراث آدم خطی (StickMan:Legacy) یک بازی استراتژیک است که در آن هر کس ارتش خودش را تجهیز می‌کند و به جنگ دشمنان می‌رود. از شما می‌خواهیم برنامه‌ای بنویسید که درخواست‌های بازیکن را مدیریت کند.

شما برای ملحق کردن افراد به ارتش باید به آن‌ها سکه بدهید. همچنین نمی‌تواند بیش از ۵۰ واحد نیرو زنده در ارتش خود داشته‌باشید. در ابتدای بازی ۵۰۰ سکه دارید و هیچ کسی در ارتش شما نیست. دولت از لحظه ابتدایی بازی، هر ۲۰ ثانیه یکبار ۱۸۰ سکه به ارتش شما می‌دهد. (به‌جز لحظه‌ی شروع)

در این ارتش ۶ سِمَت مختلف «معدن‌کار»، «شمشیرزن»، «تیرانداز»، «سپردار»، «جادوگر» و «غول» برای افراد وجود دارد. هر کدام از این سِمَت‌ها ویژگی و شرایط مختلفی دارند که در ادامه توضیح داده می‌شود.

هر کس بسته به سمتی که دارد، مقداری جان دارد. اگر در جنگ آسیبی به کسی وارد شود، به همان اندازه از جانش کم می‌شود و اگر جان کسی بعد از یک صدمه، کمتر یا مساوی ۰ شود، می‌میرد. ارتش شما باید به مبارزه با یک اژدها برود. این اژدها h واحد جان دارد و هر چندوقت یکبار به یکی از افراد ارتش شما صدمه می‌زند. در میدان جنگ ۴ معدن وجود دارد و روی هر معدن حداکثر دو معدن‌کار می‌توانند کار کنند. اگر تعداد معدن‌کارها بیشتر از ۸ باشد، ۸ نفر روی معادن کار می‌کنند و بقیه بیکار می‌ایستند.

۳-۱ ویژگی سِمَت‌ها

- **معدن‌کار** (Miner): هر معدن‌کار یک واحد نیرو حساب می‌شود. جان هر معدن‌کار ۱۰۰ است. استخدام کردن یک معدن‌کار ۱۵۰ سکه نیاز دارد. هر معدن‌کار بعد از استخدام، در صورتی که بتواند روی یک معدن مشغول شود، هر ۱۰ ثانیه، ۱۰۰ سکه به دارای شما اضافه می‌کند.

- **شمشیرزن (Swordwrath):** هر شمشیرزن یک واحد نیرو حساب می‌شود. جان هر شمشیرزن ۱۲۰ است. استخدام کردن یک شمشیرزن ۱۲۵ سکه نیاز دارد. شمشیرزن هر ثانیه بعد از استخدام، یک ضربه به اژدها می‌زند و از جان او ۲۰ واحد کم می‌کند.
- **تیرانداز (Archidon):** هر تیرانداز یک واحد نیرو حساب می‌شود. جان هر تیرانداز ۸۰ است. استخدام کردن یک تیرانداز ۳۰۰ سکه نیاز دارد. تیرانداز هر ثانیه بعد از استخدام، یک تیر به اژدها می‌زند و از جان او ۱۰ واحد کم می‌کند.
- **سپردار (Spearton):** هر سپردار دو واحد نیرو حساب می‌شود. جان هر سپردار ۲۵۰ است. استخدام کردن یک سپردار ۵۰۰ سکه نیاز دارد. سپردار هر سه ثانیه بعد از استخدام، یک نیزه به اژدها می‌زند و از جان او ۳۵ واحد کم می‌کند.
- **جادوگر (Magikill):** هر جادوگر چهار واحد نیرو حساب می‌شود. جان هر جادوگر ۸۰ است. استخدام کردن یک جادوگر ۱۲۰۰ سکه نیاز دارد. جادوگر هر پنج ثانیه بعد از استخدام، یک آتش به اژدها می‌زند و از جان او ۲۰۰ واحد کم می‌کند.
- **غول (Giant):** هر غول چهار واحد نیرو حساب می‌شود. جان هر غول ۱۰۰۰ است. استخدام کردن یک غول ۱۵۰۰ سکه نیاز دارد. غول هر چهار ثانیه بعد از استخدام، یک ضربه به اژدها می‌زند و از جان او ۱۵۰ واحد کم می‌کند.

۳-۲ درخواست‌ها

در این پروژه از شما می‌خواهیم تعداد Q درخواست را که بازیکن از ارتش دارد، مدیریت کنید. درخواست‌ها به شرح زیر هستند:

- **اضافه کردن بازیکن (add):**

```
add <role> <timestamp>
```

این درخواست یعنی بازیکن می‌خواهد فردی با سمت `<role>` در لحظه `<timestamp>` به بازی اضافه کند. (تضمین می‌شود که `<role>` یکی از ۶ کلمه بالا باشد).

- اگر اژدها در این لحظه کشته شده، پیام `game over` را چاپ کنید.
- اگر تعداد سکه‌های ما در این لحظه، از سکه‌های مورد نیاز برای استخدام این فرد کمتر است. پیام `not enough money` را چاپ کنید.
- اگر با اضافه شدن این فرد به ارتش، تعداد واحدهای نیرو از ۵۰ بیشتر می‌شود. پیام `too many army` را چاپ کنید.
- اگر هیچ‌کدام از دو حالت بالا اتفاق نیفتاد، این فرد را به ارتش اضافه کنید و یک عدد که شماره این فرد است را چاپ کنید. افرادی که به ارتش اضافه می‌شوند به ترتیب از ۱ شماره گذاری می‌شوند. (اگر افراد کشته هم شوند شماره آن‌ها حفظ می‌شود ولی اگر به هر دلیل از خطاهای بالا اضافه نشوند دیگر شماره ندارند.)
- اگر چند خطا از خطاهای بالا باهم وجود داشت، تنها پیامی را چاپ کنید که زودتر آمده است.

- **ضربه خوردن (damage):**

`damage <idx> <d> <timestamp>`

این درخواست، یعنی فرد شماره `<idx>` توسط اژدها به اندازه `<d>` در لحظه `<timestamp>` ضربه خورده است.

  - اگر اژدها در این لحظه کشته شده، پیام `game over` را چاپ کنید.
  - اگر شخصی با شماره‌ی `<idx>` به ارتش شما هنوز نیامده یا اکنون زنده نیست، پیام `no matter` را چاپ کنید.
  - اگر هیچ‌کدام از حالت‌های بالا اتفاق نیفتاد، از جان `<idx>` به‌اندازه `<d>` کم کنید. اگر بعد از این ضربه این فرد کشته شد، رشته `dead` و درغیراینصورت جان باقی‌مانده آن را چاپ کنید.
  - توجه کنید اگر ضربه اژدها باعث کشته شدن یک معدن‌کار شود و معدن‌کار بیکار داشته باشیم، معدن‌کاری که زودتر در ارتش استخدام شده، جایگزین می‌شود.

- **گزارش وضعیت اژدها (enemy-status):**

`enemy-status <timestamp>`

این درخواست، یعنی می‌خواهیم وضعیت اژدها را در لحظه `<timestamp>` بدانیم.

  - اگر اژدها در این لحظه کشته شده، پیام `game over` را چاپ کنید.
  - در غیراین‌صورت جان باقی‌مانده اژدها را در این لحظه چاپ کنید.

- **گزارش وضعیت ارتش (army-status):**

`army-status <timestamp>`

این درخواست، یعنی می‌خواهیم تعداد افراد زنده به تفکیک سمتشان در ارتش، را در لحظه `<timestamp>` بدانیم.

  - اگر اژدها در این لحظه کشته شده، پیام `game over` را چاپ کنید.
  - در غیراین‌صورت ۶ عدد صحیح که با یک فاصله از هم جدا شده‌اند و هر کدام به ترتیبی که در بالا معرفی شدند، تعداد افراد زنده را در این لحظه چاپ کنید. (هر فرد را مستقل از سمت یکبار بشمارید.)

- **گزارش وضعیت مالی (money-status):**

`money-status <timestamp>`

این درخواست، یعنی می‌خواهیم وضعیت مالی را در لحظه `<timestamp>` بدانیم.

  - اگر اژدها در این لحظه کشته شده، پیام `game over` را چاپ کنید.
  - در غیراین‌صورت تعداد سکه‌های ذخیره شده را در این لحظه چاپ کنید.

۳-۳ نکته‌ها

- هر درخواست در یک زمانی داده می‌شود که با `<timestamp>` نشان می‌دهیم. فرمت هر کدام به صورت `mm:ss:xxx` است که یعنی اتفاق در دقیقه `mm`، ثانیه `ss` و میلی‌ثانیه `xxx` اتفاق افتاد است.
- تضمین می‌شود هیچ دو اتفاقی حتی سکه اضافه کردن‌ها و صدمه زدن‌هایی که به تناوب تکرار می‌شوند، همزمان نباشد.

۳-۴ ورودی

ورودی‌های برنامه باید از یک فایل متنی خوانده شود. در سطر اول ورودی، دو عدد صحیح q و h با یک فاصله داده می‌شود که به ترتیب نشان‌دهنده تعداد درخواست‌ها و جان اژدها است. در q سطر بعدی، در هر سطر یکی از دستوراتی که در تشریح پروژه گفته شده می‌آید. تضمین می‌شود زمان همه درخواست‌ها از نظر زمانی مرتب هستند و هیچ دو اتفاقی حتی اتفاق‌هایی که به تناوب تکرار می‌شوند، همزمان رخ نمی‌دهد.

۳-۵ خروجی

بعد از اجرای برنامه، خروجی باید داخل یک فایل نوشته شود.

خروجی q سطر دارد و در هر سطر، خروجی متناسب با هر درخواست ورودی به ترتیب چاپ شده است.

۳-۶ نمونه‌های ورودی/خروجی

وردودی - نمونه ۱

```
12 132
money-status 00:19:999
money-status 00:20:001
add miner 00:20:002
money-status 00:20:003
add miner 00:20:004
money-status 00:20:005
add miner 00:20:006
money-status 00:20:007
add miner 00:20:008
money-status 00:20:009
add miner 00:29:010
add miner 00:30:011
```

خروجی - نمونه ۱

```
500
680
1
530
2
380
3
230
4
80
not enough money
5
```

- همانطور که در سوال گفته شد؛ دولت بعد از شروع بازی هر ۲۰ ثانیه، ۱۸۰ سکه به ارتش کمک می‌کند. پس تا لحظه قبل از ثانیه ۲۰، مقدار پول ارتش همان ۵۰۰ سکه‌ای است که از ابتدا دارد و اولین لحظه بعد از ثانیه ۲۰، ۱۸۰ سکه به ارتش اضافه می‌شود.
- در ۴ درخواست بعدی، در هر درخواست یک معدن‌کار به ارتش اضافه می‌شود. هر معدن‌کار برای استخدام ۱۵۰ سکه نیاز دارد. پس بعد از استخدام معدن‌کار ۴ام دیگر پول کافی برای استخدام معدن‌کار ۵ام در لحظه‌ی ۲۹:۰۰۹ نداریم. اما در لحظه‌ی ۳۰:۰۱۰، هر ۴ معدن‌کار ۱۰ ثانیه هست که مشغول به کار شدند و هر کدام ۱۰۰ سکه به ارتش اضافه کرده‌اند. پس می‌توانیم در این لحظه یک معدن‌کار داشته باشیم.

وردودی - نمونه ۲

```
32 10000
money-status 00:00:001
add miner 00:00:002
add miner 00:00:003
add miner 00:00:004
money-status 00:00:005
add miner 00:10:006
add miner 00:10:007
add swordwrath 02:00:008
add archidon 02:00:009
army-status 02:00:010
add spearton 02:00:011
add magikill 02:00:012
add giant 02:00:013
enemy-status 02:00:014
army-status 02:00:015
damage 1 10 02:01:016
damage 2 10 02:01:017
damage 3 20 02:01:018
damage 4 3 02:01:019
damage 5 15 02:01:020
damage 6 1000 02:01:021
damage 7 60 02:01:022
damage 8 16 02:01:023
damage 16 16 02:01:024
enemy-status 02:01:025
army-status 02:01:026
damage 6 17 02:31:027
damage 7 18 02:31:028
damage 8 19 02:31:029
damage 9 20 02:31:030
enemy-status 02:31:031
army-status 02:31:032
```

خروجی - نمونه ۲

```
500
1
2
3
50
4
5
6
7
5 1 1 0 0 0
8
9
10
10000
5 1 1 1 1 1
90
90
80
97
85
dead
20
234
no matter
9970
5 0 1 1 1 1
no matter
2
215
60
7070
0 1 1 1 1
```

٤ قوانین

- پروژه به صورت انفرادی انجام می‌شود.
- نمره پروژه کپی صفر است!

۵ نحوه ارزیابی

- استفاده از طراحی مناسب (کلاس‌ها و توابع متنوع و مناسب)
- رعایت سبك صحیح برنامه‌نویسی
- داشتن قابلیت‌های ذکر شده برای برنامه
- سادگی استفاده و صحت کارکرد برنامه
- خوانایی برنامه و درج مستندات کافی در برنامه
- تکمیل، کافی و دقیق بودن ملزومات خواسته شده
- هر گونه گسترش قابلیت برنامه در جهت افزایش کارائی دارای نمره اضافی خواهد بود.

٦ منبع

ایده و محتوی پروژه از سامانه Quera آورده شده است.

موفق باشید

تابستان پیش‌رو را فرصتی برای
تقویت برنامه‌سازی و مهارت حل مساله قرار دهید.